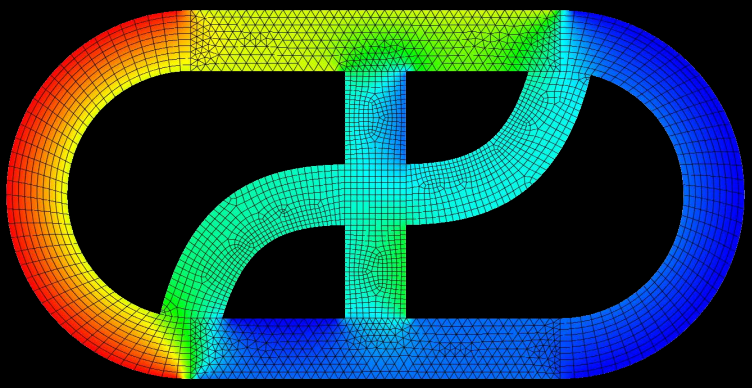


CFD EXPERTS

Simulate the Future

WWW.CFDEXPERTS.NET

Ansys 2021 R2
Engineering What's Ahead.

Ansys CFX-Solver Modeling Guide

Ansys
ANSYS, Inc.
Southpointe
2600 Ansys Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 2021 R2
July 2021

ANSYS, Inc. and Ansys Europe, Ltd. are UL registered ISO 9001: 2015 companies.

Copyright and Trademark Information

Disclaimer Notice

U.S. Government Rights

Third-Party Software

Published in the U.S.A.

Table of Contents

1. Basic Capabilities Modeling 39
1.1. Domains 40
1.2. Physical Models 40
1.2.1. Steady State and Transient Flows 41
1.2.2. Mesh Deformation 42
1.2.2.1. None 42
1.2.2.2. Regions of Motion Specified 42
1.2.2.2.1. Displacement Relative To 43
1.2.2.2.2. Mesh Stiffness 43
1.2.2.2.2.1. Increase Near Small Volumes 44
1.2.2.2.2.2. Increase Near Boundaries 44
1.2.2.2.2.3. Blended Distance and Small Volumes 45
1.2.2.2.2.4. Value (Specified Stiffness) 45
1.2.2.2.3. Mesh Motion Options 45
1.2.2.2.3.1. Conservative Interface Flux 47
1.2.2.2.3.2. Unspecified 47
1.2.2.2.3.3. Stationary 47
1.2.2.2.3.4. Specified Displacement 47
1.2.2.2.3.5. Specified Location 48
1.2.2.2.3.6. Periodic Displacement 48
1.2.2.2.3.7. Parallel to Boundary 50
1.2.2.2.3.8. Surface of Revolution 50
1.2.2.2.3.9. System Coupling 51
1.2.2.2.3.10. Rigid Body Solution 51
1.2.2.3. Periodic Regions of Motion 52
1.2.2.3.1. Periodic Regions of Motion: Complex Displacement 53
1.2.2.3.1.1. Boundary conditions 53
1.2.2.3.1.2. Complex Displacement Solution 53
1.2.2.4. Junction Box Routine 54
1.2.3. Laminar Flow 54
1.2.4. Turbulence and Turbulence Models 54
1.2.5. Heat Transfer 55
1.2.5.1. None 55
1.2.5.2. Isothermal 55
1.2.5.3. Thermal Energy 55
1.2.5.4. Total Energy 55
1.2.5.5. Turbulent Flux Closure 56
1.2.6. Conjugate Heat Transfer 56
1.2.7. Compressible Flow 56
1.2.7.1. Mixed Subsonic/Supersonic Boundaries 57
1.2.8. Setting a Reference Pressure 57
1.2.9. Buoyancy 58
1.2.9.1. Full Buoyancy Model (Density Difference) 59
1.2.9.2. Boussinesq Model 59
1.2.9.3. Buoyancy and Pressure 60
1.2.9.4. Buoyancy In Rotating Domains 60
1.2.10. Immersed Solids 60
1.2.10.1. Immersed Boundary Tracking 62
1.2.10.2. Limitations to using Immersed Solids 62

1.2.11. Multicomponent Flow 64
1.2.11.1. Assumptions About Multicomponent Flow 64
1.2.11.2. Multicomponent Flow Terminology 64
1.2.11.2.1. Pure Substance 65
1.2.11.2.2. Component 65
1.2.11.2.3. Multicomponent Fluid 65
1.2.11.2.4. Fluid 65
1.2.11.2.5. Additional Variable 65
1.2.11.2.6. Ideal Mixture 65
1.2.11.2.7. Transport Equation 66
1.2.11.2.8. Constraint Equation 66
1.2.11.2.9. Algebraic Equation 66
1.2.11.3. Multicomponent Flow Examples 66
1.2.11.3.1. Example 1: Multicomponent Multiphase 66
1.2.11.3.2. Example 2: Smoke in Air 67
1.2.11.4. Component Domain Settings 67
1.2.11.4.1. Algebraic Slip 68
1.2.11.4.2. Kinematic Diffusivity 68
1.2.11.4.3. Turbulent Flux Closure 68
1.2.11.5. Boundary Conditions 69
1.2.11.6. Multicomponent Energy Diffusion 69
1.2.12. Additional Variables 70
1.2.12.1. Kinematic Diffusivity 70
1.2.12.2. Turbulent Flux Closure 71
1.2.12.3. Volumetric and Specific Additional Variable 71
1.2.12.4. Additional Variables In Units Other Than Mass 72
1.2.12.5. Unspecified Additional Variables 72
1.2.12.6. Tensor Type 73
1.2.13. Non-Newtonian Flow 73
1.2.14. Coordinate Frames 76
1.2.14.1. Global Coordinate Frame (Coord 0) 76
1.2.14.2. Local Coordinate Frames 76
1.2.14.3. Cartesian Coordinate Frames 77
1.2.15. Rotating Frames of Reference (RFR) 77
1.2.15.1. Alternate Rotation Model 78
1.2.16. Electric Field 79
1.3. Sources 80
1.3.1. Locators for Sources 80
1.3.1.1. Boundary Sources 80
1.3.1.2. Subdomains 81
1.3.1.3. Injection Regions 81
1.3.1.4. Source Points 81
1.3.2. Types of Sources 81
1.3.2.1. General Sources 81
1.3.2.1.1. Source Coefficient / Total Source Coefficient 82
1.3.2.2. Momentum Sources 82
1.3.2.2.1. Isotropic and Directional Loss Models 82
1.3.2.2.2. General Momentum Source 83
1.3.2.2.2.1. Postprocessing the Momentum Sources 84
1.3.2.2.3. Immersed Solids Sources 84
1.3.2.3. Mass (Continuity) Sources 84

1.3.2.3.1. Mass (Continuity) Source Coefficients 86
1.3.2.4. Bulk Sources 86
1.3.2.5. Solid Sources 86
1.3.2.6. Radiation Sources 87
1.3.2.7. Particle User Sources 87
1.3.3. Multiplying Sources by Porosity 87
1.4. Material Properties 87
1.4.1. CEL Expressions 87
1.4.2. Coordinate Frame 87
1.4.3. Equation of State 87
1.4.3.1. Option 88
1.4.3.1.1. Value 88
1.4.3.1.2. Ideal Gas 88
1.4.3.1.3. Real Gas 89
1.4.3.1.4. IAPWS Library 89
1.4.3.2. Molar Mass 89
1.4.4. Specific Heat Capacity 90
1.4.4.1. Value 90
1.4.4.2. NASA Format 90
1.4.4.3. Zero Pressure Polynomial 91
1.4.4.4. Real Gas 91
1.4.4.5. Reference State Properties 92
1.4.4.5.1. Reference Temperature and Reference Pressure 92
1.4.4.5.2. Reference Specific Enthalpy and Entropy 92
1.4.5. Density and Specific Heat Dependencies 92
1.4.6. Table Generation Pressure and Temperature Limits 92
1.4.7. Transport Properties 93
1.4.7.1. Dynamic Viscosity 93
1.4.7.1.1. Value 93
1.4.7.1.2. Rigid Non Interacting Sphere and Interacting Sphere Models 93
1.4.7.1.3. Non-Newtonian Model 94
1.4.7.1.4. Ideal Mixture 94
1.4.7.1.5. Sutherland's Formula 94
1.4.7.2. Thermal Conductivity 95
1.4.7.2.1. Sutherland's Formula 95
1.4.7.2.2. Modified Euken Model 95
1.4.8. Radiation Properties 95
1.4.9. Buoyancy Properties 96
1.4.9.1. Thermal Expansivity 96
1.4.10. Electromagnetic Properties 96
1.4.10.1. Electrical Conductivity 96
1.4.10.2. Magnetic Permeability 96
1.4.11. Library Materials 97
1.4.11.1. Adding to the MATERIALS file 97
1.5. Mixture Properties (Fixed, Variable, Reacting) 98
1.5.1. Equation of State/Density 98
1.5.2. Molar Mass 100
1.5.3. Specific Heat Capacity 100
1.5.4. Electromagnetic Properties 101
1.6. Efficiency Calculation 101
1.6.1. Isentropic Efficiency and Total Enthalpy 101

1.6.2. Polytropic Efficiency 103
1.6.3. Activating Efficiency Output 105
1.6.3.1. CFX-Solver Manager Output Variables 105
1.6.3.2. Results File Output Variables 106
1.6.4. Restrictions 106
1.7. Wall Condensation Model 107
1.7.1. CFX-Pre Set-up 107
1.7.1.1. Domain Fluid Model Specification 108
1.7.1.2. Boundary Condition Specification 108
1.7.1.3. Restrictions 108
1.7.1.4. Convergence Tip 109
1.7.2. Condensation Mass Flux in CFD-Post 109
2. Boundary Condition Modeling 111
2.1. The Purpose of Boundary Conditions 111
2.2. Available Boundary Conditions 112
2.2.1. Fluid Boundaries 112
2.2.2. Solid Boundaries 113
2.3. Using Boundary Conditions 113
2.3.1. Specifying Well-Posed Boundary Conditions 113
2.3.2. Recommended Configurations of Boundary Conditions 114
2.3.3. Using Inlets, Outlets and Openings 114
2.3.3.1. Inlets 114
2.3.3.2. Outlets 115
2.3.3.3. Openings 116
2.3.3.4. Using Pressure Specified Boundaries with Buoyant Flows 118
2.3.4. Using CEL Expressions With Boundary Conditions 118
2.4. Inlet 118
2.4.1. Mesh Motion 119
2.4.2. Inlet (Subsonic) 119
2.4.2.1. Mass and Momentum 119
2.4.2.1.1. Normal Speed 119
2.4.2.1.2. Cartesian Velocity Components 119
2.4.2.1.3. Cylindrical Velocity Components 120
2.4.2.1.4. Mass Flow Rate 120
2.4.2.1.5. Total Pressure (Stable) 121
2.4.2.1.6. Stationary Frame Total Pressure (Stable) 121
2.4.2.1.7. Static Pressure 121
2.4.2.1.8. Fluid Velocity 121
2.4.2.2. Flow Direction 122
2.4.2.3. Turbulence 122
2.4.2.3.1. Default Intensity and Autocompute Length Scale 123
2.4.2.3.2. Intensity and Autocompute Length Scale 123
2.4.2.3.3. Intensity and Length Scale 123
2.4.2.3.4. Low (Intensity = 1%) 123
2.4.2.3.5. Medium (Intensity = 5%) 123
2.4.2.3.6. High (Intensity = 10%) 123
2.4.2.3.7. Specified Intensity and Eddy Viscosity Ratio 123
2.4.2.3.8. k and Epsilon 123
2.4.2.3.9. Zero Gradient 124
2.4.2.4. Heat Transfer 124
2.4.2.4.1. Static Temperature 124

2.4.2.4.2. Total Temperature 124
2.4.2.4.3. Stat. Frame Total Temperature 124
2.4.2.4.4. Total Enthalpy 124
2.4.2.4.5. Stationary Frame Total Enthalpy 124
2.4.2.5. Thermal Radiation 124
2.4.2.5.1. Radiative Heat Flux (P1 Model) 125
2.4.2.5.2. Radiation Intensity (P1 Model) 125
2.4.2.5.3. External Blackbody Temperature 125
2.4.2.5.4. Local Temperature 125
2.4.2.5.5. Sources (Discrete Transfer and Monte Carlo models) 125
2.4.2.6. Transported Additional Variables at an Inlet 125
2.4.3. Inlet (Supersonic) 126
2.4.3.1. Mass and Momentum 126
2.4.4. Inlet (Mixed Subsonic-Supersonic) 127
2.4.4.1. Supported Material Types 128
2.4.4.2. Mass and Momentum 128
2.4.4.2.1. Cartesian Velocity Components and (Total) Pressure 128
2.4.4.2.2. Cylindrical Velocity Components and (Total) Pressure 129
2.4.4.2.3. Normal Speed and (Total) Pressure 129
2.4.4.3. Heat Transfer 129
2.4.4.3.1. Static Temperature 129
2.4.4.3.2. Total Temperature 129
2.4.4.3.3. Total Enthalpy 129
2.4.4.4. Initial Guess Recommendation 129
2.5. Outlet 129
2.5.1. Mass and Momentum 130
2.5.1.1. Static Pressure 130
2.5.1.2. Normal Speed 130
2.5.1.3. Cartesian Velocity Components 130
2.5.1.4. Cylindrical Velocity Components 131
2.5.1.5. Average Static Pressure 131
2.5.1.5.1. Average Over Whole Outlet 131
2.5.1.5.2. Average Above Specified Radius 131
2.5.1.5.3. Average Below Specified Radius 131
2.5.1.5.4. Circumferential 132
2.5.1.5.5. Radial Equilibrium 132
2.5.1.6. Mass Flow Rate (Bulk Mass Flow Rate for Multiphase) 133
2.5.1.7. Exit Corrected Mass Flow Rate 133
2.5.1.8. Mass Flow Outlet Constraint 134
2.5.1.8.1. Pressure Shape Unconstrained 135
2.5.1.8.2. Uniform Mass Flux 135
2.5.1.8.3. Pressure Shape Constrained 135
2.5.1.8.3.1. Circumferential Pressure Averaging 136
2.5.1.9. Degassing Condition (Multiphase only) 136
2.5.1.10. Fluid Velocity (Multiphase only) 136
2.5.1.11. Supercritical (Multiphase only) 136
2.5.2. Turbulence, Heat Transfer, and Additional Variables 137
2.5.3. Thermal Radiation 137
2.5.4. Mesh Motion 137
2.5.5. Outlet (Supersonic) 137
2.6. Opening 137

2.6.1. Mass and Momentum 138
2.6.1.1. Cartesian Velocity Components 138
2.6.1.2. Cylindrical Velocity Components 138
2.6.1.3. Opening Pressure and Direction 138
2.6.1.4. Static Pressure and Direction 138
2.6.1.5. Entrainment 139
2.6.1.6. Fluid Velocity 139
2.6.2. Loss Coefficient 139
2.6.2.1. For a Pressure-Specified Opening 139
2.6.2.2. For a Static-Pressure-Specified Opening 140
2.6.3. Heat Transfer 140
2.6.4. Turbulence 140
2.6.5. Thermal Radiation 140
2.6.6. Additional Variables 140
2.6.7. Mesh Motion 140
2.7. Wall 140
2.7.1. Mass and Momentum 141
2.7.1.1. No Slip Wall 141
2.7.1.2. Free Slip Wall 142
2.7.1.3. Finite Slip Wall 142
2.7.1.4. Specified Shear 143
2.7.1.5. Counter-rotating Wall 143
2.7.1.6. Rotating Wall 143
2.7.2. Wall Roughness 143
2.7.3. Wall Contact Model 143
2.7.4. Wall Adhesion 144
2.7.5. Heat Transfer 144
2.7.5.1. Adiabatic 144
2.7.5.2. Fixed Temperature 144
2.7.5.3. Heat Flux and Wall Heat Flux 145
2.7.5.4. Heat Transfer Coefficient and Wall Heat Transfer Coefficient 145
2.7.5.5. System Coupling 146
2.7.5.6. Results File Variables for Postprocessing 146
2.7.5.6.1. Wall Temperature (T_w) 146
2.7.5.6.2. Wall Heat Flux and Heat Flux (q_w) 147
2.7.5.6.3. Wall Heat Transfer Coefficient (h_c) and Wall Adjacent Temperature (T_{nw}) 147
2.7.5.6.4. Wall External Heat Transfer Coefficient and Wall External Temperature 148
2.7.6. Mesh Motion 148
2.7.7. Thermal Radiation 148
2.7.7.1. Opaque 148
2.7.7.2. Sources 148
2.7.8. Equations Governing Additional Variables 149
2.8. Symmetry Plane 149
2.8.1. Mesh Motion 150
2.9. Profile Boundary Conditions 150
2.9.1. Using a Profile From One Location at Another Location 151
2.9.2. Standard Variable Names 152
2.9.3. Non-Standard Variable Names 153
2.9.4. Custom Variables 153
2.9.5. Using r-Theta Profiles 153
2.9.6. Data Interpolation Method 153

2.9.7. Extracting Profile Data from Results Files 154
2.10. General Non-Reflecting Boundary Conditions 155
2.10.1. Overview 156
2.10.2. Restrictions and Limitations 156
2.10.3. Theory 156
2.10.4. Acoustic Reflectivity Settings in CFX-Pre 159
2.11. Limitations 159
3. Initial Condition Modeling 161
3.1. Setting the Initial Conditions in CFX-Pre 162
3.1.1. Automatic 162
3.1.2. Automatic with Value 163
3.1.3. Using Expressions with Initial Conditions 163
3.2. Initialization Parameters 163
3.2.1. Coordinate Frame 164
3.2.2. Frame Type 165
3.2.2.1. Considerations for Multiple Domains and Global Initialization 165
3.2.3. Velocity Type 165
3.2.3.1. Cartesian Coordinate Frame, Cartesian Velocity Components 165
3.2.3.2. Cartesian Coordinate Frame, Cylindrical Velocity Components 165
3.2.3.3. Cylindrical Coordinate Frame, Cartesian Velocity Components 165
3.2.3.4. Cylindrical Coordinate Frame, Cylindrical Velocity Components 166
3.2.4. Cartesian Velocity Components 166
3.2.4.1. Automatic Values 166
3.2.4.2. Recommended Values 167
3.2.5. Cylindrical Velocity Components 168
3.2.5.1. Automatic Values 168
3.2.5.2. Recommended Values 168
3.2.6. Velocity Scale 168
3.2.7. Velocity Fluctuation 168
3.2.8. Static Pressure 169
3.2.8.1. Automatic Values 169
3.2.8.2. Recommended Values 169
3.2.9. Temperature 169
3.2.9.1. Automatic Values 169
3.2.9.2. Recommended Values 169
3.2.10. K (Turbulent Kinetic Energy) 169
3.2.10.1. Automatic Values 169
3.2.10.2. Recommended Values 170
3.2.11. Epsilon (Turbulence Eddy Dissipation) 170
3.2.11.1. Automatic Values 170
3.2.11.2. Recommended Values 170
3.2.11.3. Manual Specification 170
3.2.12. Omega (Turbulence Eddy Frequency) 171
3.2.12.1. Automatic Values 171
3.2.12.2. Recommended Values 171
3.2.12.3. Manual Specification 171
3.2.13. Reynolds Stress Components 171
3.2.14. Initialization of Additional Variables 172
3.2.15. Component 172
3.2.16. Volume Fraction 172
3.2.17. Radiation Intensity 172

3.2.18. Initialization of Solid Domains 172
3.2.19. Initial Conditions for a Multiphase Simulation 173
3.2.19.1. Volume Fraction 173
3.2.19.2. Velocity 173
3.2.20. Initialization Advice 173
3.3. Reading the Initial Conditions from a File 174
3.3.1. Using Configuration Results to Provide Initial Values 175
3.3.2. Continuing the History 175
3.3.3. Using Multiple Files to Provide Initial Conditions 179
3.3.4. Using the Mesh from the Initial Values File 179
3.3.5. Using an Initial Values File that Contains Particles 180
3.4. Using the CFX-Interpolator 182
3.4.1. Interpolating from a Single File 183
3.4.1.1. Mapping Data from the Source File to the Target File 185
3.4.2. Interpolating from Multiple Files 188
3.4.3. Interpolation Mapping 190
3.4.3.1. Defining Transformations for use in Interpolation Mapping Alignment Transformations 191
3.4.3.2. Creating Interpolation Mapping Objects 192
3.4.4. Adjusting the Bounding Box Tolerance 193
3.4.5. Interpolating Onto a Solver Input File with Results Fields 195
3.4.6. Miscellaneous Limitations of the CFX-Interpolator 195
3.4.7. Using the CFX-Interpolator to Calculate Difference Variables 196
4. Turbulence and Near-Wall Modeling 197
4.1. Turbulence Models 198
4.1.1. The Laminar Model 198
4.1.2. The Zero Equation Model 199
4.1.3. The k-epsilon Model 199
4.1.4. The RNG k-epsilon Model 199
4.1.5. The k-omega and SST Models 200
4.1.5.1. GEKO model 201
4.1.6. Curvature Correction for Two-Equation Models 202
4.1.7. Corner Correction 202
4.1.8. The Reynolds Stress Model 203
4.1.9. Omega-Based Reynolds Stress Models 204
4.1.10. Explicit Algebraic Reynolds Stress Model 205
4.1.11. Ansys CFX Laminar-Turbulent Transition Models 205
4.1.11.1. Estimating when the Transition Model Should be Used 206
4.1.11.2. Grid Requirements 207
4.1.11.3. Specifying Inlet Turbulence Levels 211
4.1.11.4. Summary 213
4.1.12. The Large Eddy Simulation Model (LES) 213
4.1.12.1. Using the LES model in CFX 213
4.1.12.2. Introduction to LES 214
4.1.12.3. When to use LES 214
4.1.12.4. Setting up an LES Simulation 215
4.1.12.4.1. Geometry for LES 215
4.1.12.4.2. Meshing 215
4.1.12.4.3. Boundary Layers 215
4.1.12.4.4. Analysis Type 215
4.1.12.4.5. Domains 215

4.1.12.4.6. LES Boundary Conditions: Inlet 215
4.1.12.4.7. LES Boundary Conditions: Outlets and Openings 216
4.1.12.4.8. LES Initialization 216
4.1.12.4.9. LES Solver Control 217
4.1.12.4.10. LES Timestep Considerations 217
4.1.12.5. Solver Memory 217
4.1.12.6. Useful Values For LES Runs 217
4.1.12.6.1. Statistical Reynolds Stresses 218
4.1.12.6.2. Delaying the Start of Reynolds Stress Calculations 218
4.1.13. The Detached Eddy Simulation Model (DES) 218
4.1.13.1. Using the Detached Eddy Simulation Model in CFX 219
4.1.13.1.1. When to use DES 219
4.1.13.2. Setting up a DES Simulation 219
4.1.13.2.1. Geometry for DES 219
4.1.13.2.2. Meshing Requirements for DES 219
4.1.13.2.3. DES Timestep Considerations 220
4.1.13.2.4. Boundary Conditions 220
4.1.13.2.5. DES Initialization 220
4.1.13.2.6. Monitoring a DES Simulation 220
4.1.13.3. Limitations/Concerns of Using the DES Model 222
4.1.14. The Stress-Blended Eddy Simulation (SBES) Model 223
4.1.15. The Scale-Adaptive Simulation (SAS) 224
4.1.15.1. Using the Scale Adaptive Simulation model in CFX 226
4.1.16. Buoyancy Turbulence 226
4.2. Modeling Flow Near the Wall 226
4.2.1. Standard Wall Functions 228
4.2.2. Scalable Wall Functions 228
4.2.3. Automatic Near-Wall Treatment for Omega-Based Models 228
4.2.4. Treatment of Rough Walls 228
4.2.5. Solver Yplus and Yplus 229
4.2.6. Guidelines for Mesh Generation 229
4.2.6.1. Minimum Node Spacing 229
4.2.6.1.1. Determination of the Near Wall Spacing 230
4.2.6.2. Minimum Number of Nodes 230
4.2.6.2.1. Goal 230
4.2.6.2.2. Formulation 231
5. Domain Interface Modeling 233
5.1. Overview of Domain Interfaces 233
5.2. Interface Type 234
5.3. Interface Models 234
5.3.1. Translational Periodicity 235
5.3.2. Rotational Periodicity 235
5.3.3. General Connection 236
5.3.3.1. Frame Change/Mixing Model 237
5.3.3.1.1. None 237
5.3.3.1.2. Frozen Rotor 237
5.3.3.1.2.1. Rotational Offset 237
5.3.3.1.3. Stage (Mixing Plane) 238
5.3.3.1.3.1. Pressure Profile Decay 239
5.3.3.1.3.2. Downstream Velocity Constraint 239
5.3.3.1.3.3. Implicit Stage Averaging 239

5.3.3.1.4. Transient Rotor-Stator 240
5.3.3.2. Pitch Change 240
5.3.3.2.1. None 242
5.3.3.2.2. Automatic 242
5.3.3.2.3. Value 243
5.3.3.2.4. Specified Pitch Angles 243
5.3.4. Mass and Momentum Models 243
5.3.4.1. Translational Periodicity 243
5.3.4.2. General Connection 244
5.3.4.2.1. Specified Pressure Change 244
5.3.4.2.2. Specified Mass Flow Rate 244
5.3.4.3. Further Comments 245
5.4. Mesh Connection Options 245
5.4.1. Automatic Connections 245
5.4.2. Direct (One-to-One) Connections 246
5.4.3. GGI (General Grid Interface) Connections 247
5.4.3.1. Non-overlap Boundary Conditions 247
5.4.3.2. Conditional Connections 248
5.4.4. Mesh Connection Recommendations 248
5.5. Defining Domain Interfaces as Thin Surfaces 249
5.5.1. Modeling Thin Surfaces: Overview 249
5.6. Recommendations For Using Domain Interfaces 250
5.6.1. Using Domain Interfaces 250
5.6.2. Using Multiple Domain Interfaces 250
5.6.3. Using Domain Interfaces in Turbomachinery Applications 251
5.6.3.1. Case 1: Impeller/Volute 251
5.6.3.2. Case 2: Step change between rotor and stator 252
5.6.3.3. Case 3: Blade Passage at or close to the edge of a domain 252
5.6.3.4. Case 4: Blade Passage at or close to the edge of a domain 253
5.6.3.5. Case 5: Blade with thick trailing edge 254
5.7. Automatic Creation and Treatment of Domain Interfaces 254
6. Turbomachinery Blade Row Modeling 257
6.1. Transient Blade Row Modeling 257
6.1.1. Transient Blade Row Modeling Terminology 258
6.1.1.1. Abbreviations Used in this Document 259
6.1.2. Setting up a Transient Blade Row Model 259
6.1.2.1. Setting up Monitors to Check Results 261
6.1.3. Running and Postprocessing a Simulation that uses a Transient Blade Row Model 262
6.1.3.1. Stopping and then Restarting Simulations with an Increased Number of Time Steps Per Period 262
6.1.4. Profile Transformation 262
6.1.5. Time Transformation 263
6.1.5.1. Time Transformation: Workflow Requirements 263
6.1.6. Fourier Transformation 265
6.1.6.1. Fourier Transformation: Workflow Requirements 265
6.1.7. Guidelines for Using Transient Blade Row Features 266
6.1.7.1. General Setup Guidelines 267
6.1.7.2. Guidelines for using the Time Transformation Feature 267
6.1.7.3. Guidelines for using the Fourier Transformation Feature 268
6.1.7.4. General Postprocessing Guidelines 269
6.1.8. Use Cases 269

6.1.8.1. Case 1: Transient Rotor Stator Single Stage 269
6.1.8.1.1. Profile Transformation and Fourier Transformation using Harmonic Analysis 271
6.1.8.2. Case 2: Flow Boundary Disturbance 273
6.1.8.2.1. Flow Boundary Disturbance using Harmonic Analysis 274
6.1.8.2.2. Multiple Disturbances 276
6.1.8.3. Case 3: Blade Flutter 277
6.1.8.3.1. Setting Up a Blade Flutter Simulation 281
6.1.8.3.2. Running a Blade Flutter Simulation 282
6.1.8.3.3. Blade Flutter using Harmonic Analysis 283
6.1.8.4. Case 4: Harmonic Forced Response 284
6.1.8.5. Case 5: Transient Rotor Stator Multi-Stage Cases 286
6.1.8.5.1. Modeling a Single-pitch-ratio Multistage Turbomachine using the Time Transformation Method 287
6.1.8.5.2. Modeling 1.5 Stages with Two Different Pitch Ratios using the Time Transformation Method 288
6.1.8.5.3. Modeling a multistage turbomachine with Time Transformation TRS and other interfaces: Profile Transformation & Stage 289
6.1.8.5.4. Modeling a multistage turbomachine with Time Transformation TRS and single-sided Time Transformation interfaces (STT-TRS) 290
6.1.8.6. Case 6: Transient Rotor Stator Cases with Asymmetric Flow 292
6.1.8.6.1. Modeling an Impeller in a Vaneless Volute 292
6.1.8.6.2. Modeling an Impeller in a Vaned Volute 293
6.2. Blade Film Cooling 293
6.2.1. Options for Modeling Blade Film Cooling 294
7. Multiphase Flow Modeling 295
7.1. Multiphase Terminology 296
7.1.1. Multiphase Flow 296
7.1.2. Eulerian-Eulerian 297
7.1.3. Inhomogeneous Multiphase Flow 297
7.1.4. Homogeneous Multiphase Flow 297
7.1.5. Multicomponent Multiphase Flow 297
7.1.6. Volume Fraction 297
7.1.7. Free Surface Flow 298
7.1.8. Surface Tension 298
7.2. Multiphase Examples 298
7.2.1. Water Droplets in Air 298
7.2.2. Air Bubbles in Water 298
7.2.3. Gas-Solid and Liquid-Solid Flow 298
7.2.4. Three-Phase Flow 299
7.2.5. Polydispersed Flow 299
7.3. Eulerian-Eulerian Multiphase Versus Particle Transport 299
7.4. Specifying Fluids for Multiphase Flow 300
7.4.1. Morphology 300
7.4.1.1. Continuous Fluid 301
7.4.1.2. Dispersed Fluid 301
7.4.1.3. Dispersed Solid 301
7.4.1.4. Particle Transport Fluid 301
7.4.1.5. Particle Transport Solid 301
7.4.1.6. Polydispersed Fluid 301
7.4.1.7. Droplets (Phase Change) 301
7.4.2. Mean Diameter 301

7.4.3. Minimum Volume Fraction 302
7.4.4. Maximum Packing 302
7.5. The Homogeneous and Inhomogeneous Models 302
7.5.1. The Inhomogeneous (Interfluid Transfer) Model 302
7.5.1.1. The Particle Model 302
7.5.1.2. The Mixture Model 303
7.5.1.3. The Free Surface Model 303
7.5.2. The Homogeneous Model 303
7.6. Buoyancy in Multiphase Flow 304
7.6.1. Fluid Buoyancy Model 304
7.6.1.1. Density Difference 304
7.6.1.2. Boussinesq 305
7.7. Multicomponent Multiphase Flow 305
7.8. Interphase Momentum Transfer Models 305
7.8.1. Interphase Drag 305
7.8.1.1. Interphase Drag for the Particle Model 305
7.8.1.1.1. Specifying a Drag Coefficient 306
7.8.1.1.2. Sparsely Distributed Solid Particles 306
7.8.1.1.2.1. Sparsely Distributed Solid Particles: Schiller Naumann Drag Model 306
7.8.1.1.3. Densely Distributed Solid Particles 306
7.8.1.1.3.1. Densely Distributed Solid Particles: Wen Yu Drag Model 306
7.8.1.1.3.2. Densely Distributed Solid Particles: Gidaspow Drag Model 306
7.8.1.1.4. Sparsely Distributed Fluid Particles (drops and bubbles) 307
7.8.1.1.4.1. Sparsely Distributed Fluid Particles: Ishii-Zuber Drag Model 307
7.8.1.1.4.2. Sparsely Distributed Fluid Particles: Grace Drag Model 307
7.8.1.1.4.3. Sparsely Distributed Fluid Particles: Availability 307
7.8.1.1.5. Densely Distributed Fluid Particles 307
7.8.1.1.5.1. Densely Distributed Fluid Particles: Ishii-Zuber Drag Model 307
7.8.1.1.5.2. Densely Distributed Fluid Particles: Grace Drag Model 308
7.8.1.2. Interphase Drag for the Mixture Model 308
7.8.2. Lift Force 308
7.8.3. Virtual Mass Force 309
7.8.4. Wall Lubrication Force 309
7.8.5. Interphase Turbulent Dispersion Force 310
7.8.5.1. Favre Averaged Drag Model 310
7.8.5.2. Lopez de Bertodano Model 310
7.9. Solid Particle Collision Models 311
7.9.1. Solid Pressure Force Model 311
7.9.2. Maximum Packing 312
7.9.3. Kinetic Theory Models 312
7.10. Interphase Heat Transfer 312
7.10.1. Inhomogeneous Interphase Heat Transfer Models 312
7.10.1.1. Particle Model Correlations for Overall Heat Transfer Coefficient 313
7.10.1.2. Mixture Model Correlations for Overall Heat Transfer Coefficient 314
7.10.1.3. Two Resistance Model for Fluid Specific Heat Transfer Coefficients 314
7.10.2. Homogeneous Heat Transfer in Multiphase Flow 315
7.11. Polydispersed, Multiple Size Group (MUSIG) Model 315
7.11.1. Setting up a Polydispersed (MUSIG or IMUSIG) Simulation 316
7.11.1.1. Creating a Polydispersed (MUSIG) Fluid 317
7.11.1.2. Boundary Conditions 317
7.11.1.3. Initial Conditions 318

7.11.1.4. Sources 318
7.11.1.5. Postprocessing Variables 318
7.11.2. MUSIG Modeling Advice 319
7.12. Turbulence Modeling in Multiphase Flow 319
7.12.1. Phase-Dependent Turbulence Models 319
7.12.1.1. Algebraic Models 320
7.12.1.2. Two Equation Models 320
7.12.1.3. Reynolds Stress Models 320
7.12.2. Homogeneous Turbulence in Inhomogeneous Flow 320
7.12.3. Turbulence Enhancement 320
7.12.4. Turbulence in Homogeneous Multiphase Flow 321
7.13. Additional Variables in Multiphase Flow 321
7.13.1. Additional Variable Interphase Transfer Models 321
7.13.1.1. Particle Model Correlations 321
7.13.1.1.1. Ranz-Marshall Correlation 321
7.13.1.1.2. Hughmark Correlation 321
7.13.1.1.3. Sherwood Number 322
7.13.1.1.4. Additional Variable Transfer Coefficient 322
7.13.1.1.5. Interface Flux 322
7.13.1.2. Mixture Model Correlations 322
7.13.2. Homogeneous Additional Variables in Multiphase Flow 322
7.14. Sources in Multiphase Flow 322
7.14.1. Fluid-Specific Sources 323
7.14.2. Bulk Sources 323
7.15. Interphase Mass Transfer 323
7.15.1. Double Precision Solver 324
7.15.2. User Specified Mass Transfer 324
7.15.3. Thermal Phase Change Model 324
7.15.3.1. Saturation Temperature 324
7.15.3.2. Wall Boiling Model 325
7.15.3.2.1. RPI Model 325
7.15.3.2.2. Using a Wall Boiling Model 328
7.15.3.3. Latent Heat 329
7.15.3.4. Heat Transfer Models 330
7.15.3.5. Interphase Heat Transfer Correlations 330
7.15.3.6. Modeling Advice 330
7.15.3.6.1. Saturated Vapor Bubbles in Subcooled or Superheated Liquid 330
7.15.3.6.2. Subcooled or Superheated Droplets in Saturated Vapor 331
7.15.3.6.3. Superheated Vapor Bubbles in Liquid 331
7.15.3.6.4. Thermal Energy and Total Energy Models 331
7.15.4. Cavitation Model 332
7.15.4.1. Rayleigh Plesset Model 333
7.15.4.2. User Defined Cavitation Models 334
7.15.5. Interphase Species Mass Transfer 335
7.15.5.1. Component Pairs 335
7.15.5.2. Two Resistance Model 335
7.15.5.3. Single Resistance Model 336
7.15.5.4. Interfacial Equilibrium Models 336
7.15.5.4.1. Liquid Evaporation (Raoult's Law) 336
7.15.5.4.2. Gas Absorption / Dissolution (Henry's Law) 336
7.15.5.4.3. Other Situations 337

7.15.5.5. Species Mass Transfer Coefficients 337
7.15.5.6. Modeling Advice 338
7.15.5.6.1. Liquid Evaporation 338
7.15.5.6.2. Gas Dissolution 338
7.15.5.6.3. Mixture Properties 338
7.15.5.6.4. Current Limitations 338
7.15.6. Droplet Condensation Model 339
7.16. Boundary Conditions in Multiphase Flow 340
7.16.1. Wall Boundaries in Multiphase 341
7.16.1.1. Bulk Wall Boundary Conditions 341
7.16.1.1.1. Area Contact Model 341
7.16.1.2. Fluid Dependent Wall Boundary Conditions 342
7.16.1.3. Wall Deposition 343
7.16.2. Mass Flow Inlet 343
7.16.3. Mass Flow Outlet 343
7.17. Modeling Advice for Multiphase Flow 344
7.17.1. Turbulence Models 344
7.17.2. Minimum Volume Fraction Setting 344
7.17.3. Buoyancy 345
7.17.4. Initial Conditions 345
7.17.5. Timestepping 345
7.17.6. Convergence 345
7.17.7. Transient Simulations 345
7.18. Free Surface Flow 346
7.18.1. Interface Compression Level 346
7.18.2. Supercritical and Subcritical Flow 346
7.18.3. Multiphase Model Selection 346
7.18.4. Surface Tension 347
7.18.4.1. Background 347
7.18.4.2. Discretization Options 347
7.18.4.2.1. Volume Fraction Smoothing Type 347
7.18.4.2.2. Curvature Under-Relaxation Factor 347
7.18.4.3. Initial Conditions 348
7.18.4.4. Wall Adhesion 348
7.18.5. Modeling Advice for Free Surface Flow 348
7.18.5.1. Domains 348
7.18.5.2. Turbulence Model 348
7.18.5.3. Boundary Conditions 349
7.18.5.3.1. Pressure Reference 349
7.18.5.3.2. Outlets 349
7.18.5.4. Initial Conditions 349
7.18.5.5. Timestep 350
7.18.5.6. Mesh Adaption 350
7.18.5.7. Body Forces 350
7.18.5.8. Convergence 350
7.18.5.9. Parallel 350
7.19. Algebraic Slip Model (ASM) 350
7.19.1. Algebraic Slip Model Specification 351
7.19.1.1. Fluid Models 351
7.19.1.2. Wall Deposition 351
7.19.1.3. Limitations 351

7.20. Multiphase Flow Restrictions 352
8. Particle Transport Modeling 353
8.1. Model Validity 354
8.2. Particle Transport Versus Eulerian-Eulerian Multiphase 354
8.3. Forces Acting on the Particles 355
8.3.1. Drag Force 355
8.3.2. Reference Frame Rotational Forces 355
8.3.3. Buoyancy Force 355
8.4. Creating Particle Materials 355
8.5. Particle Domain Options 356
8.5.1. Basic Settings 356
8.5.1.1. Particle Morphology Options 356
8.5.2. Fluid Models 356
8.5.2.1. Multiphase Reactions 356
8.5.2.2. Buoyancy for Particles 356
8.5.2.3. Turbulence for Particles 357
8.5.2.4. Heat Transfer for Particles 357
8.5.2.5. Radiation for Particles 357
8.5.3. Fluid Specific Models 357
8.5.3.1. Particle Diameter Distribution 357
8.5.3.1.1. Specified Diameter 357
8.5.3.1.2. Uniform in Diameter by Number 357
8.5.3.1.3. Uniform in Diameter by Mass 358
8.5.3.1.4. Normal in Diameter by Number 358
8.5.3.1.5. Normal in Diameter by Mass 358
8.5.3.1.6. Rosin Rammler 358
8.5.3.1.7. Nukiyama Tanasawa 359
8.5.3.1.8. Discrete Diameter Distribution 360
8.5.3.2. Particle Shape Factors 362
8.5.3.3. Particle Diameter Change Due to Swelling 362
8.5.3.4. Heat Transfer 362
8.5.3.5. Erosion 362
8.5.3.5.1. Finnie 362
8.5.3.5.2. Tabakoff 363
8.5.3.5.3. User Defined 364
8.5.3.6. Particle-Rough Wall Model (Virtual Wall Model) 364
8.5.3.6.1. Sommerfeld-Frank Model 364
8.5.3.7. Particle Breakup Model 364
8.5.3.8. Particle Collision Model 365
8.5.3.8.1. Requirements for the Applicability of Particle-Particle Collision Model 366
8.5.3.9. Fluid Buoyancy Model 367
8.5.4. Fluid Pairs 367
8.5.4.1. Particle Fluid Pair Coupling Options 367
8.5.4.2. Drag Force for Particles 368
8.5.4.2.1. Particle User Source Example 368
8.5.4.2.2. Linearization Blend Factor 370
8.5.4.3. Particle User Source 370
8.5.4.4. Non-Drag Forces 371
8.5.4.4.1. Virtual Mass Force 371
8.5.4.4.2. Turbulent Dispersion Force 371
8.5.4.4.3. Pressure Gradient Force 371

8.5.4.5. Interphase Heat Transfer 371
8.5.4.5.1. Particle User Source 371
8.5.4.6. Interphase Radiation Transfer 372
8.5.4.6.1. Opaque 372
8.5.4.6.2. Blended Particle Emissivity 372
8.5.4.7. Mass Transfer 372
8.5.4.7.1. Ranz Marshall 372
8.5.4.7.2. Liquid Evaporation Model 372
8.5.4.7.3. Liquid Evaporation Model: Spray Dryer with Droplets Containing a Solid Substrate 374
8.5.4.7.4. Liquid Evaporation Model: Oil Evaporation/Combustion 374
8.5.4.7.5. Latent Heat 374
8.5.4.8. Particle User Sources 374
8.5.4.8.1. Particle User Source example 375
8.5.5. Particle Injection Regions 378
8.6. Particle Boundary Options and Behavior 378
8.6.1. Inlet/Opening Boundaries 378
8.6.1.1. Mass and Momentum 379
8.6.1.2. Particle Position 379
8.6.1.2.1. Uniform Injection 379
8.6.1.2.2. Uniform Injection within Annulus 379
8.6.1.2.3. Injection With Line Weighting 380
8.6.1.2.4. Injection With Point Weighting 380
8.6.1.2.5. Injection With Circular Weighting 380
8.6.1.2.6. Injection With User Defined Weighting 380
8.6.1.2.7. Injection at Face Centers 380
8.6.1.2.8. Injection at IP Face Centers 380
8.6.1.2.9. Number of Positions 381
8.6.1.2.10. Point Data Format 381
8.6.1.3. Particle Locations 381
8.6.1.4. Particle Diameter Distribution 381
8.6.1.5. Particle Mass Flow Rate 382
8.6.1.6. Heat Transfer 382
8.6.1.7. Component Details 382
8.6.1.8. Particle Actions at Inlets and Openings 382
8.6.2. Outlet Boundaries 382
8.6.2.1. Particle Actions at Outlets 382
8.6.3. Wall Boundaries 383
8.6.3.1. Wall Interaction 383
8.6.3.1.1. Standard Particle-Wall Interaction 383
8.6.3.1.1.1. Restitution Coefficients for Particles 383
8.6.3.1.2. Wall Film Modeling 384
8.6.3.1.2.1. User Defined Wall Film Modeling 384
8.6.3.1.3. User Defined Particle-Wall Interaction 390
8.6.3.2. Erosion Model 390
8.6.3.3. Particle-Rough Wall Model (Virtual Wall Model) 390
8.6.3.4. Particle Breakup 390
8.6.3.5. Mass Flow Absorption 391
8.6.3.6. Mass and Momentum 391
8.6.3.7. Particle Impact Angle 391
8.6.3.8. Particle Position 391

8.6.3.9. Particle Diameter Distribution 391
8.6.3.10. Particle Mass Flow Rate 391
8.6.4. Symmetry Plane Boundaries 391
8.6.5. Interface Boundaries 391
8.6.6. Domain Interfaces 392
8.6.6.1. Fluid-Fluid 392
8.6.6.1.1. Frame Change Option = None 392
8.6.6.1.2. Frame Change Option = Frozen Rotor 392
8.6.6.1.3. Frame Change Option = Stage (Mixing-Plane) 392
8.6.6.1.4. Frame Change Option = Transient Rotor Stator 392
8.6.6.2. Periodic Connections 393
8.7. Subdomains 393
8.8. Particle Injection Regions 393
8.8.1. Sphere 394
8.8.2. Cone 394
8.8.2.1. Injection Velocity 395
8.8.2.2. An Example of a Point Cone 395
8.8.2.3. An Example of a Point Cone Using the Dispersion Angle 395
8.8.2.4. An Example of a Hollow Cone using the Dispersion Angle 396
8.8.3. Cone with Primary Breakup 397
8.8.4. User Defined Injection Regions 397
8.9. Particle Output Control 398
8.9.1. Transient Particle Diagnostics 398
8.9.1.1. User Diagnostics Routine 400
8.9.1.1.1. Example User Routine: CCL 401
8.9.1.1.2. Example User Routine: Mainline Routine 401
8.9.1.1.3. Example User Routine: Subroutine 402
8.9.1.1.4. Example: Complete CCL 402
8.9.1.2. Particle Track Output 404
8.9.2. List of Particle Variables 405
8.10. Particle Solver Control 405
8.10.1. Particle Coupling Control 405
8.10.1.1. First Iteration for Particle Calculation 405
8.10.1.2. Iteration Frequency 405
8.10.1.3. Particle Source Change Target 406
8.10.2. Particle Under-Relaxation Factors 406
8.10.2.1. Under-Relaxation Factor for Velocity, Energy, and Mass 407
8.10.2.2. Under-Relaxation Factor for First Particle Integration 407
8.10.2.3. Under-Relaxation at Time Step Start 407
8.10.3. Particle Integration 407
8.10.3.1. Number of Integration Steps Per Element 407
8.10.3.2. Maximum Particle Integration Time Step 407
8.10.3.3. Chemistry Time Step Multiplier 407
8.10.4. Particle Termination Control 408
8.10.4.1. Maximum Tracking Time 408
8.10.4.2. Maximum Tracking Distance 408
8.10.4.3. Maximum Number of Integration Steps 408
8.10.4.4. Minimum Diameter 408
8.10.4.5. Minimum Total Mass 408
8.10.4.6. Mass Fraction Limits 408
8.10.5. Particle Ignition 409

8.10.6. Particle Source Smoothing 409
8.10.7. Vertex Variable Smoothing 409
8.10.8. Particle Source Control 409
8.10.8.1. Particle Heat Source Bounding 410
8.10.8.2. Particle Momentum Source Bounding 411
8.10.8.3. Linearization of Particle Mass Sources 412
8.10.8.3.1. Simple Mass Transfer Model 413
8.10.8.3.2. Liquid Evaporation Model 413
8.10.8.3.2.1. Droplet Temperature Below Boiling Point 413
8.10.8.3.2.2. Droplet Temperature Above Boiling Point 414
8.10.8.4. Particle Source Control Usage Notes 414
8.11. Multiphase Reactions and Combustion 414
8.11.1. Specification of a Binary Mixture 415
8.11.2. Reactants/Products 415
8.11.2.1. Example 415
8.11.3. Multiphase Reactions 416
8.11.3.1. Mass Arrhenius 416
8.11.3.2. Field Char Oxidation Model 416
8.11.3.3. Gibb Char Oxidation Model 416
8.11.3.4. Particle User Routine 416
8.11.3.5. Heat Release/Heat Release Distribution 417
8.11.4. Hydrocarbon Fuel Model Setup 417
8.11.4.1. Setup using Library Template (Recommended) 418
8.11.4.2. Set Up Manually (Experts) 420
8.11.4.3. Using Generic Multiphase Reactions Setup 421
8.12. Restrictions for Particle Transport 421
8.13. Restrictions for Particle Materials 422
8.14. Convergence Control for Particle Transport 422
8.15. Expert Parameters for Particle Tracking 423
8.16. Particle Diagnostics 423
8.17. Integrated Particle Sources for the Coupled Continuous Phase 423
8.18. Transient Simulations: What is Different for Particles? 423
9. Combustion Modeling 425
9.1. Reaction Models 426
9.1.1. Naming Convention for Reaction Schemes 426
9.1.2. Reaction Rate Types 427
9.1.2.1. Arrhenius 427
9.1.2.2. Arrhenius with Temperature PDF 428
9.1.2.3. Expression 428
9.1.2.4. Equilibrium 428
9.2. Using Combustion Models 429
9.2.1. Which Model is the Most Appropriate? 429
9.3. Eddy Dissipation Model 430
9.3.1. Fluid Models 431
9.3.1.1. Chemical Time Scale 431
9.3.1.2. Extinction Temperature 431
9.3.1.3. Maximum Flame Temperature 431
9.3.1.4. Mixing Rate Limit 431
9.3.1.5. Eddy Dissipation Model Coefficient A/B 432
9.3.1.6. Component Details 432
9.3.2. Initialization 432

9.3.3. Solver Parameters 432
9.4. Finite Rate Chemistry Model 433
9.4.1. Fluid Models 433
9.4.1.1. Chemical Time Scale 433
9.4.1.2. Extinction Temperature 433
9.4.1.3. Component Details 433
9.4.2. Initialization 434
9.4.3. Solver Parameters 434
9.5. Combined EDM/Finite Rate Chemistry Model 434
9.5.1. Fluid Models 435
9.5.1.1. Chemical Time Scale 435
9.5.1.2. Extinction Temperature 435
9.5.1.3. Component Details 435
9.5.2. Initialization 435
9.5.3. Solver Parameters 436
9.6. Reaction-Step Specific Combustion Model Control 436
9.7. Laminar Flamelet with PDF Model 436
9.7.1. Fluid Models 437
9.7.1.1. Component Details 437
9.7.2. Boundary Settings 437
9.7.2.1. Fuel 437
9.7.2.2. Oxidizer 437
9.7.2.3. Mixture Fraction 437
9.7.2.4. Mixture Fraction Mean and Variance 437
9.7.3. Initialization 438
9.7.4. Solver Parameters 438
9.8. Burning Velocity Model (Premixed or Partially Premixed) 438
9.8.1. Fluid Models 438
9.8.1.1. Component Details 439
9.8.2. Turbulent Burning Velocity 439
9.8.2.1. Value 439
9.8.2.2. Zimont Correlation 439
9.8.2.3. Peters Correlation 439
9.8.2.4. Mueller Correlation 439
9.8.3. Spark Ignition Model 439
9.8.3.1. Spark Kernel 440
9.8.3.2. Ignition Time 440
9.8.3.3. Spark Energy 440
9.8.4. Boundary Settings 440
9.8.4.1. Fuel 440
9.8.4.2. Oxidizer 440
9.8.4.3. Mixture Fraction 440
9.8.4.4. Mixture Fraction Mean and Variance 440
9.8.4.5. Reaction Progress 441
9.8.5. Initialization 441
9.8.5.1. Reaction Progress 441
9.8.6. Solver Parameters 441
9.8.7. Other Parameters 441
9.9. Extended Coherent Flame Model (ECFM) 441
9.9.1. Fluid Models 442
9.9.1.1. Laminar Flame Thickness 442

9.9.1.2. Component Details 442
9.9.2. Spark Ignition Model 442
9.9.2.1. Spark Kernel 442
9.9.2.2. Ignition Time 442
9.9.2.3. Spark Energy 443
9.9.3. Boundary Settings 443
9.9.3.1. Mixture 443
9.9.3.2. Reaction Progress 443
9.9.3.3. Flame Surface Density 443
9.9.4. Initialization 444
9.9.4.1. Mixture 444
9.9.4.2. Reaction Progress 444
9.9.4.3. Flame Surface Density 444
9.9.5. Solver Parameters 444
9.9.6. Other Parameters 445
9.10. Residual Material Model 445
9.10.1. Fluid Models 445
9.10.1.1. Reinitialization 445
9.10.2. Laminar Burning Velocity 446
9.10.3. Boundary Settings 446
9.10.4. Initialization 446
9.10.5. Other Parameters 446
9.11. Flamelet Libraries 446
9.11.1. Loading Flamelet Libraries 447
9.11.2. Flamelet Library (FLL) File Format 447
9.11.2.1. Flamelet Library (FLL) File Contents 447
9.11.2.2. Flamelet Library (FLL) File Format 448
9.11.2.2.1. Detailed FLL File Format 448
9.11.2.2.1.1. Comment Section 448
9.11.2.2.1.2. Header 448
9.11.2.2.1.3. Unburnt Flamelet 449
9.11.2.2.1.4. Burnt Flamelet Solution 449
9.11.2.2.2. Example FLL File 449
9.11.3. Stoichiometric Mixture Fraction 451
9.11.3.1. Value 452
9.11.3.2. Reactants 452
9.11.3.3. Automatic 452
9.11.4. Laminar Burning Velocity 452
9.11.4.1. Value 452
9.11.4.2. Metghalchi and Keck 452
9.11.4.3. Equivalence Ratio Correlation 453
9.11.4.3.1. Reference Burning Velocity 453
9.11.4.3.2. Flammability Limits 453
9.11.4.3.3. Preheat Temperature Dependency 453
9.11.4.3.4. Pressure Dependency 453
9.11.4.3.5. Residual Material Dependency 453
9.12. Autoignition Model 453
9.12.1. Ignition Delay Time 454
9.12.2. Customize Knock Reaction Rate 454
9.13. NO Model 455
9.13.1. Introduction to the NO Model 455

9.13.1.1. NO Model with Eddy Dissipation / Finite Rate Chemistry / Combined Model 455
9.13.1.2. NO Model with Flamelet Model 456
9.13.1.3. NO Model for Coal Combustion / Hydrocarbon Fuel Model 456
9.13.2. Fluid Models 456
9.13.2.1. Component Details 456
9.13.2.2. Boundary Conditions 457
9.13.3. Initialization 457
9.13.4. Solver Parameters 457
9.14. Chemistry Postprocessing 457
9.14.1. Fluid Models 458
9.14.1.1. Chemistry Postprocessing 458
9.14.1.2. Materials List 458
9.14.1.3. Reactions List 458
9.14.2. Boundary and Initial Conditions 458
9.14.3. Solver Parameters 458
9.15. Soot Model 458
9.15.1. Fluid Models 459
9.15.1.1. Soot Model 459
9.15.1.2. Fuel Material 459
9.15.1.3. Soot Material 459
9.15.1.4. Fuel Consumption Reaction 459
9.15.1.5. Fuel Carbon Mass Fraction 460
9.15.1.6. Soot Particle Mean Diameter 460
9.15.2. Boundary Settings 460
9.15.2.1. Mass Concentration and Nuclei Concentration 460
9.15.2.2. Mass Fraction and Specific Nuclei Specific Concentration 460
9.15.3. Initialization 461
9.16. Phasic Combustion 461
9.17. General Advice for Modeling Combusting Flows in CFX 461
9.17.1. General Procedure for Running Simulations 461
9.17.2. Advantages and Disadvantages of Multistep Reaction Mechanisms 462
9.17.3. Tips for Improving Convergence 462
9.17.4. Advanced Combustion Controls 462
10. Radiation Modeling 465
10.1. Comparison of the Radiation Models 466
10.2. Terminology 467
10.2.1. Absorptivity 467
10.2.2. Diffuse 467
10.2.3. Gray 467
10.2.4. Opaque 467
10.2.5. Reflectivity 468
10.2.6. Spectral 468
10.2.7. Transmissivity 468
10.3. Material Properties for Radiation 468
10.4. Rosseland Model 468
10.4.1. Fluid Models 468
10.4.1.1. Include Boundary Temperature Slip 469
10.4.1.2. Spectral Model 469
10.4.1.3. Scattering Model 469
10.4.2. Initial Conditions 469
10.4.3. Solver Control 469

10.5. The P1 Model 469
10.5.1. Fluid Models 469
10.5.1.1. Spectral Model 469
10.5.1.2. Scattering Model 470
10.5.2. Initial Conditions 470
10.5.3. Solver Control 470
10.6. The Discrete Transfer Model 470
10.6.1. Fluid Models 471
10.6.1.1. Number of Rays 471
10.6.1.2. Transfer Mode 471
10.6.1.3. Spectral Model 472
10.6.1.4. Scattering Model 472
10.6.2. Initial Conditions 472
10.6.3. Solver Control 472
10.7. The Monte Carlo Model 472
10.7.1. Fluid Models 472
10.7.1.1. Number of Histories 472
10.7.1.2. Transfer Mode 473
10.7.1.3. Spectral Model 473
10.7.1.4. Scattering Model 473
10.7.2. Initial Conditions 473
10.7.3. Solver Control 473
10.8. General Radiation Considerations 473
10.8.1. Domain Considerations 474
10.8.2. Domain Interface Considerations 474
10.8.3. Boundary Details 474
10.8.3.1. External Blackbody Temperature 474
10.8.3.2. Local Temperature 475
10.8.3.3. Radiation Intensity 475
10.8.3.4. Radiative Heat Flux 475
10.8.3.5. Emissivity 475
10.8.3.6. Diffuse Fraction 475
10.8.4. Sources 475
10.8.4.1. Directional Radiation Source 475
10.8.4.2. Isotropic Radiation Source 475
10.8.4.3. Directional Radiation Flux 476
10.8.4.4. Combining Radiation Sources 476
10.8.4.5. Isotropic Radiation Flux 476
10.8.5. Thermal Radiation Control 476
10.8.5.1. Iteration Interval 476
10.8.5.2. Diagnostic Output Level 477
10.8.5.3. Ray Reflection Control 477
10.8.5.4. Coarsening Control 477
10.8.5.4.1. Target Coarsening Rate 477
10.8.5.4.2. Minimum Blocking Factor 477
10.8.5.4.3. Maximum Blocking Factor 477
10.8.5.4.4. Small Coarse Grid Size 478
10.8.5.4.5. Diagnostic Output Level 478
10.8.5.5. Ray Tracing Control 478
10.8.5.5.1. Iteration Interval 478
10.8.5.5.2. Maximum Buffer Size 479

10.8.5.5.3. Maximum Number of Track Segments 479
10.8.5.5.4. Maximum Number of Iterations 479
10.8.5.5.5. Iteration Convergence Criterion 479
10.8.5.5.6. Ray Reflection Threshold 479
10.8.5.5.7. File Path 479
10.8.6. Spectral Model 479
10.8.6.1. Gray 479
10.8.6.2. Multiband 479
10.8.6.3. Multigray/Weighted Sum of Gray Gases 480
10.8.6.4. When is a Non-Gray Spectral Model Appropriate? 480
10.8.7. Scattering Model 481
10.8.7.1. Option = None 481
10.8.7.2. Option = Isotropic 481
10.8.7.3. Option = Linear Anisotropy 481
10.8.8. Radiometers 482
11. Rigid Body Modeling 485
11.1. Introduction to Rigid Body Modeling 485
11.2. Rigid Body Motion 485
11.3. Modeling a Rigid Body 487
11.3.1. Modeling a Rigid Body as a Collection of 2D Regions 487
11.3.2. Modeling a Rigid Body using an Immersed Solid 488
11.4. CEL Access of the Rigid Body State Variables 489
11.5. Monitor Plots related to Rigid Bodies 489
11.6. Solver Control of Rigid Bodies 490
11.7. Limitations to using Rigid Bodies 490
12. Real Fluid Properties 491
12.1. Setting up a Dry Real Gas Simulation 492
12.1.1. Using a Real Gas Equation of State 492
12.1.1.1. Redlich Kwong Dry Steam 492
12.1.1.2. Redlich Kwong Dry Refrigerants 493
12.1.1.3. Redlich Kwong Dry Hydrocarbons 493
12.1.1.4. Dry Redlich Kwong 493
12.1.2. Metastable States and Saturation Curve Clipping 493
12.1.3. Additional Comments 496
12.1.4. Using the IAPWS Equation of State 496
12.1.5. Using Real Gas Property (RGP) Tables 497
12.1.5.1. Loading `.rgp` files 497
12.1.5.2. Comments on Pressure and Temperature Ranges 497
12.1.5.3. Saturation Curve Clipping 497
12.1.6. Using a General Set-up (CEL or User Fortran) 497
12.1.6.1. Saturation Curve Clipping 498
12.2. Table Interpolation and Saturation Curve Clipping 498
12.2.1. Table Interpolation 499
12.2.2. Table Inversion 500
12.3. Equilibrium Phase Change Model 501
12.4. Setting up an Equilibrium Phase Change Simulation 502
12.4.1. Using a Real Gas Equation of State 503
12.4.1.1. Redlich Kwong Wet Steam 503
12.4.1.2. Redlich Kwong Wet Refrigerants 503
12.4.1.3. Redlich Kwong Wet Hydrocarbons 503
12.4.1.4. Wet Redlich Kwong 504

12.4.1.5. Creating a Liquid Phase Material 504
12.4.1.6. Creating the Homogeneous Binary Mixture 504
12.4.2. Using Real Gas Property (.rgp) table files 504
12.4.2.1. Loading an .rgp file 504
12.4.2.2. Creating the Homogeneous Binary Mixture 504
12.4.3. Using a General Set-up 504
12.4.3.1. Constants or Expressions 505
12.4.3.2. Antoine Equation 505
12.4.4. CFX-Pre Domain Models Set-up 505
12.5. Important Considerations 506
12.5.1. Properties 506
12.5.2. Inlet Boundary Conditions 506
12.6. Real Gas Property (RGP) File Contents 507
12.6.1. Superheat Region 507
12.6.2. Saturated Region 509
12.7. Real Gas Property (RGP) File Format 510
12.7.1. Organization of an .rgp File 510
12.7.1.1. Detailed .rgp File Format 511
12.7.1.2. Example .rgp File 514
12.8. Parameters in the .rgp File Controlling the Real Gas Model 519
12.8.1. REAL_GAS_MODEL 519
12.8.2. P_CRITICAL, T_CRITICAL 519
12.8.3. UNITS 520
12.8.4. MIN_PROPERTY_T, MAX_PROPERTY_T, MIN_PROPERTY_P and MAX_PROPERTY_P 520
12.8.5. MOLECULAR_VISCOSITY, MOLECULAR_CONDUCTIVITY 520
12.8.6. GAS_CONSTANT 520
12.8.7. TMIN_SATURATION, TMAX_SATURATION 520
12.8.8. P_TRIPLE, T_TRIPLE 520
12.8.9. SUPERCOOLING 521
13. Operating Maps and Operating Point Cases 523
13.1. Defining an Operating Point Case 523
13.2. Limitations 525
13.3. Running an Operating Point Case 525
13.4. Post-processing an Operating Point Case 526
14. Coupling CFX to an External Solver 529
14.1. Coupling CFX to an External Solver: System Coupling Simulations 529
14.1.1. Supported Capabilities and Limitations 530
14.1.2. Variables Available for System Coupling 532
14.1.2.1. Wall Force Data transferred from CFX System to System Coupling System 532
14.1.2.2. Displacement transferred from System Coupling System to CFX System 533
14.1.2.3. Thermal Data exchange between CFX and System Coupling 533
14.1.3. System Coupling Related Settings in CFX 533
14.1.4. Restarting CFX Analyses as Part of System Coupling 534
14.1.4.1. Generating CFX Restart Files 535
14.1.4.2. Specify a Restart Point in CFX 535
14.1.4.3. Making Changes in CFX Before Restarting 536
14.1.4.4. Recovering the CFX Restart Point after a Workbench Crash 536
14.1.4.5. Restarting from the Beginning of the Coupled Analysis in Workbench 536
14.1.5. Initializing a Coupled CFX Analysis from an Independent CFX Analysis 536
14.1.6. Running CFX as a Participant from System Coupling's GUI or CLI 537
14.1.7. Product Licensing Considerations when using System Coupling 538

14.2. Coupling CFX to an External Solver: Functional Mock-up Interface (FMI) Co-simulation 538
14.2.1. Limitations of using CFX with FMI 538
14.2.2. Units of FMU Output Variables 539
14.3. Coupling CFX to an External Solver: GT-SUITE Coupling Simulations 540
14.3.1. Supported Capabilities and Limitations 540
14.3.2. GT-SUITE Coupling Related Settings in CFX 540
14.3.3. Setting up a CFX Simulation to use GT-SUITE 541
14.3.3.1. Initializing or Re-initializing a GT-SUITE Model 542
14.3.3.2. Editing a GT-SUITE Model 544
14.3.4. Restarting CFX Analyses as Part of GT-SUITE Coupling 544
15. Aerodynamic Noise Analysis 545
15.1. Overview of Aerodynamic Noise Analysis 545
15.1.1. Near Field Noise Prediction 546
15.1.2. Far Field Noise Prediction 546
15.2. Types of Noise Sources 547
15.2.1. Monopole Sources 548
15.2.2. Dipole and Rotating Dipole Sources 548
15.2.3. Quadrupole Sources 548
15.3. Noise Source Strength Estimation in CFX 548
15.3.1. Monopole Sources 549
15.3.1.1. Monopole Data in the CFX-Solver Manager 549
15.3.1.2. Exporting Surface Monopole Data 549
15.3.2. Dipole or Rotating Dipole Sources 549
15.3.2.1. Dipole Data in the CFX-Solver Manager 549
15.3.2.2. Exporting Surface Dipole Data 549
15.3.3. Important Boundary Results Export Notes 550
15.3.4. Quadrupole Sources 550
15.3.4.1. Exporting Acoustic Quadrupole Data 550
15.4. The CGNS Export Data Format 550
15.4.1. File Naming Conventions 551
15.4.1.1. Important Notes 551
15.4.2. CGNS File Structure 551
15.4.2.1. Common Nodes for Mesh and Solution Files 553
15.4.2.2. Mesh File Specific Nodes 554
15.4.2.3. Solution File Specific Nodes 555
16. Advice on Flow Modeling 557
16.1. Solving Problems with Ansys CFX 557
16.2. Modeling 2D Problems 557
16.3. Mesh Issues 558
16.3.1. Physical Modeling Errors: YPLUS and Mesh Resolution Near the Wall 558
16.3.2. Measures of Mesh Quality 558
16.3.2.1. Mesh Orthogonality 559
16.3.2.2. Mesh Expansion 560
16.3.2.3. Mesh Aspect Ratio 561
16.4. Timestep Selection 562
16.4.1. Steady-state Time Scale Control 562
16.4.1.1. Max. Iterations 563
16.4.1.2. Minimum Number of Iterations 563
16.4.1.3. Time Scale Control 563
16.4.1.3.1. Auto Timescale 563
16.4.1.3.1.1. Length Scale Option 564

16.4.1.3.1.2. Timescale Factor 564
16.4.1.3.1.3. Maximum Timescale 564
16.4.1.3.1.4. Controlling the Time Scale with the Command File Editor 564
16.4.1.3.2. Local Time Scale Factor 565
16.4.1.3.3. Physical Time Scale 565
16.4.1.4. Solid Time Scale Control 566
16.4.1.4.1. Solid Timescale Factor 566
16.4.1.5. Controlling the Timescale for Each Equation 566
16.4.2. Transient Timestep Control 568
16.4.2.1. Time Duration 568
16.4.2.2. Timesteps: Timesteps List 569
16.4.2.3. Timesteps: Timesteps for the Run List 570
16.4.2.4. Timesteps: Adaptive 570
16.4.2.5. Initial Time 571
16.4.2.6. Max. Iter. Per Step 571
16.4.2.7. Transient Scheme 571
16.4.2.7.1. First Order Backward Euler 572
16.4.2.7.2. Second Order Backward Euler 572
16.4.2.7.2.1. Timestep Initialization 572
16.4.2.7.3. High Resolution 573
16.4.2.7.4. Harmonic Balance 573
16.4.2.7.5. Bounded Harmonic Balance 573
16.4.2.7.6. None 573
16.5. Advection Scheme Selection 573
16.5.1. Upwind 573
16.5.2. High Resolution 573
16.5.3. Specified Blend Factor 574
16.5.4. Central Difference 574
16.5.5. Advection Scheme for Turbulence Equations 574
16.5.6. Advection Schemes for Multiphase Volume Fractions 574
16.5.7. Comparisons to CFX-TASCflow 575
16.6. Advanced Options: Dynamic Model Control 575
16.6.1. Global Dynamic Model Control 575
16.6.2. Turbulence Control 576
16.6.3. Combustion Control 576
16.6.4. Hydro Control 576
16.7. Pressure Level Information 576
16.8. Interpolation Scheme 577
16.8.1. Pressure Interpolation Type 577
16.8.2. Velocity Interpolation Type 577
16.8.3. Shape Function Option 577
16.9. Temperature Damping 577
16.9.1. Option 577
16.9.2. Temperature-Damping Limit 578
16.9.3. Under-Relaxation Factor 578
16.10. Monitoring and Obtaining Convergence 578
16.10.1. Residuals 579
16.10.1.1. Residual Type and Target Levels 579
16.10.1.1.1. RMS Residual Level 579
16.10.1.1.2. MAX Residual Level 579
16.10.2. Using Interrupt Control in Cases with Transient Convergence Behavior 580

16.10.3. Global Balances and Integrated Quantities 582
16.10.3.1. Positive and Negative Domain Source Totals 582
16.10.3.2. Conservation Target 582
16.10.4. Convergence Rate 582
16.10.5. Problems with Convergence 583
16.10.5.1. Start-up Problems 583
16.10.5.2. Later Problems 584
16.11. Solver Issues 587
16.11.1. Robustness and Accuracy 587
16.11.2. Linear Solver Failure 587
16.12. How CEL Interacts with the CFX-Solver 588
16.13. Best Practice Guides 588
17. Using the Solver in Parallel 589
17.1. Partitioning 589
17.1.1. Node-based and Element-based Partitioning 590
17.1.2. Multilevel Graph Partitioning Software - MeTiS 591
17.1.3. Recursive Coordinate Bisection 591
17.1.4. Optimized Recursive Coordinate Bisection 592
17.1.5. Simple Assignment 592
17.1.6. User Specified Direction 592
17.1.7. Directional Recursive Coordinate Bisection 593
17.1.8. Radial 593
17.1.9. Circumferential 594
17.2. Setup for Parallel Runs 594
17.3. Message Passing Interface (MPI) for Parallel 594
17.4. Advice on Using Ansys CFX in Parallel 595
17.4.1. Optimizing Mesh Partitioning 595
17.4.2. Optimizing the Parallel Run 596
17.4.3. Error Handling 597
17.4.3.1. Problems with Intel MPI 597
17.4.3.1.1. Semaphores and Shared-memory Segments 597
17.4.3.1.2. Typical Problems When You Run Out of Semaphores 597
17.4.3.1.3. Checking How Many Semaphores Are in Use, and by Whom 598
17.4.3.1.4. Deleting the Semaphores You Are Using 598
17.4.3.1.5. Shared-memory Segment Size Problems 598
17.4.3.1.6. Checking Semaphore ID and Shared Memory Segment Limits 599
17.4.3.1.6.1. Linux 599
17.4.3.1.7. Increasing the Maximum Number of Semaphores for Your System 599
17.4.3.1.8. Max Locked Memory 599
17.4.3.2. Problems with the Ansys CFX Executables 600
17.4.3.3. Problems with Ansys CFX Licenses 600
17.4.3.4. Windows Problems 600
17.4.3.5. Linux Problems 600
17.4.3.6. Convergence Problems 601
17.4.4. Measuring Parallel Performance 601
17.4.4.1. Wall Clock Performance 601
17.4.4.2. Memory Efficiency 602
17.4.4.3. Visualizing Mesh Partitions 602
18. Expert Control Parameters 603
18.1. When to Use Expert Control Parameters 603
18.2. Modifying Expert Control Parameters 603

18.3. CFX-Solver Expert Control Parameters 603
18.3.1. Discretization Parameters 604
18.3.2. Linear Solver Parameters 611
18.3.3. I/O Control Parameters 612
18.3.4. Convergence Control Parameters 615
18.3.5. Physical Model Parameters 618
18.3.6. Particle Tracking Parameters 622
18.3.7. Transient Blade Row Model Parameters 624
18.3.8. Run Control Parameters 626
18.3.9. Model Override Parameters 627
19. User Fortran 631
19.1. User CEL Functions and Routines 632
19.1.1. Structure of User CEL Functions 633
19.1.2. User CEL Function Units 635
19.2. User Junction Box Routines 635
19.2.1. Junction Box Routine Options and Parameters 635
19.2.2. Calling Junction Box Routines 639
19.2.3. Reading Data with the User Input Option 639
19.2.4. Writing Data with the User Output Option 639
19.2.5. Other Junction Box Location Options 639
19.2.6. Structure of the User Junction Box Routines 639
19.2.7. Which Call 640
19.3. Shared Libraries 640
19.3.1. Creating the Shared Libraries 640
19.3.2. Default Fortran Compilers and Compiler Options 641
19.4. User Parameters 642
19.4.1. Adding and Modifying User Parameters 642
19.4.2. Parameter Names 642
19.4.3. Parameter Values 643
19.4.4. Example of CCL File User Parameters 643
19.4.5. Looking up a String Value 643
19.4.6. String Value Example 644
19.4.7. Looking Up List Values 644
19.4.8. Real Value Example 645
19.4.9. Looking up Sizes of Lists and Strings 645
19.4.10. Real List Example 646
19.4.11. Printing Parameters 646
19.5. Utility Routines for User Functions 647
19.5.1. Introduction to Utility Routines for User Functions 647
19.5.2. Data Acquisition Routines 648
19.5.2.1. USER_GETVAR 648
19.5.2.1.1. Boundcon Operator 650
19.5.2.1.2. Variable Shape and Dimensions 651
19.5.2.2. USER_GET_GVAR 651
19.5.2.2.1. USER_GET_GVAR with Multiphase Flow 656
19.5.2.3. USER_CALC_INFO 657
19.5.2.4. USER_GET_MESH_INFO 659
19.5.2.4.1. Global Mesh Information: CZONE = ' ', LOCALE = ' ' 659
19.5.2.4.2. Zonal Information: CZONE = 'ZNm', LOCALE = ' ' 660
19.5.2.4.3. Boundary Condition Patch Information: CZONE = 'ZNm', LOCALE = 'BCPn' 661
19.5.2.4.3.1. Face Set Information: CZONE = 'ZNm', LOCALE = 'FCSn' 661

19.5.2.4.3.2. Element Set or Element Group Information: CZONE = 'ZNm', LOCALE = 'ELSn' or 'IELGn' or 'BELGn' 661
19.5.2.5. USER_GET_MESHDATA 662
19.5.2.6. USER_GET_PHYS_INFO 664
19.5.2.6.1. Zonal Information: CZONE = 'ZNm', CPHASE = ' ' 665
19.5.2.6.2. Phase Information: CZONE = 'ZNm', CPHASE = 'FLm' or 'SLm' 666
19.5.2.7. USER_GET_TRANS_INFO 667
19.5.2.8. USER_ASSEMBLE_INFO 668
19.5.2.9. GET_PARALLEL_INFO 669
19.5.2.10. CAL_MESHMAP 669
19.5.3. Name Conversions 670
19.5.3.1. CONVERT_NAME_U2S 670
19.5.3.2. CONVERT_NAME_S2U 672
19.5.3.3. VAR_ALIAS 672
19.5.3.4. LOCALE_ALIAS 673
19.5.3.5. CONVERT_USER_NAMES 673
19.5.3.6. GET_PHASE_FROM_VAR 673
19.5.3.7. PARSMP 674
19.5.4. Character Handling 674
19.5.4.1. CFROMD 674
19.5.4.2. CFROMI 674
19.5.4.3. CFROMR 675
19.5.4.4. DFROMC 675
19.5.4.5. IFROMC 675
19.5.4.6. RFROMC 675
19.5.4.7. LENACT 675
19.5.4.8. EDIT 675
19.5.4.9. IOPNAM 676
19.5.4.10. CCATI 677
19.5.5. Output Routines 677
19.5.5.1. MESAGE 677
19.5.5.2. ERRMSG 678
19.6. CFX Memory Management System (MMS) 678
19.6.1. Introduction to the Memory Management System 679
19.6.2. Error Conventions 680
19.6.3. Stack Pointers 681
19.6.4. Subroutines 682
19.6.4.1. Directories 682
19.6.4.1.1. MAKDIR 682
19.6.4.1.2. DELDIR 682
19.6.4.1.3. CHGDIR 682
19.6.4.1.4. CHMDIR 683
19.6.4.1.5. PSHDIR 683
19.6.4.1.6. PSHDRH 683
19.6.4.1.7. POPDIR 683
19.6.4.1.8. LISDAT 684
19.6.4.1.9. PUTDIR 684
19.6.4.1.10. FNDFIL 684
19.6.4.2. Data Areas 687
19.6.4.2.1. MAKDAT 687
19.6.4.2.2. MAKVEC 688

19.6.4.2.3. GRBSTK 688
19.6.4.2.4. SQZDAT 688
19.6.4.2.5. RESIZE 688
19.6.4.2.6. DELDAT 689
19.6.4.2.7. LOCDAT 689
19.6.4.2.8. INFDAT 689
19.6.4.3. Renaming And Linking 689
19.6.4.3.1. RENAM 689
19.6.4.3.2. MAKLNK 690
19.6.4.3.3. DELLNK 690
19.6.4.4. Copying 690
19.6.4.4.1. COPDAT 690
19.6.4.4.2. COPDIR 690
19.6.4.5. Setting and Reading Individual Values 691
19.6.4.5.1. POKECA 691
19.6.4.5.2. POKECS 691
19.6.4.5.3. POKED 691
19.6.4.5.4. POKEI 691
19.6.4.5.5. POKEL 691
19.6.4.5.6. POKER 691
19.6.4.5.7. PEEKCA 692
19.6.4.5.8. PEEKCS 692
19.6.4.5.9. PEEKD 692
19.6.4.5.10. PEEKI 693
19.6.4.5.11. PEEKL 693
19.6.4.5.12. PEEKR 693
19.6.4.6. Name lists 693
19.6.4.6.1. NAMLST 693
19.6.4.7. Memory Management Statistics 695
19.6.4.7.1. GETMMS 695
19.6.4.7.2. WSTAT 697
19.7. User Fortran in Ansys Workbench 697
19.8. User CEL Examples 697
19.8.1. User CEL Example 1: User Defined Momentum Source 698
19.8.1.1. Problem Setup 698
19.8.1.2. Creating the User CEL Function 698
19.8.1.3. User Fortran Routine 699
19.8.2. User CEL Example 2: Using Gradients for an Additional Variable Source 700
19.8.2.1. Problem Setup 701
19.8.2.1.1. Creating the Additional Variables 701
19.8.2.1.2. Creating the Domain 701
19.8.2.1.3. Creating the User CEL Routine and Function 701
19.8.2.1.4. Defining the Source Term 702
19.8.2.2. User Fortran Routine 702
19.8.3. User CEL Example 3: Integrated Quantity Boundary Conditions 704
19.8.3.1. Problem Setup 704
19.8.3.1.1. Creating the User Function 704
19.8.3.2. User Fortran Routine 705
19.9. User Junction Box Examples 706
19.9.1. Junction Box Example 1: Profile Boundary Conditions 706
19.9.1.1. Problem Setup 706

19.9.1.1.1. Creating the Junction Box Routine and User CEL Function 707
19.9.1.1.2. Setting the Boundary Condition 707
19.9.1.1.3. Enabling the Junction Box Routine 708
19.9.1.2. User Fortran Junction Box Routines 708
19.9.2. Junction Box Example 2: Integrated Residence Time Distribution 714
19.9.2.1. Problem Setup 714
19.9.2.1.1. Creating the Additional Variables 714
19.9.2.1.2. Creating the Domains 715
19.9.2.1.3. Creating the Subdomains and Additional Variable Sources 715
19.9.2.1.4. Creating the Junction Box Routine 715
19.9.2.1.5. Enabling the Junction Box Routine 715
19.9.2.2. User Fortran Junction Box Routine 716
19.9.3. Junction Box Example 3: Timestep Control 718
19.9.4. Junction Box Example 4: Solver Control 723
19.9.5. Junction Box Example 5: Transient Information 725
19.10. Using CFX-4 Routines in CFX 727
19.11. State Point Evaluation 729
19.11.1. General Usage Notes 729
19.11.2. Supported Independent Thermodynamic Input Properties 730
19.11.3. Supported Thermodynamic Output Properties 730
19.11.4. Input and Output Property Name Format 731
19.11.4.1. Pure Substance Example 731
19.11.4.2. Mixture Example 732
19.11.5. Supported Material Types 733
19.11.6. Error Checks 733
19.11.7. USER_STATEPT Example 1 733
19.11.8. USER_STATEPT Example 2 737
19.12. Calculating the Particle Drag Coefficient 740
19.12.1. USER_PARTICLE_INFO Routine 741

List of Figures

1.1. Direction of Rotation 78
1.2. T-s Diagrams for Compression and Expansion Processes 102
2.1. Poor Location 117
2.2. Better Location 117
2.3. Ideal Location 118
2.4. Heat Transfer 145
2.5. Variables at a Wall Boundary 146
2.6. Symmetry Plane 149
2.7. The Local Orthogonal Coordinate System onto which Euler Equations are Recasted for the General NRBC Method 157
2.8. Waves Leaving and Entering a Boundary Face on Inflow and Outflow Boundaries. The Wave Amplitudes are Shown with the Associated Eigenvalues for a Subsonic Flow Condition 158
3.1. Initial Condition Values 167
3.2. Inaccurate Initial Conditions 167
4.1. Effect of increasing y+ for the flat plate T3A test case 208
4.2. Effect of decreasing y+ for the flat plate T3A test case 208
4.3. Effect of wall normal expansion ratio for the flat plate T3A test case 209
4.4. Effect of streamwise grid density for resolving the separation-induced transition due to a leading edge separation bubble for a wind turbine airfoil 210
4.5. Effect of streamwise grid density for the flat plate T3A test case 211
4.6. Decay of turbulence intensity (Tu) as a function of streamwise distance (x) 212
4.7. Blending Function for DES Model for Flow Around Cube. 221
4.8. Strain Rate Invariant for Flow Around Cube 222
4.9. Unsteady flow from DES over backward facing step using insufficient spanwise grid resolution 223
4.10. Resolved structures for cylinder in cross flow (top: URANS; bottom: SAS-SST) 225
5.1. Periodic Interfaces 235
5.2. General Connection: No Frame Change 236
5.3. General Connection: Frame Change and Pitch Change 236
5.4. Rotational Offset 238
5.5. Radial vs. Z Direction 241
5.6. Misaligned Components 241
5.7. Applying a Mass Flow at a Periodic Domain Interface 243
5.8. Applying a Mass Flow at a General Connection 244
5.9. Nodes Shared by Interfaces with Different Transformations 246
5.10. Element Aspect Ratio 251
5.11. Basic Impeller/Volute 252
5.12. Step change 252
5.13. Blade extending to the edge of the rotating domain. 253
5.14. Blade passage 254
5.15. Thick trailing edge 254
6.1. Example Fan Geometry 266
6.2. Typical Transient Rotor Stator Simulation using Profile Transformation 270
6.3. Typical Transient Rotor Stator Simulation using Time Transformation 270
6.4. Typical Transient Rotor Stator Simulation using Fourier Transformation 271
6.5. Inlet Disturbance Case 274
6.6. Application of Phase-Shifted Boundaries to a Blade Flutter Case 279
6.7. Direction of Forward Traveling Wave 280
6.8. 1.5 stage machine with equal number of blades in rows 1 & 3 but different than in row 2. Flow can be accurately modeled via a sequence of TT-corrected TRS interfaces. 288

6.9. 1.5 stages with differing blade numbers between rows 1, 2 & 3. Flow can be modeled via a sequence of TT-TRS interfaces. 288
6.10. Multistage modeled with a combination of TT and PT interfaces. For high speed flows, the TT-TRS interfaces should be placed where shocks are crossing between the adjacent passages. This is typically happening at the interface between the exit of the stator row and the inlet of the rotor row. 289
6.11. Multistage modeled with a combination of TT, PT and Stage interfaces 290
6.12. Multistage modeled with combination of TT-TRS and STT-TRS interfaces 291
6.13. Multistage modeled with several STT-TRS interfaces in sequence 291
8.1. Particle Size Distributions for Various Rosin Rammler n Parameters 359
8.2. Annulus projected onto a boundary region 379
8.3. Particle Track Behavior at a Wall Boundary 383
8.4. Point cone with specified cone angle 396
8.5. Point cone with specified cone angle and dispersion angle 396
8.6. Hollow cone with specified cone angle 396
8.7. Hollow cone with specified cone angle and dispersion angle 397
8.8. Spray Penetration 399
8.9. Spray Angle Calculation for Different Spray Shapes 399
8.10. Spray Angle Calculation with Finite Injection Radius with and without Spray Radius Specified 400
10.1. Example of a Reheat Furnace 481
11.1. Rigid Body with Initial Orientation 486
11.2. Rigid Body Reoriented with Euler Angles Shown 487
12.1. Pressure Volume Diagram for Water (Peng Robinson Equation of State) 494
12.2. Pressure Volume Diagram for Water (Aungier Redlich Kwong Equation of State) 494
12.3. Vapour Pressure Diagram for Water (Peng Robinson Equation of State) 495
12.4. Vapour Pressure Diagram for Water (Aungier Redlich Kwong Equation of State) 495
12.5. Typical Pressure-Temperature Diagram 498
12.6. Phase Diagram 501
12.7. Representation of Superheat Tables Associated with a Material 508
12.8. Representation of 1D Tables for a Saturation Property in Terms of Pressure 508
12.9. Representation of 1D Tables for Saturation Temperature in Terms of Pressure 508
12.10. Representation of a Saturation Table Associated with a Given Material 510
12.11. Schematic of the .rgp File Contents 511
17.1. The Partitioning Process 590
17.2. Node-based and Element-based Partitioning 590
17.3. Recursive Coordinate Bisection 592
17.4. User Specified Direction 593
17.5. Radial Partitioning 594
17.6. Circumferential Partitioning 594

List of Tables

1.1. Mesh Motion Options 45
1.2. Non-Newtonian Models 73
2.1. Supersonic Inlet Settings 126
3.1. Data-mapping Criteria 185
4.1. Fraction of laminar flow for a variety of different devices 207
7.1. Eulerian-Eulerian Multiphase vs. Particle Transport 299
8.1. Advantages and Disadvantages of Particle Transport 355
8.2. Coefficients for Some Materials Using the Tabakoff Erosion Model 363
14.1. Variables On Boundary Wall Regions 532
16.1. Convergence Problems Due to Local Effects 584
16.2. Convergence Problems Due to Global Effects 586

Chapter 1: Basic Capabilities Modeling

This chapter describes the basic physical models and some other basic capabilities found in Ansys CFX:

1.1. Domains

1.2. Physical Models

1.3. Sources

1.4. Material Properties

1.5. Mixture Properties (Fixed, Variable, Reacting)

1.6. Efficiency Calculation

1.7. Wall Condensation Model

Additional information on using boundary and initial conditions is available:

- Boundary Condition Modeling (p. 111)
- Initial Condition Modeling (p. 161).

Descriptions of more advanced physical models and how the basic models extend to more complex cases are provided in:

- Turbulence and Near-Wall Modeling (p. 197)
- Domain Interface Modeling (p. 233)
- Transient Blade Row Modeling (p. 257)
- Multiphase Flow Modeling (p. 295)
- Particle Transport Modeling (p. 353)
- Combustion Modeling (p. 425)
- Radiation Modeling (p. 465)
- Rigid Body Modeling (p. 485)
- Real Fluid Properties (p. 491)
- Coupling CFX to an External Solver: System Coupling Simulations (p. 529)
- Aerodynamic Noise Analysis (p. 545)
- Wall Condensation Model (p. 107)

Additional information is available:

- For general solver and convergence advice, see Advice on Flow Modeling (p. 557).
- For information on parallel processing and its implementation, see Using the Solver in Parallel (p. 589).
- For information on expert parameters, see Expert Control Parameters (p. 603).
- For information on extending the capabilities of CFX through Fortran subroutines, see User Fortran (p. 631).
- For a description of the mathematics used in the implementation of these models, see Basic Solver Capability Theory in the *CFX-Solver Theory Guide*.

1.1. Domains

Regions of fluid flow and/or heat transfer in CFX are called domains. Fluid domains define a region of fluid flow, while solid domains are regions occupied by conducting solids in which volumetric sources of energy can be specified. The domain requires three specifications:

- The region defining the flow or conducting solid. A domain is formed from one or more 3D primitives that constrain the region occupied by the fluid and/or conducting solids. For details, see Mesh Topology in CFX-Pre in the *CFX-Pre User's Guide*.
- The physical nature of the flow. This determines the modeling of specific features such as heat transfer or buoyancy.
- The properties of the materials in the region.

There can be many domains per model, with each domain defined by separate 3D primitives. Multidomain problems may be created from a single mesh if it contains multiple 3D primitives or is from multiple meshes. For details, see Importing and Transforming Meshes in the *CFX-Pre User's Guide*.

1.2. Physical Models

When you set up a simulation, you must select which physical models to include. The physical models define the type of simulation you want to perform.

Currently in CFX, the following models are available:

- Steady State and Transient Flows (p. 41)
- Mesh Deformation (p. 42)
- Laminar Flow (p. 54)
- Turbulence and Turbulence Models (p. 54)
- Heat Transfer (p. 55)
- Conjugate Heat Transfer (p. 56)
- Compressible Flow (p. 56)

- Setting a Reference Pressure (p. 57)
- Buoyancy (p. 58)
- Immersed Solids (p. 60)
- Multicomponent Flow (p. 64)
- Equilibrium Phase Change Model (p. 501)
- Additional Variables (p. 70)
- Non-Newtonian Flow (p. 73)
- Coordinate Frames (p. 76)
- Rotating Frames of Reference (RFR) (p. 77)
- Sources (p. 80)

In addition to these models, you can also introduce volumetric sources of mass, momentum energy, resistance, and Additional Variables using fluid subdomains. Point sources can also be applied to a single control volume. For details, see Source Points (p. 81).

Descriptions of more advanced physical models and how the basic models extend to more complex cases are provided in:

- Boundary Condition Modeling (p. 111)
- Initial Condition Modeling (p. 161)
- Turbulence and Near-Wall Modeling (p. 197)
- Domain Interface Modeling (p. 233)
- Multiphase Flow Modeling (p. 295)
- Particle Transport Modeling (p. 353)
- Combustion Modeling (p. 425)
- Radiation Modeling (p. 465)
- Real Fluid Properties (p. 491)
- Coupling CFX to an External Solver: System Coupling Simulations (p. 529)

1.2.1. Steady State and Transient Flows

The time dependence of the flow characteristics can be specified as either steady-state or transient. Steady-state simulations, by definition, are those whose characteristics do not change with time and whose steady conditions are assumed to have been reached after a relatively long time interval. They therefore require no real time information to describe them. Many practical flows can be assumed to be steady after initial unsteady flow development, for example, after the start up of a rotating machine.

Transient simulations require real time information to determine the time intervals at which the CFX-Solver calculates the flow field. Transient behavior can be caused by the initially changing boundary conditions of the flow, as in start up, or it can be inherently related to the flow characteristics, so that a steady-state condition is never reached, even when all other aspects of the flow conditions are unchanging. Many flows, particularly those driven by buoyancy, do not have a steady-state solution, and may exhibit cyclic behavior.

Sometimes simulations that are run in steady-state mode will have difficulty converging, and no matter what action you take regarding mesh quality and time step size, the solution does not converge. This could be an indication of transient behavior. If you have run a steady-state calculation and you see oscillatory behavior of the residual plots, you can test to see if you are observing a transient effect by reducing/increasing the time step size by known factors:

- If the period of oscillation of the residual plot changes by changing the time step size, then the phenomenon is most likely a numerical effect.
- If the period stays the same, then it is probably a transient effect.

In transient mode, you must set both physical time steps and the maximum number of coefficient iterations per time step.

1.2.2. Mesh Deformation

Mesh deformation is an important component for solving problems with moving boundaries or moving subdomains. The motion might be imposed, or might be an implicit part of a coupled fluid-structure simulation. Mesh deformation is available only for fluid domains. The following mesh deformation modeling options are available for a domain (in the **Domain** details view on the **Basic Settings** tab):

1.2.2.1. None
1.2.2.2. Regions of Motion Specified
1.2.2.3. Periodic Regions of Motion
1.2.2.4. Junction Box Routine

1.2.2.1. None

Select this option if no mesh deformation is desired.

1.2.2.2. Regions of Motion Specified

This mesh deformation option enables the specification of the motion of nodes on boundary or subdomain regions of the mesh using CEL. The motion of all remaining nodes is determined by the mesh motion model, which is currently limited to `Displacement Diffusion`. With this model, the displacements applied on domain boundaries or in subdomains are diffused to other mesh points by solving the equation:

$$\nabla \cdot \left(\Gamma_{\text{disp}} \nabla \delta \right) = 0 \tag{1.1}$$

In this equation, δ is the displacement relative to the previous mesh locations and Γ_{disp} is the mesh stiffness, which determines the degree to which regions of nodes move together. This equation is

solved at the start of each outer iteration or time step for steady-state or transient simulations, respectively.

It is worth noting that the displacement diffusion model for mesh motion is designed to preserve the relative mesh distribution of the initial mesh. For example, if the initial mesh is relatively fine in certain regions of the domain (for example, in boundary layers), then it will remain relatively fine after solving the displacement diffusion equation.

1.2.2.2.1. Displacement Relative To

The available **Displacement Relative To** options are:

- `Initial Mesh`

  This is the default option for transient blade row cases, and for cases where periodic mesh motion is set.

- `Previous Mesh`

  This is the default option for all other cases.

Note:

When using the `Initial Mesh` option, convergence problems may occur in cases where there are large mesh displacements, large changes in control volumes, or moving GGI interfaces. In such cases, you may need to write a custom CEL expression for mesh stiffness or solve relative to `Previous Mesh` instead if periodic mesh motion is not a requirement.

1.2.2.2.2. Mesh Stiffness

When any constant value of mesh stiffness is applied, specified displacements are homogeneously diffused throughout the mesh. When the mesh stiffness is specified as varying throughout the domain, nodes in regions of high stiffness move together (that is, there is little relative motion). Variable mesh stiffness is particularly useful to preserve the mesh distribution (and quality) near fine geometrical features (such as sharp corners) or in boundary layers.

The available **Mesh Stiffness** options are:

- `Increase Near Small Volumes`

  For details, see Increase Near Small Volumes (p. 44).

- `Increase Near Boundaries`

  For details, see Increase Near Boundaries (p. 44).

- `Blended Distance and Small Volumes`

  For details, see Blended Distance and Small Volumes (p. 45).

- `Value`

For details, see Value (Specified Stiffness) (p. 45).

1.2.2.2.2.1. Increase Near Small Volumes

The premise of increasing the stiffness near small mesh volumes is that mesh quality will benefit from having larger control volumes absorb more mesh motion. This option is desirable because it does not require the solution of additional equations (as for boundary dependent options). However, its behavior does depend on the initial mesh distribution (for example, having a fine mesh in regions where the motion is likely to be most significant).

The following relationship is used to determine the mesh stiffness, Γ_{disp}, applied in the displacement diffusion equation:

$$\Gamma_{\text{disp}} = \left(\frac{\forall_{ref}}{\forall}\right)^{C_{\text{stiff}}} \tag{1.2}$$

This relationship provides an exponential increase in the mesh stiffness as the control volume size, $\forall$, decreases. The stiffness **Model Exponent**, C_{stiff}, determines how quickly this increase occurs. For example, large values will yield a much more abrupt stiffness variation. This exponent is set to 2.0 by default. $\forall_{ref}$ is the reference volume, which can be automatically computed (**Reference Volume** > **Option** set to `Mean Control Volume`) or directly specified (**Reference Volume** > **Option** set to `Value`, with a default value of `1 [m^3]`). The reference volume should be representative of a typical control volume within the domain. Stiffness values are clipped between minimum and maximum values that are by default 1e-15 and 1e15, respectively.

When using `Increase Near Small Volumes` combined with **Displacement Relative To** set to `Initial Mesh`, then control volume size, $\forall$, is computed using the initial mesh in Equation 1.2 (p. 44).

1.2.2.2.2.2. Increase Near Boundaries

The premise of increasing the stiffness near certain boundaries (wall, inlet, outlet, and opening) is that mesh quality will benefit from having the mesh interior (that is, away from wall, inlet, outlet, and opening boundaries) absorb more mesh motion. Although this option requires the solution of an additional boundary scale equation, it is useful because it does not depend upon the original mesh distribution (that is, control volume size distribution) and it treats regions near all wall, interface, inlet, outlet, and opening boundaries identically. However, the mesh stiffness is unaffected near symmetry and one-to-one periodic interfaces.

The following relationship is used to determine the mesh stiffness, Γ_{disp}, applied in the displacement diffusion equation:

$$\Gamma_{\text{disp}} = \left(\frac{L_{ref}}{d}\right)^{C_{\text{stiff}}} \tag{1.3}$$

This relationship provides an exponential increase in the mesh stiffness as the distance from the nearest boundary, d, decreases. The stiffness **Model Exponent**, C_{stiff}, determines how quickly this increase occurs. For example, large values will yield a much more abrupt stiffness variation. This exponent is set to 2 by default. L_{ref} is the reference length, which can be automatically computed (**Reference Distance** > **Option** set to `Global Length Scale`) or directly specified (**Reference Distance** > **Option** set to `Value`, with a default value of `1 [m]`). The reference length should be representative of a typical length within the model. Stiffness values

are clipped between minimum and maximum values that are by default 1e-15 and 1e15, respectively.

Detailed information on the boundary scale equation is available at Wall and Boundary Distance Formulation in the *CFX-Solver Theory Guide*.

1.2.2.2.2.3. Blended Distance and Small Volumes

This option provides a more general model that adapts itself to cases, reducing the need for tuning. It combines features of `Increase Near Small Volumes` and `Increase Near Boundaries` and makes use of information about the local domain mesh.

The model has the following form:

$$\Gamma_{\text{disp}} = A\left(\frac{\forall_{\text{ref}}}{\forall}\right)^{C_{\text{vol}}} + B\left(\frac{L_{\text{ref}}}{\max\left(d, d_{\text{wall}}\right)}\right)^{C_{\text{dis}}} \tag{1.4}$$

Here, $\forall$ and d are the local values as before, but the reference values are instead found from the properties of the local domain mesh. For example:

$\forall_{\text{ref}}$ = mean control volume in domain

L_{ref} = 0.5 * (volume of domain)$^{1/3}$

d_{wall} = 10.0 * (minimum control volume in domain)$^{1/3}$

The factors A and B are simple default weights that indicate a dominance of one term over the other, or equality, and should sum to unity.

1.2.2.2.2.4. Value (Specified Stiffness)

The mesh stiffness can also be specified directly using the CEL. As suggested above, the mesh stiffness should increase in regions where mesh folding is most likely to occur. For example:

- The mesh stiffness could be made to increase with total mesh displacement
- The independent variable, d or $\forall$, could be clipped directly, or
- Multiple independent variables could be used.

1.2.2.2.3. Mesh Motion Options

The mesh motion options that are available on subdomain and boundary regions are nearly identical. The options listed in Table 1.1: Mesh Motion Options (p. 45) are accessible on the **Mesh Motion** and **Boundary Details** tabs for the subdomain and boundary regions, respectively.

Table 1.1: Mesh Motion Options

Option	Inlet, Outlet, Opening	Wall	Symmetry	Interface	Subdomain
Conservative Interface Flux (p. 47)				X	

Option	Inlet, Outlet, Opening	Wall	Symmetry	Interface	Subdomain
Unspecified (p. 47)	X	X	X	X	X
Stationary (p. 47)	X	X	X	X	X
Specified Displacement (p. 47)	X	X		X	X
Specified Location (p. 48)	X	X		X	X
Periodic Displacement (p. 48)	X	X			
Parallel to Boundary (p. 50)	X	X			
Surface of Revolution (p. 50)	X	X			
System Coupling (p. 51)		X			
Rigid Body Solution (p. 51)	X	X		X	X

Note:

When using `Unspecified` or `Conservative Interface Flux` conditions, there is no constraint that attempts to keep the meshes on either side of the interface together. One can expect that the mesh surfaces on either side of the interface will float apart or even cross over each other slightly. This will not generally affect solution accuracy because all interface values and transport flows are evaluated on control surfaces that represent an average of the surfaces on either side of the interface. However, if the mesh surfaces are too far apart (due to floating apart or due to crossover) then the solver may fail while trying to generate the average surface. Although very uncommon, this failure is fatal. GGI control expert parameters exist to relax the tolerances used while intersecting the meshes on either side of the interface. For details, see CFX-Solver Expert Control Parameters (p. 603).

The motion of all remaining nodes is implicitly unspecified, and is determined by solving a mesh displacement diffusion equation at the start of each time step for transient runs, or outer loop iteration for steady-state runs.

Note:

There is a three-level hierarchy for boundary conditions with mesh motion. That is, for mesh vertices shared between two or more boundary conditions, the applied condition is calculated from the attached boundary conditions based on the following priorities:

- `Stationary` takes precedence over a specified value: either `Specified Displacement` or `Specified Location`.
- A specified value takes precedence over a sliding condition (for example, `Parallel to Boundary` or `Surface of Revolution`).

- Any condition takes precedence over the `Unspecified` condition.

1.2.2.2.3.1. Conservative Interface Flux

This option is similar to Unspecified (p. 47) because no constraints on mesh motion are applied to nodes. The important difference is that motion of nodes in all domains adjacent to the interface influence, and are influenced by, the motion of the nodes on the interface.

Important:

Applying `Conservative Interface Flux` implies that the quantity in question will "flow" between the current boundary and the boundary on the other side of the interface. This means that `Conservative Interface Flux` must also be used on the boundary on the other side of the interface. Accordingly, the CFX-Solver will not be able to handle cases where `Conservative Interface Flux` is set on just one side of the interface, or where the quantity being transferred does not exist on the other side. CFX-Pre will issue a warning if either of these cases exist.

1.2.2.2.3.2. Unspecified

No constraints on mesh motion are applied to nodes. Their motion is determined by the motion set on other regions of the mesh. When this option is used on a domain interface boundary condition, only motion from other nodes in the current domain is "felt". It is worth noting that this motion will enable the mesh on one side of a domain interface to pull away from the mesh on the other side of the interface.

1.2.2.2.3.3. Stationary

Nodes are not permitted to move relative to the local boundary frame.

1.2.2.2.3.4. Specified Displacement

Nodes are displaced according to the specified CEL expression or User CEL. This displacement is always relative to the initial mesh and the local coordinate frame, if defined.

The following displacement options exist:

- `Cartesian Components`

  Provide X, Y, and Z components of the displacement.

- `Cylindrical Components`

  Provide axial, radial, and theta components of the displacement. The theta (circumferential) component is an arc length rather than an angle. The displacement in `Cylindrical Components` also requires the specification of **Axis Definition** either as a `Coordinate`

Axis of an existing coordinate frame (for details, see Coordinate Frames (p. 76)) or Two Points.

Note:

- If the **Use Mesh From** option is set to Initial Values, be careful with restarts because the definition of the initial mesh may have changed.
- When specifying mesh displacements using CEL, the variable Initial Cartesian Coordinates (with components Initial X, Initial Y, and Initial Z) may be useful. This variable represents the position of each node as it was at the start of the simulation (that is, the current position with Total Mesh Displacement subtracted).

1.2.2.2.3.5. Specified Location

All nodes are located according to the specified CEL expression or User CEL.

Note:

- Care should be taken when using this option because it is easy to get unexpected results. In particular, if a constant location (such as 0, 0, 0) is set for an entire region, all the mesh points will collapse to that point location.
- When specifying mesh locations using CEL, the variable Initial Cartesian Coordinates (with components Initial X, Initial Y, and Initial Z) may be useful. This variable represents the position of each node as it was at the start of the simulation (that is, the current position with Total Mesh Displacement subtracted).

1.2.2.2.3.6. Periodic Displacement

The **Periodic Displacement** option allows you to set a transient periodic mesh motion that repeats itself at a given frequency and has an associated phase offset.

The periodic mesh displacement input vector components can be specified as:

- **Cartesian Components**

  Provide X, Y, and Z components of the displacement as well as a **Scaling Factor** with which to multiply the displacement components in order to obtain the intended maximum displacement.

- **Normalized Cartesian Components**

  Provide X, Y, and Z components of the displacement and specify directly the maximum **Amplitude** of the displacement.

- **Complex Cartesian Components**

Provide X, Y, and Z components for the real and imaginary parts of the displacement as well as a **Scaling Factor** with which to multiply the displacement components in order to obtain the intended maximum displacement.

- **Complex Normalized Cartesian Components**

  Provide X, Y, and Z unit-normalized components for the real and imaginary parts of the displacement and specify directly the maximum **Amplitude** of the displacement.

For all of the above options, you need to provide the vibration **Frequency** as well as a **Phase Angle** that offsets the displacement in time. The following **Phase Angle** options are available:

- **Value**

  You can use this option to set a **Specified Phase Angle** to offset the mode shape displacement.

- **Nodal Diameter (Phase Angle Multiplier)**

  This option is only available for Transient Blade Row simulations.

  The corresponding settings are:

  – **Nodal Diameter Mag.**: the magnitude of the nodal diameter

  – **Traveling Wave Dir.**: the traveling wave direction: `Forward` or `Backward`.

    By default, the direction considered to be `Forward` is:

    → For a model that has a rotating component: the direction of machine rotation

    → For a model that has only stationary components (for example, a model of a stand-alone stator): the direction of increasing sector tag number

    A forward traveling wave is illustrated in Case 3: Blade Flutter (p. 277).

  – **Passage Number**

- **Interblade Phase Angle**

  This option is only available for Transient Blade Row simulations.

  The corresponding settings are:

  – **Interblade Phase Angle**

  – **Passage Number**

- **From Profile**

  When using a profile that has been expanded using CFX-Pre's profile expansion tool, the Nodal Diameter is read directly from the Harmonic Index entry in the original passage mode

shape profile. Similar to the `Nodal Diameter` option, you can select the **Traveling Wave Direction** (`Forward` or `Backward`). For details, see Case 3: Blade Flutter (p. 277).

Note:

- You can change which direction is considered to be forward (corresponding to a positive phase angle). For details, see the description for expert parameter `meshdisp phase angle convention` given in Physical Model Parameters (p. 618).
- If the **Use Mesh From** option is set to **Initial Values**, be careful with restarts because the definition of the initial mesh may have changed.

1.2.2.2.3.7. Parallel to Boundary

The `Parallel to Boundary` option allows the mesh to slide over an arbitrary boundary definition, with no component of deformation normal to it. This option attempts to preserve the geometry of the boundary as defined by its initial mesh.

Note:

Problems can occur in the following cases:

- Mesh nodes slide past a boundary perimeter, which is possible if the neighboring boundary has a mesh motion option other than `Stationary`.

  In this case, mesh nodes move along an extrapolation of the boundary in which they originated; they do not strictly follow the geometry of the next boundary.

  In addition, mesh motion can become non-periodic if a sliding mesh surface touches a Periodic Interface boundary.

- The mesh slides over a region of high curvature (or even a sharp edge).

  In this case, mesh nodes can move away from the geometry because the local surface normal is not well defined.

  To reduce this problem, ensure adequate mesh resolution near regions of high curvature.

1.2.2.2.3.8. Surface of Revolution

The `Surface of Revolution` option allows the mesh to slide over a boundary definition while maintaining the radial profile defined by the initial boundary mesh, and the axis of revolution.

For the `Surface of Revolution` option you can specify the **Axis Definition** either as a `Coordinate Axis` of an existing coordinate frame, or a local cylindrical axis defined by `Two Points`.

- The `Coordinate Axis` option allows you to select all available local and global coordinate axes.
- The `Two Points` option requires you to specify a pair of coordinate values as **Rotation Axis From** and **Rotation Axis To**.

Note:

- The `Surface of Revolution` option assumes that the radius is always constant in the circumferential direction, over the entire boundary mesh definition. As such, the mesh can accurately slide circumferentially beyond the extents of the initial boundary mesh definition. It can also slide axially beyond those initial extents, but in this case the radius used will be defined by the nearest point on the initial boundary mesh definition.
- The `Surface of Revolution` option requires that a genuine surface of revolution defines the underlying geometry, with no radial variation in the circumferential direction. Surfaces are permitted to run perpendicular to the axis of revolution, where multiple vertices exist at the same axial coordinate. If there are convergence difficulties with the mesh displacement equations, consider using the `Parallel to Boundary` option instead.

1.2.2.2.3.9. System Coupling

Nodes are displaced according to the displacement data received from the System Coupling system. For more information about simulations involving System Coupling and CFX, refer to Coupling CFX to an External Solver: System Coupling Simulations in the *CFX-Solver Modeling Guide* (p. 529).

1.2.2.2.3.10. Rigid Body Solution

Set **Rigid Body** to the name of the rigid body object that will govern the motion of the boundary/subdomain.

You can restrict which aspects of the rigid body motion are applied to the boundary/subdomain by selecting **Motion Constraints** and choosing one of the following options:

- `None`

  No constraints are applied to the boundary/subdomain. The motion of the rigid body object is applied to the boundary/subdomain.

- `Ignore Translations`

  Any translational motion of the rigid body object is ignored when applying the motion of the rigid body object to the boundary/subdomain.

- `Ignore Rotations`

  Any rotational motion of the rigid body object is ignored when applying the motion of the rigid body object to the boundary/subdomain.

For additional information on modeling rigid bodies, see Rigid Body Modeling in the *CFX-Solver Modeling Guide* (p. 485).

1.2.2.3. Periodic Regions of Motion

The `Periodic Regions of Motion` option can be an alternative to the `Regions of Motion Specified` option (described in Regions of Motion Specified (p. 42)) for transient cases that involve periodic motion of boundary walls (for example, transient blade row cases that involve blade flutter).

The **Periodic Model** setting (in CFX-Pre > **Domain** details view > **Domain Models** > **Mesh Deformation**) has the following option:

- `Complex Displacement`

  This model takes advantage of the nature of periodic motion, and solves the mesh motion equations at only the start of the simulation. Given the solution at the start of the simulation, the model is able to algebraically recompute the new mesh locations at different time steps during the simulation, reducing computational effort. Mesh motion robustness is also improved; once a full period of motion has been successfully completed, the mesh will not "fold" at any time during the remainder of the simulation.

  For implementation details of the `Complex Displacement` model, see Periodic Regions of Motion: Complex Displacement (p. 53).

The **Mesh Stiffness** options that are available for the `Periodic Regions of Motion` option are the same as those available for the `Regions of Motion Specified` option, described in Mesh Stiffness (p. 43).

The **Mesh Motion** options that are available for the `Periodic Regions of Motion` option are a subset of those available for the `Regions of Motion Specified` option, described in Mesh Motion Options (p. 45); specifically, all options are available except for the following:

- Specified Displacement
- Specified Location
- System Coupling
- Rigid Body Solution

Other limitations of the Periodic Regions of Motion mesh deformation model are listed next:

- The mesh stiffness must not change over time.
- There must be only a single frequency for the specified motion at the wall boundaries.
- In the current implementation, the `Periodic Regions of Motion` option cannot be used with the Harmonic Analysis solution method (described in Harmonic Analysis in the *CFX-Solver Theory Guide*).
- All mesh vertices are constrained to move along straight lines (not curves or arcs) during a period of motion; this is implicit in Equation 1.9 (p. 54). A consequence of this constraint is that sliding boundary conditions, particularly those on surfaces of revolution or other non-planar surfaces,

might not behave as expected. For example, if the `Surface of Revolution` mesh motion option is specified for a shroud, a mesh vertex on the shroud is unable to remain on the shroud if it tries to move in the circumferential direction (as might happen in a blade flutter case where the moving blade tip is near the shroud); instead of moving along an arc of constant radius, the vertex moves in a straight line from its initial position (generally along a tangent to the original surface at the vertex's original location). If the amplitude of the vertex displacement is small compared to the radius of the surface then the error might not be significant.

If this problem is severe, then you should switch to using the Region of Motion Specified mesh deformation model.

1.2.2.3.1. Periodic Regions of Motion: Complex Displacement

In the mesh motion method in CFX, the mesh deformation is found by solving a Laplacian diffusion equation:

$$\nabla \cdot \left(\Gamma \nabla \delta \right) = 0 \tag{1.5}$$

for a local stiffness Γ.

It has been noticed that the solution of the mesh motion equations can form a significant overhead in the total solution time of a simulation (as much as 20%). This is particularly unnecessary within a certain class of solution where boundary conditions are periodic, as the motion solution may also be periodic in the same way.

One simple solution is discussed here in which the whole mesh deformation throughout the domain is itself periodic.

1.2.2.3.1.1. Boundary conditions

If we limit the class of relevant solutions to cases similar to blade flutter calculations, for example, then the imposed displacements at a boundary are of the form:

$$\delta_{\text{boun}} = A \sin\left(\omega t + \varphi_{\text{boun}} \right) \tag{1.6}$$

1.2.2.3.1.2. Complex Displacement Solution

If we assume that all displacements satisfy Equation 1.5 (p. 53), and that all prescribed boundary conditions are either stationary or of the form in Equation 1.6 (p. 53), then it can be shown that the solution of the mesh displacement equations can be found in frequency space, rather than the time-domain. Posed as a complex number problem we find:

$$\tilde{\delta} = R + i\hat{R}, \quad \nabla \cdot \left(\Gamma \nabla \tilde{\delta} \right) = 0 \tag{1.7}$$

With boundary conditions:

$$\begin{aligned} \tilde{\delta} &= 0 && :\text{Stationary} \\ \tilde{\delta} &= A_i \left[\sin\left(\omega t + \sigma_i \right) - i \cos\left(\omega t + \sigma_i \right) \right] && :\text{Prescribed} \end{aligned}$$

where σ_i represents the phase angle at each boundary/blade, and A_i the prescribed amplitude of oscillation. By using a separation of variables, we can solve directly for the real and imaginary parts, subject to known boundary conditions:

$$\begin{aligned} \nabla\cdot\left(\Gamma\nabla R\right)&=0, \quad R=A_i\cos\left(\sigma_i\right) \\ \nabla\cdot\left(\Gamma\nabla \hat{R}\right)&=0, \quad \hat{R}=A_i\sin\left(\sigma_i\right) \end{aligned} \tag{1.8}$$

And the periodic solution at any point in time is reconstructed from:

$$\delta=R\cos\left(\omega t-\frac{\pi}{2}\right)-\hat{R}\sin\left(\omega t-\frac{\pi}{2}\right) \tag{1.9}$$

The big advantage with this approach is that the equations in Equation 1.8 (p. 54) can be solved to convergence right at the start of the simulation before any other equation systems (Note that in 3D simulations, R and A are essentially vectors but the 3 components of R can be solved independently with 3 Laplacian equations). Once the solution is found, Equation 1.9 (p. 54) is used on all time steps. Note that Equation 1.9 (p. 54) represents a linear system, which is why the mesh vertices only travel in straight lines.

1.2.2.4. Junction Box Routine

When this option is chosen, a User Fortran routine that explicitly sets the coordinates of all nodes in a given domain is specified and called. The Junction Box Routine that is called is specified with this mesh deformation option rather than with other Junction Box Routines on the **Solver Control** dialog box. The specified routine is always called at the start of each time step, and all mesh coordinate dependent quantities (for example, the volumes of control volumes) are automatically updated.

Note:

This option becomes available only once one or more Junction Box Routines have been created; only one Junction Box Routine can be selected.

Important:

Only the coordinates of mesh nodes may change; the topology (that is, the connectivity) of the mesh must remain fixed.

1.2.3. Laminar Flow

For details, see The Laminar Model (p. 198).

1.2.4. Turbulence and Turbulence Models

Information on turbulent flow, turbulence models and wall functions is available in Turbulence and Near-Wall Modeling (p. 197). A description of how turbulence modeling can be extended to multiphase flow is available in Turbulence Modeling in Multiphase Flow (p. 319).

The mathematical details of the turbulence models used in CFX is available in Turbulence and Wall Function Theory in the *CFX-Solver Theory Guide*.

1.2.5. Heat Transfer

A heat transfer model is used to predict the temperature throughout the flow. Heat transfer by conduction, convection, and (where appropriate) turbulent mixing and viscous work are included.

The following options for heat transfer available in CFX:

1.2.5.1. None

Select this option if your simulation does not involve the modeling of heat transfer. It eliminates the heat transfer calculation from the governing equations. This option reduces the number of calculations performed, and subsequently the time required, by the CFX-Solver.

1.2.5.2. Isothermal

This model requires you to enter a uniform temperature for the fluid in absolute temperature terms. This can be used for the purpose of evaluating fluid properties that are temperature-dependent; for example, the density of an ideal gas. For general fluids, a constant temperature can be used as the basis for a series of isothermal simulations using temperature-dependent fluid properties. You may also use this option to create an initial results file for a more complex model. Heat transfer is not modeled.

1.2.5.3. Thermal Energy

This models the transport of enthalpy through the fluid domain. It differs from the `Total Energy` model in that the effects of mean flow kinetic energy are not included. It consequently reproduces the same results as the `Total Energy` model when kinetic energy effects vanish, and is therefore adequate for low speed flows where kinetic effects are negligible.

For cases where viscous heating and the effects of turbulence on it are important, it is recommended that you use the `Total Energy` model.

1.2.5.4. Total Energy

This models the transport of enthalpy and includes kinetic energy effects. It should be used where kinetic energy effects become significant, for example gas flows where the Mach number exceeds 0.3. The selection of the `Total Energy` model has implications for whether the fluid is modeled as compressible or incompressible (see Compressible Flow (p. 56)) and for which buoyancy model (see Buoyancy (p. 58)) is employed by the CFX-Solver.

The mathematical model for heat transfer is available in Transport Equations in the *CFX-Solver Theory Guide*.

Note:

The **Incl. Viscous Work Term** option should be selected for a case involving MFR (multiple frames of reference) and the Total Energy equation. For details, see The Total Energy Equation in the *CFX-Solver Theory Guide*.

1.2.5.5. Turbulent Flux Closure

When the heat transfer model is set to either **Thermal Energy** or **Total Energy**, **Turbulent Flux Closure for Heat Transfer** may be optionally selected in the **Turbulence** settings. When **Turbulent Flux Closure for Heat Transfer** is not selected, the default is the **Eddy Diffusivity** model with the turbulent Prandtl number of 0.9. The turbulent Prandtl number may be non-constant using by expressions (CEL), for example, as a function of physical space or solution variables.

By default the turbulent Prandtl number specified for heat transfer will also be applied as the turbulent Schmidt number for component mass fractions and combustion scalars.

Also see Turbulent Flux Closure for Heat Transfer in the *CFX-Solver Theory Guide*.

1.2.6. Conjugate Heat Transfer

CFX enables you to specify regions that are occupied by conducting solids. You can specify such a region as a solid domain using the domains form.

To specify a source of energy in a solid domain, you should create a subdomain for the solid region (which may include the entire region) and apply the source term to the subdomain. Energy sources can be specified in terms of a source value and a linear source coefficient.

Information on the mathematical implementation of subdomain sources is available in Sources in the *CFX-Solver Theory Guide*.

The mathematical equations solved in a solid domain are available in Conjugate Heat Transfer in the *CFX-Solver Theory Guide*.

1.2.7. Compressible Flow

CFX can solve for subsonic (less than the speed of sound), transonic (close to the speed of sound), and supersonic (greater than the speed of sound) flows when compressible fluid models are used. Compressible flow is activated in CFX-Pre by using an ideal gas or real fluid or a general fluid whose density is a function of pressure.

When modeling high-speed compressible flow (transonic or supersonic), you may want to use supersonic inlet and/or supersonic outlet boundaries. In this case, you must use the total energy heat transfer model.

A large source of convergence problems arise due to non-physical problem specification, not only in supersonic flows. When you are setting up a problem make sure that the boundary conditions, material properties and geometry are all physically consistent. For instance, for certain internal geometries, there is a maximum achievable mass flow rate. The solver will not produce a converged solution with mass flow boundary conditions specified with a higher value than this limit. Another potentially problematic situation is when an oblique shock wave impinges on a boundary. To begin with, you should try using a slip wall to represent this boundary. Once convergence is achieved, you can switch the boundary to a supersonic Outlet. This may or may not converge, as a domain boundary/shock wave interaction is difficult to model at an Outlet. It may be necessary to move the side boundary in such cases.

For compressible flows, the pressure level should always be set. The pressure can be specified:

- By one of the boundaries in the simulation in the form of a pressure specified inlet, outlet or opening (in transient compressible flows, the initial conditions can set the pressure level).
- If the pressure level is not set at any boundary, it can be set at a particular location using the **Pressure Level Information** section on the **Advanced** tab of the **Solver Control** dialog box.

If the pressure level is not set, the CFX-Solver will set zero relative pressure at node 1. During the course of solution, this may yield negative relative pressures, and if the reference pressure is also small, this may lead to negative absolute pressure values, which is physically inconsistent. The CFX-Solver will then attempt to calculate the density with a negative value of absolute pressure, and will fail. For details, see Setting a Reference Pressure (p. 57).

1.2.7.1. Mixed Subsonic/Supersonic Boundaries

If you expect the conditions at an Inlet to be mixed subsonic/supersonic, you should use the Mixed flow regime option.

If you expect the conditions at an Outlet to be mixed subsonic/supersonic, there is no way to currently deal with this in CFX. Usually, the way to resolve such a situation is similar to the Outlet boundary problem where inflow occurs (that is, to move the boundary to a position where the conditions are known to be either fully subsonic or fully supersonic).

1.2.8. Setting a Reference Pressure

In CFX, you must specify a **Reference Pressure** for your simulation. The **Reference Pressure** is specified on the **Basic Settings** tab of the Domains form, but is a property of the entire simulation so all domains must use the same value. Each time you create a new domain or apply a change to an existing domain, the Reference Pressure in that domain is applied to all domains.

All relative pressure specifications set in CFX are measured relative to this **Reference Pressure** value. The **Reference Pressure** will affect the value of every other pressure set in the simulation.

The reference pressure is used to avoid problems with round-off errors. These can occur when the dynamic pressure change in a fluid, which is what drives the flow, are small compared to the absolute pressure level.

For example, low speed atmospheric air flow may have dynamic pressure changes of only a few Pascals or less, but the changes are relative to the atmospheric pressure of around 100,000 Pa. If you are dealing only in absolute pressure terms, these small pressure changes can get lost in round-off errors when performing calculations (for example, a change of 1 Pa is a change to the sixth significant digit). To rectify the situation, you should set a sensible **Reference Pressure** level. In this case, the local atmospheric pressure of 100,000 Pa is suitable. This value will be used as the new datum (instead of 0 Pa) about which all pressures are calculated. A change of 1 Pa will now be a change to the first significant digit.

As a counterexample, a reference pressure of 0 Pa can be used without any problems when the dynamic pressure changes are significant compared to the absolute pressure level. When modeling a liquid flow where nothing depends on the pressure level, there is no need to specify an atmospheric reference pressure.

When boundary and initial conditions are specified, they are set relative to the reference pressure. If you require a boundary to have an absolute pressure level of 100,000 Pa, you could:

- Set a relative pressure value of 0 Pa for the boundary if the reference pressure is 100,000 Pa or,
- Set a relative pressure value of 100,000 Pa for the boundary if the reference pressure is 0 Pa.

Note:

The **Reference Pressure** may be evaluated as a CEL expression. However, **Reference Pressure** values that vary over time are not supported.

If the system variable called `Pressure` (or `p`) is used in an expression defining material properties (for example, when setting a custom equation of state for density), this variable is interpolated as absolute pressure rather than relative pressure.

1.2.9. Buoyancy

Natural and mixed convection flows and flows in which gravity is important can be modeled by CFX by the inclusion of buoyancy source terms. Natural convection refers to the case where convection of the fluid is driven only by local density variations, for example, in a closed box with a heat source. Mixed convection refers to the case where convection of the fluid is driven by both a pressure gradient and buoyancy forces.

Buoyancy is driven by variations in density that can arise from a number of sources:

- Local temperature variations cause changes in density; this is natural convection.
- In multicomponent flows, variations in the mass fraction cause density variations because each component usually has a different density.
- In multiphase flows, including particle transport modeling, the difference in density between the phases results in a buoyancy force. For details, see Buoyancy in Multiphase Flow (p. 304).
- If density is variable for a **General Fluid** (that is, defined by an expression), a buoyancy force will arise.
- For ideal gases and real fluids, local pressure variations also cause changes in density. These changes are often small and the buoyancy effect is usually not important in the flow. Buoyancy does not necessarily need to be modeled if there are no other sources of buoyancy.

Therefore, for an isothermal, single phase, single component flow using a **General Fluid** with constant density, there are no buoyancy forces.

The relative importance of buoyancy forces due to temperature variations in a mixed convection flow can be estimated by using the ratio of Grashof and Reynolds Number,

$$\frac{Gr}{Re^2} = \frac{g \beta L \Delta T}{U^2} \tag{1.10}$$

where β is the thermal expansion coefficient. A value approaching or exceeding unity indicates that buoyancy effects are significant in the flow, while small values indicate that buoyancy effects can be ignored.

In purely natural convection problems, the Raleigh Number (Ra) indicates the relative strength of the buoyancy induced flow and is given by:

$$Ra = Gr\,Pr \tag{1.11}$$

where Pr is the fluid Prandtl number. The laminar flow regime is generally characterized by $Ra < 10^8$, while turbulent buoyant flow is characterized by $Ra > 10^{10}$.

For calculations involving buoyancy, the gravity vector components in *x*, *y* and *z* must be set. These are interpreted in the coordinate frame for the domain. For information on setting the gravity vector components for a rotating domain, see Buoyancy In Rotating Domains (p. 60).

Buoyancy effects can be simulated using one of two available models in CFX:

- Full Buoyancy Model (Density Difference) (p. 59)
- Boussinesq Model (p. 59)

1.2.9.1. Full Buoyancy Model (Density Difference)

For single phase flows, this model is used when the fluid density is a function of temperature or pressure (which includes all ideal gases and real fluids) and when a multicomponent fluid is used. For Eulerian multiphase or particle tracking, it is also set even if all phases have constant density. Significant density variations with temperature occur for most gases. You should specify a **Buoyancy Reference Density** as an approximate average value of the expected domain density. For multiphase simulations, other factors must be considered. For details, see Buoyancy in Multiphase Flow (p. 304).

An explanation of the mathematical treatment of the **Full Buoyancy** model is available in Full Buoyancy Model in the *CFX-Solver Theory Guide*.

1.2.9.2. Boussinesq Model

For many applications involving buoyancy, it is sufficient to assume a constant fluid density when the change in density over the expected range of conditions is relatively small. This is often true for many liquids. When the fluid density is not a function of pressure or temperature, the Boussinesq model is employed.

The Boussinesq model uses a constant density fluid model, but applies a local gravitational body force throughout the fluid that is a linear function of fluid thermal expansivity, β, and the local temperature difference with reference to a datum called the **Buoyancy Reference Temperature**. You should specify the reference temperature as an approximate average value of the expected domain temperature.

The Boussinesq model can be used for the following cases:

- Single-phase, single component simulations with heat transfer, using a constant density **General Fluid**. This is the default usage of the Boussinesq model.
- Constant density fluids in an Eulerian or particle tracking multiphase flow with heat transfer in conjunction with the full **Buoyancy Model**.

. For details, see Buoyancy in Multiphase Flow (p. 304). Note that the Boussinesq model is not available for multicomponent fluids and Eulerian-Eulerian multiphase flows. To include the effect of temperature on buoyancy in these cases, you must specify the components for which density is a function of temperature (for example, ideal gases).

An explanation of the mathematical treatment of the Boussinesq approximation is available in Boussinesq Model in the *CFX-Solver Theory Guide*.

1.2.9.3. Buoyancy and Pressure

When buoyancy is activated, the pressure calculated by the solver excludes the hydrostatic pressure gradient. This modified pressure is often called motion pressure because it is responsible for driving the flow. All initial conditions and boundary conditions are interpreted in terms of this modified pressure. For details, see Buoyancy in the *CFX-Solver Theory Guide*.

In some situations, the true pressure is required to calculate fluid properties. For example, if the fluid is compressible, the density depends on the true pressure rather than the motion pressure. In addition, it is often useful to visualize the true pressure. For these reasons, the solver automatically includes the hydrostatic contribution in the variable called `Absolute Pressure`. The relationship between absolute pressure and the pressure calculated by the solver is available in Buoyancy in the *CFX-Solver Theory Guide*. When fluid properties are functions of pressure, the absolute pressure is automatically used to evaluate them. Absolute pressure is also written to the results file.

The calculation of absolute pressure requires a buoyancy reference location to be defined. By default, the solver chooses this to be the centroid of one of the pressure-specified boundaries, or if there are no pressure-specified boundaries, the pressure reference location. The pressure reference location is specified under **Pressure Level Information** on the **Solver Control** dialog box in CFX-Pre. For details, see Advanced Options Tab. You can optionally specify the buoyancy reference location directly by providing Cartesian coordinates in the domain buoyancy settings.

1.2.9.4. Buoyancy In Rotating Domains

For steady-state rotating domain simulations involving buoyancy, the gravity vector must be aligned with the axis of rotation.

For transient rotating domain simulations involving buoyancy, if the gravity vector is not aligned with the vector of rotation, the solver automatically counter-rotates the relative frame gravity vector such that it has the specified value in the stationary domain.

1.2.10. Immersed Solids

The Immersed Solids capability of Ansys CFX enables you to model steady-state or transient simulations involving rigid solid objects that can move through fluid domains. The model involves the use of an immersed solid domain that is placed inside a fluid domain. As a simulation proceeds, CFX-Solver applies a source of momentum to the fluid inside the immersed solid domain in order to force the flow to move with the solid.

In a parallel run, all partitions hold a copy of the entire mesh of the immersed solid domain. This facilitates the parallelization of the intersection process and eliminates unnecessary communications.

Steps to create an immersed solid:

1. Define an immersed solid domain to represent the solid. For details, see Domain Type in the *CFX-Pre User's Guide*.

   This domain should be entirely or partly within a fluid domain. Care must be taken to ensure that the immersed solid domain does not cross any fluid boundaries or collide with any solid

domains or immersed solid domains. An immersed solid domain should not cross any GGI interface that involves a non-stationary domain. Do *not* create a domain interface between the immersed solid domain and the fluid domain.

2. Specify the **Domain Motion** settings for the immersed solid domain in order to prescribe the motion of the immersed solid. For details, see Domain Motion in the *CFX-Pre User's Guide*.

The immersed solid is represented as a source term in the fluid equations that drives the fluid velocity to match the solid velocity. The size of the source term is controlled by the **Momentum Source Scaling Factor** setting, which can be set globally (in the global **Solver Control** settings) or for individual immersed solids (on the immersed solid domain **Solver Control** tab). The default value of 10 is acceptable most of the time. If robustness problems are encountered, the scaling factor may be decreased (for example, by a factor of 2), but at the expense of accuracy; the difference between the fluid velocity and the specified solid velocity will generally increase, even if only by a small amount.

For problems that have convergence difficulties, you can try using better initial conditions. You can obtain initial conditions by using one of the following methods:

- Run a case with **Momentum Source Scaling Factor** set to zero or a very small value (for example, less than 1.0).
- Run a stationary case and use the results to initialize a moving immersed solid case.

Once you have good initial conditions, you can improve the stability of the solution by using one of the following methods:

- Use the `inside()` function to initialize the flow field so that the flow inside the immersed solid moves with the immersed solid.
- Use a tighter convergence criterion and enable more coefficient loops per time step.

When you postprocess a simulation that involves an immersed solid, the results file and any full transient results files contain a variable named "Inside [domain name]". This variable is set to 1 for mesh nodes inside the immersed solid, and 0 for mesh nodes outside the immersed solid.

Variables used for plots or calculations on immersed solid domain boundaries are not taken from the immersed solid domain; instead, they are interpolated from the fluid/porous domain in which the solid is immersed. The accuracy of such interpolation is dependent on the mesh densities of both the fluid/porous domain and the surface of the immersed solid domain. To visualize, or perform computations with, variables that are associated with the immersed solid domain, use slice planes, user surfaces, or other locators that are offset into the immersed solid domain, and set the applicable **Domains** setting to refer to the immersed solid domain.

Note:

CFD-Post does not add viscous stresses to the evaluation of variables or expressions on immersed solid surfaces. Thus, using CFX-Solver Manager to monitor "force()@Immersed solid" (where 'Immersed solid' is the name you have given to a surface) gives a different result to calculating "force()@Immersed solid" in CFD-Post.

1.2.10.1. Immersed Boundary Tracking

By default, the fluid just outside an immersed boundary has no forcing terms to account for the boundary layer or other effects of the boundary on the flow. To better resolve these boundary effects, CFX-Solver can impose a modified forcing term near the immersed boundary. To activate these forcing terms, visit the **Immersed Solid Control** settings (see Immersed Solid Control in the *CFX-Pre User's Guide*), then set **Boundary Model** to `Modified Forcing` and set the **Boundary Tracking** settings.

In order for CFX-Solver to calculate the forcing terms, the nearest point on the immersed boundary must be identified for each near-wall node (see Notation in the *CFX-Solver Theory Guide*). This involves using one of two search algorithms. Which search algorithm is used depends on which option is selected for the **Boundary Tracking** setting: `Search Through Elements` or `Boundary Face Extrusion`.

The search algorithm that is used when the **Boundary Tracking** setting is `Boundary Face Extrusion` has the advantage of finding the wall normal direction in physical space (as opposed to computational space), but the success of this algorithm depends on the distance that the boundary face is extruded along the wall normal direction.

When using the `Search Through Elements` method, it is important to have the immersed solid mesh element lengths larger in the direction normal to the immersed boundary than the fluid mesh element lengths near the immersed boundary, so that the in-wall nodes tend to fall within the outer layer of immersed solid boundary elements. When using this option, CFX-Solver will search through elements near the immersed boundary and project the near-immersed-boundary fluid nodes onto the face edge or vertices of the immersed solid element. The weakness of this option is that the projection is done in the computational space so that, compared to the other option, the normal direction is generally less accurate.

When using the `Boundary Face Extrusion` method, it is important to have the extrusion distance larger than the fluid mesh element lengths near the immersed boundary, to ensure that the near-wall nodes fall within the bounding boxes of the imaginary volumes extruded from the immersed boundary face. When using this option, CFX-Solver will search through fluid nodes near the immersed boundary and project the near-immersed-boundary fluid nodes onto the faces or edges of the immersed boundary, with the projection being carried out in physical space. Because the projections are done in physical space, as opposed to computational space, accuracy is better than that of the `Search Through Elements` option.

1.2.10.2. Limitations to using Immersed Solids

The following modeling restrictions and limitations apply to simulations involving immersed solids:

- An immersed solid domain cannot undergo mesh deformation.
- An immersed solid domain cannot model heat transfer.
- When modeling an immersed solid governed by the rigid body solver, you cannot set initial conditions.
- A fluid domain that contains (or partly contains) an immersed solid domain cannot support many modeling options, including:

  - heat transfer
  - Additional Variables
  - combustion

- For transient cases, immersed solids do not interact properly with fluid domains that involve compressibility or multiphase flow.
- For steady-state simulations, if the immersed solid is moving, the motion must not have any normal component.
- The solid surfaces of an immersed solid are not explicitly resolved by the mesh. In addition, a wall function cannot be applied to the boundary of an immersed solid. As a consequence, the quality of simulation results may be lower than can be obtained using mesh deformation or other techniques that support the use of wall boundaries to directly resolve solid surfaces.

  In general, the mesh for the immersed solid should be sufficiently fine on the boundaries to resolve the shape of the boundary surface; the mesh inside the immersed solid may be arbitrarily coarse. The fluid mesh around the immersed solid should be fine enough to enable effective interpolation of near-immersed-boundary fluid nodes onto the immersed solid. It is advised that you have at least 10 fluid domain elements on either side of the immersed solid boundary.
- Particles do not interact with the walls of an immersed solid domain; particles are tracked inside an immersed solid domain based on the fluid velocity, which is driven to match the velocity of the immersed solid domain.
- When an immersed solid straddles or crosses a periodic interface, it is not automatically replicated by the solver. If you expect this to happen, make additional copies of the immersed solid as necessary.
- If an immersed solid case is run in parallel, in some cases (including some rigid body cases) the path taken to convergence is sensitive to the number or placement of partitions, although the converged results are unaffected. You should ensure that a steady-state simulation, or each time step in a transient simulation, is sufficiently well-converged to avoid differences in results between runs with different parallel run settings.
- If an immersed solid boundary coincides with a GGI interface during a parallel run, the results of the immersed solid case may vary slightly depending on the number of partitions. This is because the solver does not assemble a coefficient matrix on the overlapping elements, and the Rhie Chow coefficients will therefore not take into account the contributions of the immersed solids on the overlapping elements.
- The immersed solid `Modified Forcing` method is targeted at predicting turbulence for the standard k-epsilon (k–ε) and the Shear Stress Transport (SST) models only. While other two-equation turbulence models can be used with immersed solids, turbulence predictions may not be accurate. The Modified Forcing method cannot be used with turbulence models that are not two-equation models (for example, Reynolds stress models, LES models, EARSM models, Zero Equation and Eddy Viscosity Transport Equation, and so on). Regardless of the immersed solid settings used, you should ensure that you understand the limitations of the accuracy of the immersed solids model in predicting turbulence for your particular application. This limitation also applies to using the `Modified Forcing` option with scalable wall functions for low Reynolds number cases.

- Except for force and torque callbacks, CEL callbacks are not supported on immersed boundaries.
- CEL callbacks for force and torque on an immersed boundary or region are not supported for rotating fluid domains.
- The force and torque callback functions may not be accurate. When calculating forces and torques on an immersed solid, the viscous contribution is typically underestimated. This is also the case when forces and torques are calculated on a rigid-body immersed solid. The accuracy of the force and torque information passed to the rigid body solver depends on the details on the problem set up, and is affected by the near-wall flow predictions. Also, callback results for forces and torques may differ on an immersed solid region from those on an immersed boundary because wall shear is not included when using a mesh region. The forces and torques calculated on immersed solids will differ between CFD-Post and CFX-Solver.
- If a fluid region or boundary intersects with an immersed solid, any callback function on that fluid region or boundary will not take into account the presence of the immersed solid.
- For low Reynolds number turbulence cases, the immersed solids turbulence model cannot accurately predict the pressure fields near the immersed boundaries.

1.2.11. Multicomponent Flow

CFX has the capability to model fluid mixtures consisting of an arbitrary number of separate physical components (or "species"). Each component fluid may have a distinct set of physical properties. The CFX-Solver will calculate appropriate average values of the properties for each control volume in the flow domain, for use in calculating the fluid flow. These average values will depend both on component property values and on the proportion of each component present in the control volume.

This section describes how to model multicomponent flow simulations, in addition to offering modeling advice.

1.2.11.1. Assumptions About Multicomponent Flow

In multicomponent flow, assume that the various components of a fluid are mixed at the molecular level, that they share the same mean velocity, pressure and temperature fields, and that mass transfer takes place by convection and diffusion. The more complex situation involving fluid interfaces, where different components are mixed on larger scales and may have separate velocity and temperature fields is called multiphase flow. This type of flow cannot be modeled with the multicomponent model. For details, see Multiphase Flow Modeling (p. 295).

It is further assumed that the diffusion velocity of a component obeys Fick's Law.

1.2.11.2. Multicomponent Flow Terminology

The following terms are used in CFX multicomponent flow. Examples to illustrate the difference between fluids and components are shown in the next section. For details, see Multicomponent Flow Examples (p. 66).

1.2.11.2.1. Pure Substance

A pure substance is a material with specified physical properties, such as density and viscosity. These properties may be numeric constants, or the CFX Expression Language (CEL) may be used to specify non-constant properties.

1.2.11.2.2. Component

A component is one part of a multicomponent fluid. When a pure substance is used as part of a multicomponent fluid it is referred to as a component of that fluid. A fixed composition mixture could also be treated as a component of another fluid. The properties of a component are calculated from the mass fractions of the constituent materials and are based on the materials forming an ideal mixture below. For details, see Ideal Mixture (p. 65).

1.2.11.2.3. Multicomponent Fluid

A multicomponent fluid contains two or more components and its properties are calculated from those of the constituent components. The components are assumed to be mixed at the molecular level and the properties of the fluid are dependent on the proportion of its components.

The components can exist in fixed mass fractions (**Fixed Composition Mixture**) or in variable mass fractions (**Variable Composition Mixture**). For **Variable Composition Mixtures**, the proportions of each component present may vary in space or time. This may be caused by the conversion of one component to another through a chemical reaction, such as combustion, driven by diffusion or caused by specifying different proportions at different boundaries or in the initial conditions.

In the solution to a multicomponent simulation, a single velocity field is calculated for each multicomponent fluid and written to the results file. Individual components move at the velocity of the fluid of which they are part, with a drift velocity superimposed, arising from diffusion.

The properties of multicomponent fluids are calculated on the assumption that the constituent components form an ideal mixture below. For details, see Ideal Mixture (p. 65).

1.2.11.2.4. Fluid

A fluid is a general reference to any pure substance or multicomponent fluid.

1.2.11.2.5. Additional Variable

Additional Variables are non-reacting, scalar components that are transported through the flow. Additional Variables can be used to model the transport of a passive material in the fluid flow, such as smoke in air and dye in water, or to model other scalar variables, such as electric field. The presence of an Additional Variable does not affect the fluid flow by default, although some fluid properties may be defined to depend upon an Additional Variable by means of the CFX Expression Language. Additional Variables may be carried by, as well as diffuse through, the fluid; they are typically specified as concentrations.

1.2.11.2.6. Ideal Mixture

An ideal mixture is a mixture of components such that the properties of the mixture can be calculated directly from the properties of the components and their proportions in the mixture.

1.2.11.2.7. Transport Equation

If the flow of a component is modeled using a transport equation, then it is transported with the fluid and may diffuse through the fluid, and its mass fraction is calculated accordingly.

1.2.11.2.8. Constraint Equation

If the flow of a component is modeled using a constraint equation, its mass fraction is calculated to ensure that all the component mass fractions sum to unity, that is, the mass fraction of the component is set equal to the total mass fractions of the other components in the same fluid subtracted from unity.

1.2.11.2.9. Algebraic Equation

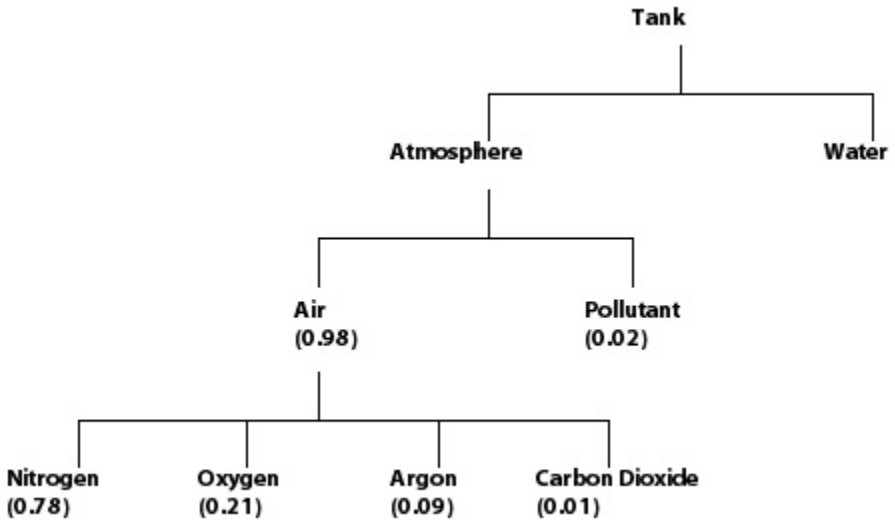
If a component is modeled using an algebraic equation, then its mass fraction is calculated from an expression that you can specify using the CFX Expression Language (CEL).

1.2.11.3. Multicomponent Flow Examples

1.2.11.3.1. Example 1: Multicomponent Multiphase

All pure substances and multicomponent fluids are specified in the **Materials** details view. This can be accessed from the **Materials** workspace in CFX-Pre. A description of the use of the **Materials** details view is available in Material Details View: Pure Substance in the *CFX-Pre User's Guide*. In this section, the relation between fluids and components is discussed.

The following example shows a CFD simulation of a tank that contains two fluids, `Water` and `Atmosphere`. `Water` is a pure substance (single-component fluid) while `Atmosphere` is a **Variable Composition Mixture** containing `Air` and `Pollutant`. `Pollutant` is a **Pure Substance** while `Air` is a **Fixed Composition Mixture** made from other pure substances.



You cannot nest **Fixed and Variable Composition Mixtures** further than as shown in this example (for example, **Fixed Composition Mixtures** cannot be made from other **Fixed Composition Mixtures**).

Because there is more than one fluid, this is a multiphase simulation (regardless of whether or not the fluids have the same phase). For details, see Multiphase Flow Modeling (p. 295).

1.2.11.3.2. Example 2: Smoke in Air

This example illustrates the relationship between components, fluids, and Additional Variables.

Smoke in `Air` can be considered in a number of different ways:

- The flow can be considered to be that of a single fluid, `Air`. The smoke is modeled as an Additional Variable using a transport equation. For details, see:
  - Additional Variable (p. 65)
  - Transport Equation (p. 66).
- Using this model, it is assumed that the presence of the smoke does not affect the air flow. It is, therefore, most suitable for small concentrations of small smoke particles.
- The flow can be considered to be a multicomponent fluid, consisting of two components: `Smoke` and `Air`. Using this model, the presence of `Smoke` will affect the flow of the fluid through the fluid properties. However, the smoke and air are assumed to have a common velocity. This model could be used for larger concentrations of small smoke particles.
- The flow can be considered to be that of a single fluid, `Air`, with the smoke particles modeled using particle tracking.

1.2.11.4. Component Domain Settings

Component mass fractions can be calculated by the CFX-Solver using a **Transport Equation**, an **Algebraic Equation** and the **Constraint** condition. Additionally, they can be based on a flamelet library or they can use the algebraic slip model. For details, see:

- Flamelet Libraries in the *CFX-Solver Theory Guide*
- Algebraic Slip Model (ASM) (p. 350).

If an **Algebraic Equation** is used, you should enter a CEL expression that evaluates to values between 0 and 1 throughout the domain. This describes the mass fraction of the component. When more than one component uses an **Algebraic Equation**, the sum of the mass fractions at any location must be less than or equal to 1.

If the **Transport Equation** option is used, CFX solves for the transport and diffusion of the component. At least one component must be solved for using a transport equation.

The mass fractions of the components must always add up to 1. This is enforced by specifying one component as the constrained component. The mass fraction of this component is always the difference between 1 and the sum of the mass fractions of all other components. Exactly one component must use **Constraint** as the option. This results in the following restrictions:

- When only two components exist, one component must use the **Transport Equation** option and one must use the **Constraint** option.

- When more than two components exist, one component must use the **Transport Equation** option, one must use the **Constraint** option and the remaining components can use either the **Transport Equation** option or the **Algebraic Equation** option.

An **Automatic** option is also available; this will set the option to either **Transport Equation** or **Library**. If the combustion model is:

- `None`
- `Eddy Dissipation`
- `Finite Rate Chemistry`
- `Finite Rate Chemistry and Eddy Dissipation`

then **Automatic** will translate into **Transport Equation**. If the combustion model is:

- `Laminar Flamelet with PDF`
- `Partially Premixed and Laminar Flamelet with PDF`

then **Automatic** will translate into **Library**. For details, see Flamelet Libraries in the *CFX-Solver Theory Guide*.

The **Automatic** option will not set any component as the **Constraint** component, so you must change at least one component from **Automatic** to **Constraint** in CFX-Pre.

1.2.11.4.1. Algebraic Slip

The **Algebraic Slip** model is a multiphase model. For details, see Algebraic Slip Model (ASM) (p. 350).

1.2.11.4.2. Kinematic Diffusivity

If a **Transport Equation** is being solved, you can enter a value for the **Kinematic Diffusivity**. If a value is not set, then either the dynamic viscosity (unity Schmidt number assumption for cases without heat transfer) or the thermal conductivity (unity Lewis number assumption for cases with heat transfer) of the fluid is used to derive the molecular diffusion coefficient for all components (see Multicomponent Energy Diffusion (p. 69) for further information).

Turbulent diffusivity is not affected by this setting. A fluid property is a weighted average of the component properties based on the local mass fractions. If you would like to set the dynamic diffusivity, simply set the kinematic diffusivity as an expression equal to the intended value of the dynamic diffusivity divided by the density.

1.2.11.4.3. Turbulent Flux Closure

If the component model is set to either transport equation or to Algebraic Slip, then you may optionally enable the specification of the **Turbulent Flux Closure**. When the specification is not enabled, the default is to apply the same settings as for the energy equation, see Heat Transfer (p. 55).

The **Turbulent Flux Closure** option can be used for specification of component-dependent turbulent Schmidt numbers, which may be non-constant using expressions (CEL).

1.2.11.5. Boundary Conditions

The implementation of boundary conditions for multicomponent flow is very similar to that for single-component flow.

- Boundary Condition Modeling (p. 111)

  (Physical description)

- Boundary Conditions in the *CFX-Solver Theory Guide*

  (Mathematical models)

However, you need to specify mass fractions of components on inlet and opening type boundary conditions.

1.2.11.6. Multicomponent Energy Diffusion

This option is used to control the numerical treatment for the energy diffusion term for multicomponent fluids. There are two modes: The generic formulation and the unity Lewis number formulation, which are valid only under certain conditions. For details, see Multicomponent Energy Diffusion in the *CFX-Solver Theory Guide*. The less general mode can significantly reduce the numerical cost for assembly of the energy equation, in particular when the fluid has a large number of components.

Three options are available:

- `Automatic`

  forces the solver to use the numerically efficient unity Lewis number assembly when physically valid, and use the generic assembly otherwise; this is the default.

- `Generic Assembly`

  forces the solver to use the general enthalpy diffusion term; the default molecular diffusivity for mass fractions is equal to the fluid viscosity (unity Schmidt number, $Sc = 1$). Use this option to revert to the behavior of CFX release 5.7.1 or previous.

- `Unity Lewis Number`

  forces the solver to use the unity Lewis number formulation; the default molecular diffusivity for mass fractions is thermal conductivity divided by heat capacity (unity Lewis number, $Le = 1$).

- `Generic Molecular and Unity Turbulent`

  forces the solver to use the general formulation for molecular transport and the unity Lewis number formulation for turbulent transport; the default molecular diffusivity for mass fractions is equal to the fluid viscosity (unity Schmidt number, $Sc = 1$). Use this option to revert to the behavior of CFX release 5.7.1 or previous.

- `Unity Molecular and Generic Turbulent`

forces the solver to use the unity Lewis number formulation for molecular transport and the general formulation for turbulent transport; the default molecular diffusivity for mass fractions is thermal conductivity divided by heat capacity (unity Lewis number, $Le = 1$).

Note:

Forcing `Unity Lewis Number` mode when not appropriate may lead to inconsistent energy transport. For example, this option should not be used in combination with user-specified component diffusivities. Therefore, using the `Unity Lewis Number` option is not recommended.

For multicomponent fluids, the effective assembly options for molecular and turbulent energy transport, respectively, are reported to the CFX-Solver Output file in the section *Multi-Component Specific Enthalpy Diffusion*. For details, see Multicomponent Specific Enthalpy Diffusion (MCF) in the *CFX-Solver Manager User's Guide*.

For single-component fluids or for isothermal flow the option has no effect. In the latter case the fluid viscosity is applied as the default molecular diffusivity for components.

1.2.12. Additional Variables

Additional Variables are non-reacting, scalar components that are transported through the flow, or through a solid (including the solid portion of a porous domain). They can be used to model, for example, the distribution of dye through a liquid, or smoke from a fire. Ansys CFX typically interprets Additional Variables as concentrations within the fluid domain.

Additional Variables can be set up as either algebraic equations, Poisson equations, or transport equations. For algebraic Additional Variables, you must provide an expression for its value throughout the domain.

Mathematical information on the Additional Variable transport equation is available in Transport Equations for Additional Variables in the *CFX-Solver Theory Guide* and Additional Variable Transfer Through the Fluid and Solid in the *CFX-Solver Theory Guide*.

1.2.12.1. Kinematic Diffusivity

For most applications, the transport of Additional Variables is both a convective and diffusive process (including both laminar and turbulent diffusion), and you will therefore need to specify the molecular kinematic diffusivity for each Additional Variable you use. This describes how rapidly the scalar quantity would move through the fluid in the absence of convection. It is generally a function of the properties of the fluid and the medium that the Additional Variable represents, and consequently has no default value. It is usually small (of the order 10^{-5} m^2/s for smoke in air).

For convection-dominated flows, the kinematic diffusivity can have little effect because convection processes dominate over diffusion processes. You may want to specify an Additional Variable whose transport through the fluid is a purely convective process. You can neglect diffusion effects by not setting a kinematic diffusivity for an Additional Variable when you include it in your domain. For turbulent flows, the turbulent diffusion (which is a consequence of averaging the advection term) is still included even when you do not set the kinematic diffusivity.

Poisson equations are diffusive and therefore the kinematic diffusivity must be specified for a successful run.

Note:

Diffusivity of an Additional Variable can depend on what fluid it is diffusing through. It is not possible to have density or specific heat dependent on Additional Variables. A more logical approach is to use a multicomponent fluid. For details, see Multicomponent Flow (p. 64).

1.2.12.2. Turbulent Flux Closure

The **Turbulent Flux Closure** may be optionally enabled for specification for Additional Variables that are set up as a transport equation. The default when not enabled for specification is the **Eddy Diffusivity** model with the turbulent Schmidt number set to a value of 0.9. The turbulent Schmidt number may be non-constant using expressions (CEL).

1.2.12.3. Volumetric and Specific Additional Variable

Volumetric and specific Additional Variables are distinguished only by a factor of material density in that the volumetric form of an Additional Variable can be obtained from the specific form by multiplying the specific Additional Variable value by density. The CFX-Solver always directly solves for the variable using the specific form, and then calculates the volumetric values at the end of the run for postprocessing.

A volumetric Additional Variable is specified in terms of the amount of the Additional Variable per unit volume of fluid.

$$\Phi = \frac{\text{conserved quantity}}{\text{volume of fluid}} \tag{1.12}$$

The conserved quantity often has units of mass, in which case Φ is called mass concentration and you should set units that have dimensions of Mass / Length^3 (for example, kg quantity / m^3 fluid).

A specific Additional Variable is specified in terms of the amount of the Additional Variable per unit mass of fluid.

$$\Phi = \frac{\text{conserved quantity}}{\text{mass of fluid}} \tag{1.13}$$

Like the volumetric form, in most cases the conserved quantity will have units of mass, in which case Φ is called mass fraction and you should set units that have dimensions of Mass / Mass (for example, kg quantity / kg fluid).

In addition, the units for boundary condition and source specifications must be consistent with the Additional Variable units:

- Additional Variable value: Units of Φ.
- Flux boundary condition: Units of conserved quantity per unit area per unit time.
- Source value: Units of conserved quantity per unit volume per unit time.

- Source coefficient: Units of Source value per unit Φ.

CFX-Pre will automatically make the appropriate units available when you set one of these quantities. If you use an expression and provide incorrect units, CFX-Pre will not enable you to set the quantity.

At the end of the solver run, the Additional Variable will be made available in the results file in both volumetric and specific form for postprocessing purposes.

1.2.12.4. Additional Variables In Units Other Than Mass

The units of Additional Variables are more flexible than the two examples given in the previous section. In general, any units are acceptable for an Additional Variable. In this case, you just enter whatever units you like for the variable, and, if desired, also enter the appropriate diffusivity with dimensions of Length^2 / Time.

For example, to solve an Additional Variable equation for temperature (T), with units of Kelvin [K], you could create a specific Additional Variable with units of [K], and the CFX-Solver will solve the following transport equation:

$$\frac{\partial(\rho T)}{\partial t} + \nabla\cdot(\rho \boldsymbol{U} T) = \nabla\cdot\left(\left(\rho\alpha + \frac{\mu_t}{Sc_t}\right)\nabla T\right) + S_T \tag{1.14}$$

where α is the thermal diffusivity (that is, $a=k/\ r\ c_p$), ρ is the fluid density, μ_t is the turbulent eddy viscosity and Sc_t is the turbulent Schmidt number. In this case, the solver writes two fields that can be viewed in CFD-Post: the specific form of the variable (temperature) with units of [K], and the volumetric form (which is simply the specific form multiplied by fluid density) with units of [K kg m^{-3}].

1.2.12.5. Unspecified Additional Variables

Unspecified Additional Variables can be used when you want to calculate the value of a quantity using an algebraic expression. This type of variable can have any units you like and, if an expression is assigned to this variable, then the expression result must have the same units as the variable.

For example, you could create an Additional Variable for a pressure coefficient by creating an unspecified Additional Variable with dimensionless units and assigning it to an expression:

$$\Phi = \frac{p_{abs} - p_{ref}}{\frac{1}{2}\rho\left|\boldsymbol{V}\right|^2} \tag{1.15}$$

where p_{abs} is absolute pressure p_{ref} is the domain reference pressure or some other reference pressure state, ρ is the fluid density and $\boldsymbol{V}$ is the fluid velocity.

Note:

For cases involving multicomponent fluids, if any component-specific variables are involved in the algebraic expression for an Additional Variable, be sure to qualify those variables with their respective component names (write "<Component_Name>.<Variable_Name>", replacing "<Component_Name>" and "<Variable_Name>" as appro-

priate). Any unqualified variables are assumed to be in the context of the mixture rather than any particular mixture component.

1.2.12.6. Tensor Type

The Additional Variable's **Tensor Type** can be set to `Scalar` or `Vector`. If an Additional Variable is defined as type `Vector`, the components of a vector algebraic equation can be defined at the domain level.

Vector Additional Variables cannot be directly referenced in CEL expressions. The syntax for referencing a component of a vector Additional Variable is as follows:

```
<Component Name>.<Additional Variable Name>_x
```

1.2.13. Non-Newtonian Flow

Some fluids are non-Newtonian; that is, they do not obey the simple linear relationship between shear stress and shear strain rate. Many practical fluids fall into this class, and their behavior is generally well understood and described using various mathematical models.

Ansys CFX has several models for calculating viscosity based on shear strain rate. These models are listed in Table 1.2: Non-Newtonian Models (p. 73) where μ is the dynamic viscosity, and $\dot{\gamma}$ is the shear strain rate.

Table 1.2: Non-Newtonian Models

Model	Description	Settings
Bingham	This is a model for viscoplastic fluids. $$\mu=\frac{\tau_Y}{\dot{\gamma}}+K$$ Examples of viscoplastic fluids include tomato paste and tooth paste. A few electro-rheological fluids can be modeled as Bingham fluids with the yield stress as a function of the intensity of the electric field, or the electric current. Note that Ansys CFX supports only a single-valued yield stress. The yield stress may be evaluated as a CEL expression with, for instance, values that vary over time or iteration. Yield stress values that vary spatially are not supported.	Yield Stress: τ_Y
		Viscosity Consistency: K
		Minimum Shear Strain Rate
		Maximum Shear Strain Rate

Model	Description	Settings
Bird Carreau	This is a model intended for shear-thinning fluids. $$\mu=\mu_{\infty}+\frac{\left(\mu_0-\mu_{\infty}\right)}{\left(1+\left(\lambda\dot{\gamma}\right)^2\right)^{\frac{1-n}{2}}}$$ The model reverts to a Newtonian behavior of $\mu=\mu_0$ for $n=1$, or $\lambda=0$. Examples of shear-thinning fluids include applesauce, banana puree, and orange juice concentrate.	Low Shear Viscosity: μ_0
		High Shear Viscosity: μ_{∞}
		Time Constant: λ
		Power Law Index: n
Carreau Yasuda	This is a generalization of Carreau's original model. $$\mu=\mu_{\infty}+\frac{\left(\mu_0-\mu_{\infty}\right)}{\left(1+\left(\lambda\dot{\gamma}\right)^a\right)^{\frac{1-n}{a}}}$$	Low Shear Viscosity: μ_0
		High Shear Viscosity: μ_{∞}
		Time Constant: λ
		Power Law Index: n
		Yasuda Exponent: a
Casson	This model is a variation of the Bingham model for viscoplastic fluids with a square root/quadratic dependency. $$\sqrt{\mu}=\sqrt{\frac{\tau_Y}{\dot{\gamma}}}+\sqrt{K}$$	Yield Stress: τ_Y
		Viscosity Consistency: K
		Minimum Shear Strain Rate
		Maximum Shear Strain Rate

Model	Description	Settings
Cross	This is a model intended for shear-thinning fluids. $\mu=\frac{\mu_0}{1+\left(\lambda\dot{\gamma}\right)^n}$	Low Shear Viscosity: μ_0
		Time Constant: λ
		Power Law Index: n
Herschel Bulkley	This is a model for viscoplastic fluids that, after yield, exhibits a power law behavior in shear stress versus shear strain rate. $\mu=\frac{\tau_Y}{\dot{\gamma}}+K\left(\lambda\dot{\gamma}\right)^{n-1}$	Yield Stress: τ_Y
		Viscosity Consistency: K
		Minimum Shear Strain Rate
		Maximum Shear Strain Rate
		Time Constant: λ
		Power Law Index: n
Ostwald de Waele	This is perhaps the most popular viscosity model because of its simplicity. However, it does not have bounded behavior either on the low or high shear limits, unlike Carreau's models. $\mu=K\left(\lambda\dot{\gamma}\right)^{n-1}$	Viscosity Consistency: K
		Minimum Shear Strain Rate
		Maximum Shear Strain Rate
		Time Constant: λ
		Power Law Index: n

At the start of a steady-state calculation of non-Newtonian flow, the CFX-Solver may take several time steps to 'get going.' This is probably due to excessively high fluid viscosities resulting from small or

zero initial velocities. In such cases, you can promote faster convergence by specifying a sensible initial velocity field.

Note:

If you want to model a non-Newtonian fluid without using any of the available models, you can specify a viscosity by value and use an expression. Note that shear strain rate is available as a CFX system variable. When developing your expression, you should consider non-physical shear strain rates and the possibility of divide-by-zero errors. One way of doing this is to use the available built-in functions `min` and `max` to bound the values of shear strain rate.

1.2.14. Coordinate Frames

1.2.14.1. Global Coordinate Frame (Coord 0)

In addition to being the default coordinate frame in CFX-Pre, the CFX-Solver always computes solutions in the global Cartesian coordinate frame, `Coord 0`. This coordinate frame's origin is located at [0 0 0] and has the Cartesian basis vectors <1 0 0>, <0 1 0> and <0 0 1>. Unless specified otherwise all material properties, boundary conditions, source terms and initial conditions are calculated in the global coordinate frame.

If you are using CEL expressions, the built-in variables `xGlobal`, `yGlobal`, and `zGlobal` can be used to refer to the three coordinate indexes of the global coordinate frame.

1.2.14.2. Local Coordinate Frames

In some cases, it may be useful to specify quantities using coordinates from a coordinate frame other than the global coordinate frame `Coord 0`. This is done by creating a local coordinate frame and applying it to a particular location (such as a boundary condition, initial condition, subdomain, domain) or other object that supports local coordinate frames (for example, a monitor point). For details on creating local coordinate frames in CFX-Pre, see Coordinate Frames in the *CFX-Pre User's Guide*.

Note:

- A variable, CEL expression, or profile data that is used in the specification of some quantity is evaluated with coordinates treated as being local, and all other non-scalar variables treated as being global. Non-scalar CEL functions (such as `torque`, `force`) are treated as being local. The quantity that was specified by the variable or CEL expression, once evaluated, will be treated as being local, if applicable (that is, if it is a vector or tensor quantity, or component thereof).
- If a domain uses a local coordinate frame other than the global coordinate frame, locations within that domain, such as subdomains and boundary conditions, do not necessarily use the same coordinate frame, because each location has its own independent coordinate frame specification.

Local coordinate frames are particularly useful for setting quantities that are not aligned with the global coordinate system, such as:

- Cartesian velocity components for initial or boundary conditions
- Cartesian flow directions for boundary conditions
- Cartesian momentum source components
- Spatially varying scalar boundary conditions
- Spatially varying properties

Note:

Cylindrical velocity components and flow directions use either a local axial coordinate direction specification or the domain rotation axis for RFR domains, as applicable. The axis specification is made using Cartesian coordinates that are interpreted in the local coordinate frame.

A local coordinate frame can have rotational frame motion applied (see Frame Motion Settings in the *CFX-Pre User's Guide*). A boundary condition can use such a coordinate frame to cause the *location of application* of profile data (or CEL expressions) to move in accordance with the coordinate frame motion.

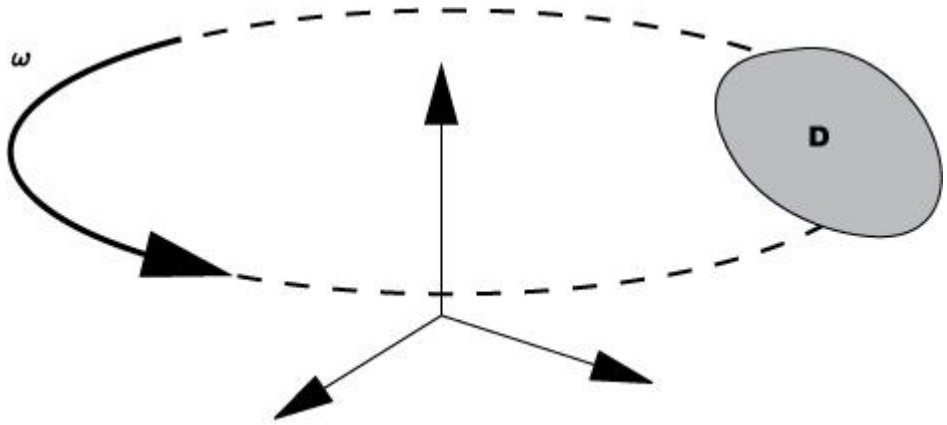
1.2.14.3. Cartesian Coordinate Frames

Cartesian vector, vector component, and tensor quantities can be associated with any local Cartesian coordinate frame in which they are meant to be interpreted by the solver.

1.2.15. Rotating Frames of Reference (RFR)

Rotating Frames of Reference (RFR) are useful for rotating fluid machinery applications, such as pump impeller or turbine blade problems. CFX-Pre enables you to specify a domain that is rotating about an axis. When you specify a **Rotating Frame of Reference**, you must also specify the angular velocity, ω, which can be a function of the built in CEL variable time (`t`) in a transient simulation. To determine the direction of the rotation, use the right-hand rule.

Figure 1.1: Direction of Rotation



When a domain with a rotating frame is specified, the CFX-Solver computes the appropriate Coriolis and centrifugal momentum terms, and solves a rotating frame total energy equation. Additional mathematical information is available in Rotational Forces in the *CFX-Solver Theory Guide*.

You can create simulations with multiple reference frames. For details, see Domain Interface Modeling (p. 233).

Note:

For any CCL expression that is defined for a location (such as a domain, boundary, subdomain), coordinates (x, y, z, r, theta) are interpreted in the local frame but variables are normally interpreted in the global frame.

1.2.15.1. Alternate Rotation Model

The alternate rotation model is a model for the advection term in the momentum equations. Instead of advecting the relative frame velocity, the flow solver advects the absolute frame velocity.

The main advantage of this model comes into play when the absolute frame velocity is a constant, but the relative frame velocity has a high swirl component. In this case, advecting the relative frame velocity has a high component of error, while advecting a constant absolute frame velocity will have much less error. If the relative and absolute frame velocities are both changing equally, for example, inside a blade passage on a rotating machine, then the alternate rotation model has a similar level of numerical error compared to the default advection model.

The alternate rotation model makes a significant reduction in numerical error when the absolute frame flow is essentially a constant flow parallel to the axis of rotation. For example, the approach flow to a fan or propeller is nearly constant in the absolute frame, but highly rotating flow in the relative frame. At very large radii, small errors in the advection model of the relative frame flow become large errors in the computed flow in the absolute frame. When the alternate rotation model is used in this situation, the numerical error is greatly reduced, because the absolute frame velocity is close to constant. In fact, the numerical error in the advection model reduces to zero as the absolute frame flow becomes axially constant, independent of the radial location and rotation rate.

Conversely, the model can increase errors when, for example, the exit stream is highly swirling. Errors are magnified as the length of the inlet or outlet regions is increased. The choice of whether to apply the model therefore depends on the nature of the inlet and outlet flow conditions, and the length of the inlet and outlet regions.

It has been found that in most realistic flow situations, the model reduces numerical error (or at least does not increase numerical error). The one exception is solid body rotation, where the relative frame flow is constant (V=0) and the absolute frame flow is nonzero and varying with radius.

For more details, see Alternate Rotation Model in the *CFX-Solver Theory Guide*.

1.2.16. Electric Field

There are 2 options for the modeling of electric field effects:

- The `Electric Potential` option is formulated for conductive media only and it assumes a quasi-steady field. Other quantities of interest are derived from it such as `Electric Potential`, `Electric Field`, `Current Density`, and `Joule heating`.
- The `User Defined` option can be used to directly specify an externally imposed electric field. This option does not require any boundary conditions.

If an electric model is active in combination with other models, it may have some direct influence on those models as well. For example:

- If the magnetic model is also active, and the material is conductive, the Lorentz force is added to the momentum equations. For details, see Magnetohydrodynamics (MHD) in the *CFX-Solver Theory Guide*.
- If the model includes heat transfer, the Joule heating quantity is automatically added to the energy equation and is defined by:

$$Q_{joule} = \boldsymbol{J} \cdot \boldsymbol{E} \tag{1.16}$$

For multiphase flows, the electric surface current at the interface between the phases is ignored and a volume fraction weighted electrical conductivity is used.

Electric fields can be specified at boundaries when the `Electric Potential` model has been activated for a domain. For details, see Electromagnetic Model. The following options are available:

- `Voltage`

  Specifies the **Voltage** at the boundary.

- `Ground`

  Equivalent to `Voltage` with a **Voltage** equal to `0`.

- `Flux in`

  Specifies the **Electric Current** at the boundary according to the equation:

$$-n \cdot J = J_b \tag{1.17}$$

Where J_b is the normal component of the electric current density at the boundary with incoming electric current being positive.

- `Zero Flux`

  Equivalent to `Flux in` with J_b equal to `0`.

- `Electric Current Transfer Coefficient`

  This boundary condition can be used to model an additional electric resistance by the equation:

  $$-n \cdot J = Transfer\ Coefficient(Voltage - External\ Voltage) \tag{1.18}$$

  The **Transfer Coefficient** could be interpreted as a thin material of electrical conductivity, σ, and of finite thickness, t, such that:

  $$Transfer\ Coefficient = \frac{\sigma}{t} \tag{1.19}$$

- `Conservative Interface Flux`

  This boundary condition is used at domain interfaces to indicate that the electric current will flow between the current boundary towards the other side of the interface. This means, that a similar `Conservative Interface Flux` option must be specified on the boundaries on the other side of the interface.

- `Electric Field Contact Resistance`

  This model represents an electric resistance between two sides of a domain interface.

1.3. Sources

Sources are optional terms that you may attach to most equations in order to model additional or specialized physical processes. They can be specified on boundaries, subdomains, injection regions, and points as described in the following sections.

This section describes:

1.3.1. Locators for Sources

1.3.2. Types of Sources

1.3.3. Multiplying Sources by Porosity

The standard governing equations on these locators are modified to include user-defined source terms. For details, see Sources in the *CFX-Solver Theory Guide*.

1.3.1. Locators for Sources

1.3.1.1. Boundary Sources

Boundary sources permit the specification of sources as fluxes (source per unit area) on boundary condition surfaces. This applies to all equations, including continuity.

Total sources are also available on boundaries, for which the solver assumes a uniform source density in order to obtain the flux-based quantity.

Momentum sources are not currently available on boundaries.

1.3.1.2. Subdomains

Subdomains are regions of space in a domain in which volumetric sources of mass, momentum, energy, turbulence, Additional Variables, mass fractions and radiation can be specified. They can also be used to model flow resistance. Many subdomains can be created in each domain.

Subdomain regions are defined in the same way as domains, which is from 3D primitives. A 3D primitive used for a subdomain must also be contained in the parent domain. For details, see Mesh Topology in CFX-Pre in the *CFX-Pre User's Guide*.

Sources on subdomains may be specified as either volumetric quantities (source per unit volume) or as the total source. If a total source is specified, the solver assumes a uniform source density over the subdomain in order to obtain the volumetric quantity.

The total source option is not supported for momentum sources.

1.3.1.3. Injection Regions

As a convenient alternative to specifying numerous mass sources or sinks on a surface or within a volume, you can define one or more injection regions. An injection region allows for varying conditions at each of its injection (source/sink) positions. For details, see Injection Regions in the *CFX-Pre User's Guide*.

1.3.1.4. Source Points

Sources can also be specified at a point. The specified source is distributed among the vertices of the element in which the point is located. The effect of the source will therefore become more pronounced as the mesh is refined.

Momentum sources are not currently available at points.

1.3.2. Types of Sources

1.3.2.1. General Sources

A general source is a source term that is simply added to a general scalar equation. This includes the **Energy**, **Additional Variable**, **Turbulence** and **Mass Fraction** equations.

There is no restriction as to what the source term may depend on, although additional convergence control may be necessary for strongly-varying terms.

Sources for a particular component in a multicomponent fluid do not introduce any additional mass and so the sum of all component sources (including the constraint) must be zero. Note that sources may not be specified for the constraint component; the implied source for the constraint is equal to the negative of the sum of all specified component sources. For this reason, component sources might be useful to model single phase reactions, but multiphase reactions (in which a particular

component is transferred from one phase to another) require the use of mass (continuity) sources. Further details on mass sources, and particular issues related to multicomponent fluids, are provided in Mass (Continuity) Sources (p. 84).

General sources can be applied on the solid part of a porous domain. Such a source can be multiplied by the solid fraction in the porous domain.

1.3.2.1.1. Source Coefficient / Total Source Coefficient

The converged solution depends only on the source value. For nonlinear sources, a linearization coefficient may be specified to improve convergence. This is specified by the optional **Source Coefficient** or **Total Source Coefficient**, which should be a reasonable approximation to the derivative

$$\text{Source Coefficient}=\frac{\partial S_\varphi}{\partial \varphi} \tag{1.20}$$

where S_φ is the source or total source term. Note that this derivative must be negative for stability purposes. If a positive value is specified, the solver will reverse the sign. Leaving the coefficient unspecified is equivalent to using a zero coefficient. For the energy equation, the derivative is taken with respect to temperature.

Although it is possible to set a coefficient when the source term is zero, it is not normally of practical use; it will change convergence rates without adding any source.

1.3.2.2. Momentum Sources

CFX supports three types of momentum sources:

- isotropic loss model
- directional loss model
- general momentum sources

The theory for these momentum source types is given in Momentum Sources in the *CFX-Solver Theory Guide*. This section gives some additional practical tips on how to use the sources.

1.3.2.2.1. Isotropic and Directional Loss Models

Isotropic and directional loss models are useful for modeling porous momentum losses either with or without the volume effect of porosity. The loss model formulations are the same in both cases, but are set up in different locations in the user interface. If the volume effect of porosity is included, you need to create a porous domain and set the loss model details on the domain details view. If it is not included, the momentum loss model is set on a subdomain. For details on how the volume effect of porosity modifies the governing equations, see Domain Type in the *CFX-Pre User's Guide*. Note also that loss coefficients may be derived from experimental or analytical data using either true or superficial velocity. When applying a loss model, you must be aware of which velocity was used to derive the loss coefficients.

- When applying a loss model on a subdomain of a fluid domain, the solver assumes the coefficients multiply the superficial velocity, so if the coefficients were derived using true velocity,

you will need to adjust the coefficients accordingly. The permeability must be multiplied by the porosity, and the loss coefficient must be divided by the porosity squared.

- When applying a loss model on a porous domain, you may choose whether the coefficients multiply the true or superficial velocity.

For many applications, a certain resistance loss is desired in a specified direction, with flow inhibited in the transverse directions. This is the case when you want to model the effect of flow straightening devices such as honeycombs, porous plates, and turning vanes without modeling the details of the flow around the obstacles. For situations like this, CFX enables the independent specification of loss for the streamwise and transverse directions.

For both the streamwise and transverse directions, both types of the loss formulations available for the isotropic loss model are available. In many cases, however, the loss coefficients are known only for the streamwise direction, and you know only that the flow is inhibited in the transverse direction. When this occurs, you may select the **Streamwise Coefficient Multiplier** for the transverse loss model. In this case, the transverse coefficients are taken to be the specified factor times the streamwise coefficients. (If the streamwise loss includes a permeability, the implied transverse permeability is divided, not multiplied, by this factor). The transverse multiplier is typically taken to be about 10-100.

In some cases, you may want to inhibit only the transverse flow without having any streamwise loss. In this case, the **Zero Loss** option may be selected for the streamwise loss. Of course, if this is chosen, the **Streamwise Coefficient Multiplier** is not appropriate for the transverse loss, because it will result in zero transverse loss.

In all cases, the directional loss model requires the streamwise direction to be specified. It may be described in either Cartesian or Cylindrical coordinates.

1.3.2.2.2. General Momentum Source

Momentum sources can be specified directly in terms of a force per unit volume in a specified direction. To obtain good convergence when the source is a function of velocity, the source should be linearized by including a **Momentum Source Coefficient**. In CFX, scalar coefficients may be specified as representative of the tensor:

$$-\frac{\partial p}{\partial x} = K\, U_x \tag{1.21}$$

Typically, the scalar coefficient is chosen to be the maximum of the three principal components of this tensor.

The **General Momentum Source** option is useful for applications that are not covered by the isotropic loss model or the directional loss model. For example, the velocity in a region may be forced to be a specified value (Dirichlet velocity condition) by setting C to a large number, for example, 10^5 kg m^{-3} s^{-1}, for the following:

$$\begin{aligned} S_x &= -C\left(U_x - U_{x,\mathrm{spec}}\right) \\ S_y &= -C\left(U_y - U_{y,\mathrm{spec}}\right) \\ S_z &= -C\left(U_z - U_{z,\mathrm{spec}}\right) \end{aligned} \tag{1.22}$$

and selecting $-C$ as the Momentum Source Coefficient.

When setting a General Momentum Source, there are two optional parameters that may be set:

- **Redistribute in Rhie Chow** indicates whether S is included in the Rhie Chow redistribution algorithm (see Equation 11.32 in Pressure-Velocity Coupling in the *CFX-Solver Theory Guide*).
- **Include Coefficient in Rhie Chow** indicates whether C is included in the Rhie Chow coefficient (see Equation 11.29 in Pressure-Velocity Coupling in the *CFX-Solver Theory Guide*).

The defaults for both of these parameters are false, but there are cases when they should be activated. For example:

- When setting a momentum source to give a specified velocity (for example, when modeling solidification, a momentum source that scales with solid mass fraction can be used to force the solid to move at the belt velocity), **Include Coefficient Rhie Chow** should be set in order to ensure that the mass flows are consistent with the specified velocity.
- When the momentum source is meant to induce a pressure change (for example, a fan model), set both parameters to true to avoid pressure wiggles near the subdomain boundary.

1.3.2.2.2.1. Postprocessing the Momentum Sources

If a momentum source has been applied, a new variable [<fluid>.]User Momentum Source.Bforce may be available in the results file. This variable represents the sum of:

1. Any momentum sources for the fluid defined using a Loss Model (Isotropic or Directional), plus
2. Any momentum sources for the fluid defined by a General Momentum Source for which the **Redistribute in Rhie Chow** check box is selected.

Momentum sources defined by a General Momentum Source where the **Redistribute in Rhie Chow** check box is not selected (the default) are not included in this variable.

1.3.2.2.3. Immersed Solids Sources

The presence of an immersed solid in the flow field is modeled through a body force similar to the general momentum source.

The velocity in a fluid region that lies inside the immersed solid is then enforced to be the same as the velocity of the immersed solid through a body source given in Equation 1.22 (p. 83). The momentum source coefficient is calculated automatically inside the solver by taking account of advection and diffusion contributions from the mesh elements in the fluid region that lies inside the immersed solid. You can apply a scaling factor to the solver-calculated source coefficient to control how strongly the fluid velocity is forced to match the immersed solid velocity; this affects accuracy and robustness; for details, see Immersed Solid Control in the *CFX-Pre User's Guide*.

1.3.2.3. Mass (Continuity) Sources

A source term to the continuity equation introduces additional fluid into the simulation. The amount of fluid introduced can be specified as `Fluid Mass Source` (on subdomains), `Fluid Mass Flux` (on boundaries), or `Total Fluid Mass Source`. A fluid mass source is the mass source per unit volume; therefore, the total mass of fluid introduced will depend on the volume of the

subdomain in which the source term is applied. Similarly, a fluid mass flux is the mass source per unit area. A total fluid mass source is the total mass of fluid introduced.

Continuity sources differ from other equation sources in that they induce corresponding secondary source terms in all other equations. Therefore, in addition to specifying the mass source itself, the values of velocity, temperature, and so on, must also be specified. These values are used to compute the secondary source terms only when the mass source is positive and have a function similar to boundary condition values that are specified at an opening. In particular, for mass outflow (or negative continuity source) fluid is transported out of the domain with local values of temperature, velocity, turbulence quantities, and so on, while for mass inflow (or positive continuity source), fluid is transported into the domain with specified values. Depending on the simulation, the following variables must be set for the 'inflow' (positive mass source) case:

- Additional Variables

  If the parent domain includes Additional Variables that are solved using a transport equation, then you must set the level of the Additional Variable in the fluid that is introduced into the domain. Note that there is no corresponding secondary source term for equations with no advection term (such as diffusive and Poisson Additional Variables). Therefore, values for these variables need not be specified.

- Component Mass Fractions

  For fluids that are **Variable Composition Mixtures**, you must set the mass fraction values for the fluid that is introduced into the domain. The **Constraint** component will take a mass fraction such that the sum of the mass fractions is unity.

- Temperature

  You must set a temperature for a continuity source if heat transfer is modeled in the domain. The property recipes of the interior material will be used automatically when needed to convert temperature to enthalpy.

- Velocity

  You must set **Cartesian Velocity Components** for a continuity source to describe the direction in which the new fluid is moving.

- Eddy Dissipation and Turbulence Intensity

  You must set the turbulence quantities for a continuity source when the domain uses a turbulence model (that is, when `Laminar` is not specified).

You may provide expressions for the scalar variable values introduced with the mass source. An expression for a variable associated with the mass source may depend on other variables specified for the same fluid mass source, but care is required to avoid recursion.

If the continuity source is negative (a mass sink), the values you specify will be ignored by the solver, and local solution variables will be used instead (like at an outlet or the outflow portion of an opening). However, CFX-Pre still requires that you supply values for these variables. This is because it does not know (in general) whether the mass source will evaluate to a positive or negative number.

If the simulation involves multicomponent fluids or heat transfer, however, you may force the solver to use the specified values of mass fraction and temperature even if the mass source is negative. This is controlled by the **MCF/Energy Sink Option** setting on the **Sources** tab in CFX-Pre. The **MCF/Energy Sink Option** setting can be set to one of the following:

- `Local Mass Fractions and Temperature` - The local solution values are used for all variables.
- `Specified Mass Fractions and Local Temperature` - Specified mass fractions are used when the mass source is positive or negative. The enthalpy is evaluated using the specified mass fractions and temperature when the mass source is positive and the specified mass fractions and local temperature when the mass source is negative.
- `Specified Mass Fractions and Temperature` - Specified mass fractions are used when the mass source is positive or negative. The enthalpy is evaluated using the specified mass fractions and temperature when the mass source is positive or negative.

Additional information on theory is available in Mass (Continuity) Sources in the *CFX-Solver Theory Guide*.

1.3.2.3.1. Mass (Continuity) Source Coefficients

Nonlinear continuity sources may need to be linearized to improve convergence behavior. For continuity sources that are dependent on pressure, convergence behavior can be improved by supplying a pressure coefficient such as **Mass Flux Pressure Coefficient**, **Total Mass Source Pressure Coefficient** or **Mass Source Pressure Coefficient** where the pressure coefficient is defined as:

$$\text{Pressure Coefficient}=\frac{\partial S_C}{\partial p} \tag{1.23}$$

For example, in the instance where the continuity source is driven by the difference between the local fluid pressure and some external pressure outside the boundary:

$$S_C=K\left(P_{\text{outside}}-P_{\text{fluid}}\right) \tag{1.24}$$

then the pressure coefficient is given by $-K$.

Similarly, for continuity sources that are dependent on volume fraction, convergence behavior can be improved by supplying a volume fraction coefficient such as **Mass Flux Volume Fraction Coefficient**, **Total Mass Source Volume Fraction Coefficient** or **Mass Source Volume Fraction Coefficient**, as appropriate.

1.3.2.4. Bulk Sources

Bulk sources are available in multiphase flows. Information on the use of bulk sources is available in Bulk Sources (p. 323).

1.3.2.5. Solid Sources

Solid sources are available in a porous domain when there are physics (energy, Additional Variables) associated with the solid part of the porous domain.

Selecting the **Multiply by Solid Fraction** check box causes the equation source to be scaled by the solid fraction value $(1-\gamma)$. For example, if a porous domain has a volume porosity of 0.8 and the **Multiply by Solid Fraction** check box is selected, 20% of the source is applied to the solid; if the check box is not selected, then 100% of the source is applied to the solid.

1.3.2.6. Radiation Sources

Information on radiation sources is available in Sources (p. 475).

1.3.2.7. Particle User Sources

Information on particle user sources is available in Particle User Sources (p. 374).

1.3.3. Multiplying Sources by Porosity

Selecting the **Multiply by Porosity** check box causes the equation source to be scaled by the porosity value. For example, if a porous domain has a volume porosity of 0.8 and the **Multiply by Porosity** check box is selected, 80% of the source is applied to the fluid; if the check box is not selected, then 100% of the source is applied to the fluid.

1.4. Material Properties

The **Materials** details view in CFX-Pre is used to create and modify material properties for both pure substances and mixtures. Pure substances can be solids, liquids or gases. Liquids or gases can be used in fluid domains and solids can be used for either conjugate heat transfer models or particle tracking.

Information on how to set up pure substance or mixture properties in CFX-Pre is available in Materials and Reactions in the *CFX-Pre User's Guide*.

1.4.1. CEL Expressions

CEL expressions can be used to set certain material properties. Note that the variable `p`, when used in a CEL expression to define a material property, will be interpreted as an absolute pressure (not a relative pressure). To use reference pressure in an expression, use `Pref`.

1.4.2. Coordinate Frame

A coordinate frame can optionally be used with any material. The coordinate frame is used to evaluate properties of the material that are defined using expressions that are a function of a spatial variable (x, y, z or r, theta, z). For details, see Coordinate Frames (p. 76) and Coordinate Frames in the *CFX-Pre User's Guide*.

1.4.3. Equation of State

Equations of state can be modeled in the following ways in CFX:

- Setting **Density** directly
- Using the built in **Ideal Gas** equation

- Using the built in **Real Gas** equations
- Using the built in **IAPWS** equation
- Reading properties from a CFX-TASCflow RGP table. For details, see Real Fluid Properties (p. 491).

All properties except density and specific heat capacity, can be modeled using any valid expressions containing CFX System Variables.

1.4.3.1. Option

1.4.3.1.1. Value

This selection gives a fair amount of flexibility in modeling the variation in density. For example, it can be used to model a variety of different fluid types:

- Constant
- Variation with temperature and pressure not available as a built-in model

Variation with temperature and pressure can be set up using analytic CEL expressions or CEL user functions. Many of the popular thermal equations of state (for example van der Waals, Peng-Robinson, Beattie-Bridgemann,...) are explicit in pressure, rather than density. So, if the built-in Redlich Kwong equation is not suitable, then you will need to either create your own RGP table, or use a CEL user function to define your equation of state. Simply pass pressure and temperature into your user CEL function and then use a root finder to come up with values of density. For details, see Real Gas Property (RGP) File Format (p. 510).

The constitutive relation is set by specifying specific heat at constant pressure. Again, this can be a constant or enabled to vary as a function of temperature and pressure the same as density.

Transport properties can also be specified in the same ways as density and specific heat, but can additionally depend on other CFX System Variables, not just temperature and pressure.

The special dependency of density and specific heat capacity on local pressure or temperature can be set using the **Density Dependency** and **Specific Heat Dependency** forms. For details, see Density and Specific Heat Dependencies (p. 92).

1.4.3.1.2. Ideal Gas

For many situations involving compressible flow, the thermodynamic properties for real fluids can be closely approximated using the relationships for an ideal gas. The Ideal Gas model uses the Ideal Gas law to calculate the local density variation in the fluid. The density is automatically computed using the specified molecular weight. For details, see Dynamic Viscosity (p. 93).

The relationships are especially suited to compressible gas flows at low density under the following conditions:

- At low pressures (~ 1 bar), regardless of temperature.
- At high pressures (>> 1 bar), providing the temperature is also reasonably high (that is, > 2 x critical temperature for the gas). At temperatures lower than this, together with relatively low

pressures (such as atmospheric), significant deviations from the ideal gas approximations may exist.

1.4.3.1.3. Real Gas

The real gas option enables you to use the Redlich Kwong [85] or Peng Robinson [157] equations of state. These are three parameter state equations that can be used to model for gas density, and additionally, liquid density with the Peng Robinson equation. Minimal input is required. You need to enter the critical temperature, pressure and volume as well as the acentric factor for the pure substance. These equations are suitable for modeling subcritical gas phases and supercritical properties. Only the Peng Robinson equation will provide reasonably accurate results for subcritical liquid phase materials.

When you select a cubic equation of state the flow solver automatically generates a table of values for density at a range of temperatures and pressures. The default range is T = 100 K - 1000 K and p = 0.01 bar - 2 bar. You can change this range by selecting the table generation option and setting minimum and maximum temperature and pressure limits to what makes sense for your model.

In order to accurately model the influence of the vapor dome, the critical volume and acentric factor are provided. This enables the CFX-Solver to calculate where the vapor pressure curve cuts through the default temperature and pressure range or the range you have supplied.

Finally, to provide a default reference temperature and pressure the flow solver also requires the Boiling Temperature of the fluid at one atmosphere.

For details on real gases, see Real Gas Properties in the *CFX-Solver Theory Guide*.

1.4.3.1.4. IAPWS Library

The IAPWS equation of state is specific for water. If your model set-up requires subcooled water or superheated steam, especially with phase change, then this is the best option available.

The library materials are grouped into several temperature and pressure ranges for convenience, but you can define new materials with custom temperature and pressure ranges.

For specific temperature and pressure limitations, see IAPWS Equation of State in the *CFX-Solver Theory Guide*. For details on importing library data, see Library Materials in the *CFX-Pre User's Guide*.

1.4.3.2. Molar Mass

For all pure substances CFX requires that you provide the **Molar Mass** (relative molecular mass). The molecular mass of an element or compound is defined as the average mass of its molecules on a scale where one atom of C^{12} (isotope of carbon) has a mass of 12 units.

For an Ideal Gas or the Redlich Kwong equation of state, it is essential to set the correct Molar Mass because it is always used by the CFX-Solver. When you specify density directly using Option=Value, the Molar Mass is used only in certain situations:

- When the fluid is involved in a chemical reaction.

- When species transfer occurs for a multiphase-multicomponent simulation. For details, see Interphase Species Mass Transfer (p. 335).

In other cases, it is not essential to specify an accurate Molar Mass. However, for a multicomponent flow, the derived solution variables `<component>.Molar Concentration` and `<component>.Molar Fraction` will not be accurate unless the correct Molar Mass is set.

1.4.4. Specific Heat Capacity

The Specific Heat Capacity at constant pressure determines the amount of heat energy required to raise the temperature of a fixed mass of the fluid by 1 K. You can model Specific Heat Capacity using these options:

- By directly setting specific heat.
- Using the built in NASA Format polynomials.
- Using the built in Zero Pressure Polynomial.
- Using the Real Gas option if a real gas equation of state has been selected. For details, see Real Fluid Properties (p. 491).
- By reading properties from a TASCflow RGP table. For details, see Real Fluid Properties (p. 491).

1.4.4.1. Value

This is the most flexible option for specific heat capacity. With this option you can set specific heat to:

- A constant (Calorically Perfect)
- Varying with temperature
- Varying with temperature and/or pressure

1.4.4.2. NASA Format

This option defines specific heat as a function of temperature, in two distinct ranges, as a quartic polynomial and is available only when the equation of state is set to Ideal Gas or Real Gas. The format for the polynomial coefficients is also used by CHEMKIN [26]

The **Temperature Limit** frame expects three temperatures that define two temperature ranges for the upper and lower interval coefficients. In each interval, the same form for the polynomial is used but different coefficients can be used.

The **Upper** and **Lower Interval** coefficients are directly entered in their individual frames and correspond to the upper and lower range coefficients in the standard NASA SP-273 format. There are seven coefficients in each interval for a total of 14 coefficients. When entering these coefficients, note that the NASA format states that upper range coefficients are listed first, and lower range coefficients are listed second.

Also note that the format has changed slightly in order to accommodate proper units checking. In CFX the format followed the NASA Format to the letter: the first 7 are for the upper temperature interval, the last 7 for the lower temperature interval. In order to convert your CFX materials, you should need to load only your existing materials file into CFX-Pre and then write out the materials to a new materials file.

For each interval, the coefficients define the thermodynamic properties of the material according to the following formulae for specific heat capacity at constant pressure,

$$\frac{C_p^0}{R} = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{1.25}$$

specific static enthalpy,

$$\frac{H^0}{R} = a_1 T + \frac{a_2}{2} T^2 + \frac{a_3}{3} T^3 + \frac{a_4}{4} T^4 + \frac{a_5}{5} T^5 + a_6 \tag{1.26}$$

and specific static entropy,

$$\frac{S^0}{R} = a_1 \ln T + a_2 T + \frac{a_3}{2} T^2 + \frac{a_4}{3} T^3 + \frac{a_5}{4} T^4 + a_7 \tag{1.27}$$

1.4.4.3. Zero Pressure Polynomial

This option is available for both **Ideal Gas** equation of state or when the equation state is given by a function of temperature and pressure (Option = Value with a CEL or User Fortran function). In either case the flow solver will build a table of values for enthalpy and entropy. In the former case the enthalpy will be a function only of temperature, and in the latter it will be a function of both temperature and pressure.

For this option, the specific heat c_p is expressed in polynomial form:

$$\frac{C_p^0}{R} = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{1.28}$$

where R is the specific gas constant.

The coefficients for a_i are entered on the **Material Properties** form in CFX-Pre. The minimum and maximum temperature limits can also be supplied and define the range of applicability of the polynomial expression.

The maximum temperature limits for zero pressure polynomials are consistent in the RULES file, CFX-Pre, and the CFX-Solver (= 1000 K). Temperature limits for table generation are set by CFX-Pre: 5000 K for ideal gases, 1000 K for real gases.

1.4.4.4. Real Gas

This option is available only for materials that use a Real Gas equation of state. Ideal gas coefficients a_i must be supplied in the same way as for Zero Pressure Polynomial (p. 91) or NASA Format (p. 90). Whereas the `Zero Pressure` or NASA `Format` options allow c_p only to be a function of temperature, the `Real Gas` option takes the same coefficients as input to predict c_p as a function of both temperature and pressure.

1.4.4.5. Reference State Properties

1.4.4.5.1. Reference Temperature and Reference Pressure

When defining specific heat capacity, you must set a reference temperature and reference pressure. These quantities are used only to define a reference level from which some derived quantities, such as a change in enthalpy or entropy, are calculated. The defaults for these values are 1 atm and 25°C. However, you should set these values to values that are representative of the average temperature and pressure that will occur in your model. You should also consider table limits as well in that it is better to choose reference values that fall in the valid range for table property generation.

1.4.4.5.2. Reference Specific Enthalpy and Entropy

These values are available for input if the specific heat capacity is not given by NASA coefficients. When NASA coefficients are used, the **Reference Specific Enthalpy** and **Reference Specific Entropy** are ignored by the CFX-Solver because they are built into the expressions for enthalpy and entropy through the upper and lower interval coefficient "constant" values.

The reference specific enthalpy is the enthalpy of formation at the specified **Reference Pressure** and **Reference Temperature** (often 1 atm, 25°C). The reference specific entropy is also evaluated at the specified reference pressure and temperature.

The reference enthalpy must be accurately set for simulations involving phase change (cavitation, evaporation and so on) or chemical reactions (for example, combustion). For details, see Latent Heat (p. 329).

1.4.5. Density and Specific Heat Dependencies

For a pure substance, the flow solver allows only density and specific heat capacity to be functions of temperature or pressure.

When you select the `Value` option for density or specific heat capacity, you can use an expression to specify **Density** and/or **Specific Heat Capacity**, but the expression is allowed to depend only on pressure and temperature for liquids and gases. For a solid, they can depend on temperature or spatial location (x, y, z), but not both at the same time.

For ideal gases, the density is evaluated using the Ideal Gas law and the specific heat may be a function of temperature only.

If density, specific heat, or other properties are set as expressions that depend on pressure, the solver automatically uses the absolute pressure when evaluating the expression.

1.4.6. Table Generation Pressure and Temperature Limits

If you have defined density or specific heat capacity as a function of pressure and/or temperature using a CEL expression, User Fortran, IAPWS or Redlich Kwong equation of state then the CFX-Solver will generate tables for properties. The default temperature range is 100 K to 3000 K and the default pressure range is 0.01 bar to 10 bar.

If this range is not appropriate for the operating conditions of your model you should consider setting appropriate temperature and pressure limits for property table generation. See the **Materials** details

view in CFX-Pre and complete the **Table Generation** section. For details, see Material Properties Tab in the *CFX-Pre User's Guide*. The temperature limits specified should include enough range to convert the static temperature and total temperature range for your problem. Similarly, the pressure limits should be based on the minimum and maximum total pressure and static pressure expected in the simulation.

If the limits you enter are too narrow, then the CFX-Solver will clip properties at the bounds you have supplied. That being said though, you want to use the smallest range you can get away with because the more accurate the calculation of enthalpy will be, but the temperature and pressure in your simulation should not fall outside of the range specified.

Important:

The minimum and maximum pressures are *not* entered relative to the reference pressure (as most other pressures in CFX are). They are entered as absolute values. This is because the reference pressure is a property of a particular CFD problem and not a generic property of the material. However, an expression for density or specific heat capacity involving the CEL variable p will still use relative pressure.

The ranges are required to enable the CFX-Solver to construct a property table for the pressure-temperature-enthalpy relationship as well as entropy and, when using the Redlich Kwong and IAPWS equations of state, density and specific heat capacity.

Additional information on fluid properties and the treatment of enthalpy in Ansys CFX is available in Equations of State in the *CFX-Solver Theory Guide*.

1.4.7. Transport Properties

1.4.7.1. Dynamic Viscosity

Dynamic viscosity (μ), also called *absolute viscosity*, is a measure of the resistance of a fluid to shearing forces, and appears in the momentum equations.

1.4.7.1.1. Value

You can specify a dynamic viscosity directly using either a constant value or a CEL expression.

1.4.7.1.2. Rigid Non Interacting Sphere and Interacting Sphere Models

This model is based on elementary kinetic theory [82] and is valid for gases using user supplied equations of state or either of the built in equation of state models (Ideal Gas, Real Gas):

$$\mu=26.69\frac{\sqrt{wT}}{\Omega(T)\sigma^2} \tag{1.29}$$

where μ is dynamic viscosity in μP, w is the molecular weight in g/mol and T is temperature in Kelvin and σ is the collision diameter, in Angstroms, and is calculated using [83]:

$$\sigma=0.809\ \sqrt[3]{V_c} \tag{1.30}$$

where V_c is the critical molar volume in cm^3/mol. The Rigid Non Interacting Sphere model assumes that the collision function, Ω, is unity, while the Interacting sphere model assumes that molecular collisions contribute to the viscosity and uses non-unity Ω:

$$\Omega\left(T^*\right)=A\left(T^*\right)^B+Ce^{DT^*}+Ee^{FT^*} \tag{1.31}$$

where

$$A=1.16145$$

$$B=-0.14874$$

$$C=0.52487$$

$$D=-0.7732$$

$$E=2.16178$$

$$F=-2.43787$$

$$T^*=\frac{T}{\left(\varepsilon/k_B\right)}$$

$$\frac{\varepsilon}{k_B}=\frac{T_c}{1.2593}$$

and where T_c is the critical temperature. This model was published by Chung et al. [83] [158].

1.4.7.1.3. Non-Newtonian Model

For Generalized Newtonian fluids, you can specify a shear-strain-rate-dependent viscosity model. Several models are available. For details, see Non-Newtonian Flow (p. 73).

1.4.7.1.4. Ideal Mixture

For fixed and variable composition mixtures, the `Ideal Mixture` option is used by default when you do not set a dynamic viscosity. This option causes the viscosity of the mixture to be computed by a mass-fraction-weighted average.

1.4.7.1.5. Sutherland's Formula

This approximation for viscosity is valid for dilute gases and was obtained from the kinetic theory by Sutherland [81]. In this case viscosity varies only with temperature as:

$$\frac{\mu}{\mu_0}=\frac{T_{\text{ref}}+S}{T+S}\left(\frac{T}{T_{\text{ref}}}\right)^n \tag{1.32}$$

where μ_0 is the reference molecular viscosity, S is the Sutherland constant and is a characteristic of the gas, T_{ref} is usually 273.0 K, and n is the temperature exponent, usually set to 1.5 for most gases.

1.4.7.2. Thermal Conductivity

Thermal conductivity, λ, is the property of a fluid that characterizes its ability to transfer heat by conduction.

Thermal conductivity of gases tends to increase with increasing temperature, although it is relatively insensitive to changes in pressure, except at very high or very low pressures.

Thermal conductivity of liquids generally decreases with increasing temperature (notable exceptions are water and glycerine).

Typical values are in the range 0.005-0.5 W/mK for gases and 0.08-0.6 W/mK for liquids.

For **Fixed** and **Variable Composition Mixture**, the Thermal conductivity is determined by a mass fraction weighted arithmetic average. You can set Thermal conductivity for the mixture in the **Materials** details view instead, if you want.

1.4.7.2.1. Sutherland's Formula

Sutherland's formula for thermal conductivity is identical to that for dynamic viscosity. The only change being the reference value is now a reference thermal conductivity. This formula is still based on kinetic theory so still applies only to dilute gases.

1.4.7.2.2. Modified Euken Model

The modified Euken thermal conductivity model is available whenever you have selected Real Gas, NASA Format or Zero Pressure polynomial for specific heat capacity. You must use one of those two specific heat capacity options because the Euken model requires the zero pressure polynomial coefficients. This model is also based on the kinetic theory of gases and is valid for gases over a fairly large range of temperatures below the critical point. The Modified Euken Model is described in Poling, Prausnitz & O'Connell [84] and is given by:

$$\frac{\lambda}{\mu c_v} = 1.32 + \frac{1.77\, R_g}{c_v} \tag{1.33}$$

1.4.8. Radiation Properties

Radiation properties are required if a thermal radiation model is used. The **Absorption Coefficient** and **Scattering Coefficient** should be positive values or zero for transparent media. If you are using either the P1 or Rosseland radiation models, and at least one of the coefficients must be nonzero. The **Refractive Index** is a value greater than or equal to 1; it is approximately equal to 1 for most gases.

For a **Fixed** or **Variable Composition Mixture**, the radiation properties are determined by a mass fraction weighted arithmetic average. This is often not the most appropriate averaging method for these quantities. You can set radiation properties for the mixture in the **Materials** details view instead, if you want. For details, see Mixture Properties Tab in the *CFX-Pre User's Guide*.

The radiation properties specified for a mixture are ignored when using the `Weighted Sum of Gray Gases`or `Multigray` spectral models.

Important:

Explicitly setting radiation properties for multicomponent flows is recommended. When a large number of components are involved, it takes a significant amount of solver CPU time to calculate these mixture properties. Often, the radiation properties are the same for all components, so computing radiation mixture properties results in wasted effort. You can avoid the radiation mixture property calculation by providing values for the mixture when it is created. An example of this is available.

For further details, see Material Properties for Radiation (p. 468).

1.4.9. Buoyancy Properties

1.4.9.1. Thermal Expansivity

Thermal Expansivity (or coefficient of thermal expansion), β, describes how a fluid expands with temperature. It is used in the Boussinesq approximation for buoyant flow and has dimensions of θ^{-1} (where θ is temperature). For details, see Buoyancy in the *CFX-Solver Theory Guide*. This parameter will be used by the flow solver only if the fluid has a constant density. Otherwise, the full buoyancy model will be used, for which thermal expansivity is not a required parameter.

1.4.10. Electromagnetic Properties

The electromagnetic properties of a material include its electrical conductivity and its magnetic permeability. You can specify values for both of these characteristics in the CFX-Pre **Material** details tab.

1.4.10.1. Electrical Conductivity

The **Electrical Conductivity** area has the following settings:

Option

Can be set to **Value**.

Electrical Conductivity

Can be set to an expression or to a value (default units are `[S m^-1]`).

1.4.10.2. Magnetic Permeability

Magnetic permeability is the equivalent of the electrical permittivity, but for the magnetic field. For permeable linear materials it can be specified from the relative permeability or from the magnetic susceptibility. The **Magnetic Permeability** area has the following settings:

Option

Can be set to **Value**.

Magnetic Permeability

Can be set to an expression or to a value (default units are `[H m^-1]`).

By default, the **Magnetic Permeability** value is set to a CEL constant called mupermo, which is defined as 4pi*1E^-7[A^-2 N]. See CEL Constants in the *CFX Reference Guide* for details on CEL constants.

1.4.11. Library Materials

CFX-Pre provides an extensive list of library materials for which the properties have already been set. These materials can be used directly in a simulation or as a template to define customized materials with similar properties. The default materials library is stored in the `MATERIALS` file located in the `CFX/etc/` directory. You can also edit/add to the `MATERIALS` file, but this is considered an expert feature. For details, see Adding to the MATERIALS file (p. 97). Exporting and Importing CCL files is a simpler way to transferring material data from one simulation to another.

The properties for many materials have been evaluated at Standard Temperature and Pressure (STP), defined as 0°C (273.15 K) and 1 atm (101325 Pa). The long name provided with most materials will indicate the condition at which constant properties were evaluated or possibly the range over which the properties apply. It is important to note that using library materials will produce inaccurate results if your simulation conditions depart significantly from those at which the material properties were evaluated.

Some pre-supplied materials have properties defined in different ways, depending on the type of modeling for which they will be used. For example, there are two material groups for "gases." In the material group drop-down list, you will see that there is a group for constant properties and a group for Ideal Gases. All the materials in the first group have both constant density and constant specific heat capacity. All materials in the second group use the Ideal Gas equation of state but have constant specific heat capacity.

In particular, Air at STP and Air Ideal Gas are two examples. In addition, a third material such as Air at 25 C is also available if Air at STP is not appropriate. The last version of `Air` uses material properties evaluated at 25 C instead of 0 C. You should select the appropriate material for your simulation; this may require you to modify the properties of an existing material or create a new material.

To view the properties of the template materials, you can select it in the tree view and then click the *Edit*  icon. The **Materials** details view will open and you will be able to see the material properties.

1.4.11.1. Adding to the MATERIALS file

If you frequently use a material that is not defined in the `MATERIALS` file, you may want to add it to the main `MATERIALS` file. Using a text editor, you can access the file

CFX/etc/MATERIALS and make any necessary revisions.

Note:

You should make a local copy of the MATERIALS file and rename it so that the backup database can be restored at a later date. Your changes should then be made to copy named MATERIALS.

To enable CFX-Pre to correctly read your new material, the format should exactly match that of existing materials. Therefore, you should use one of the existing materials as a template for entering the properties of your material.

The best way to do this is to create your new material in CFX-Pre, write the material to a file using **File** > **Export** > **CCL**, and then cut and paste the material from that file into your local copy of the materials file.

1.5. Mixture Properties (Fixed, Variable, Reacting)

By default, for fixed composition, variable composition, and reacting mixtures, all mixture properties use the Ideal Mixture option. With this option, a mixture property (or its inverse, for the cases of density and molar mass) is computed as a mass-weighted average of the component properties (or their inverses, for the cases of density and molar mass).

In order to set options other than Ideal Mixture for the **Thermodynamic Properties**, select **Mixture Properties** in the **Mixture Properties** tab in the **Material** details view and set **Option** to Non Ideal Mixture. When the global option is set to Non Ideal Mixture, individual properties may still be set to use the ideal mixture rule by selecting Ideal Mixture for the respective property.

Details of the non-ideal mixture options are described in the following subsections:

1.5.1. Equation of State/Density

1.5.2. Molar Mass

1.5.3. Specific Heat Capacity

1.5.4. Electromagnetic Properties

1.5.1. Equation of State/Density

For fixed composition, variable composition, and reacting mixtures, for which mixture density is governed by the Ideal Mixture option, the inverse of the mixture density is computed as a mass-fraction-weighted average of specific volumes:

$$\frac{Y_{\mathrm{A}}}{\rho_{\mathrm{A}}}+\frac{Y_{\mathrm{B}}}{\rho_{\mathrm{B}}}+...+\frac{Y_{N}}{\rho_{N}}=\frac{1}{\rho_{\mathrm{mix}}} \tag{1.34}$$

When the global option for **Mixture Properties** is set to Non Ideal Mixture, you can explicitly set an equation of state for the mixture in the **Mixture Properties** tab in the **Material** details view. You may choose one of two options:

- Mixture Density

  Sets density as a function of pressure, temperature, and composition.

- `Mixture Pressure`

  Sets pressure as a function of specific volume, temperature, and composition.

For either option, the partial derivatives with respect to temperature and pressure, for `Mixture Density`, or temperature and specific volume, for `Mixture Pressure`, have to be specified. CEL expressions may be used for defining the parameters as functions of the mixture composition and the corresponding properties of the mixture components. The following quantities are available for use in the expressions:

- **Density**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component densities, component specific volumes
- **d(rho)/dp at Const. T**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component densities, component specific volumes, component density derivatives with respect to pressure at constant temperature
- **d(rho)/dT at Const. p**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component densities, component specific volumes, component density derivatives with respect to temperature at constant pressure
- **Absolute Pressure**: temperature, component mass fractions, component mole fractions, component molar masses, component densities, component specific volumes
- **dp/dv at Const. Temp.**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component densities, component specific volumes, component pressure derivative with respect to volume at constant temperature
- **dp/dT at Const. Vol.**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component densities, component specific volumes, component pressure derivative with respect to temperature at constant volume

The text boxes below provide CCL examples for the `Mixture Density` and `Mixture Pressure` options, respectively, for a mixture consisting of four components: H2, O2, H2O, and N2. For simplicity the examples resemble the ideal mixture rule, where for the `Mixture Pressure` option it is additionally assumed that all components are ideal gases.

```
EQUATION OF STATE:
  Option = Mixture Density
  Density  = (H2.Mass Fraction/H2.Density + \
              O2.Mass Fraction/O2.Density + \
              H2O.Mass Fraction/H2O.Density + \
              N2.Mass Fraction/N2.Density)^-1
  drhodp t = (H2.Mass Fraction/H2.drhodp t + \
              O2.Mass Fraction/O2.drhodp t + \
              H2O.Mass Fraction/H2O.drhodp t + \
              N2.Mass Fraction/N2.drhodp t)^-1
  drhodt p = (H2.Mass Fraction/H2.drhodt p + \
              O2.Mass Fraction/O2.drhodt p + \
              H2O.Mass Fraction/H2O.drhodt p + \
              N2.Mass Fraction/N2.drhodt p)^-1
END


EQUATION OF STATE:
  Option = Mixture Pressure
  Absolute Pressure = (H2.Mass Fraction/H2.Molar Mass + \
                       O2.Mass Fraction/O2.Molar Mass + \
                       H2O.Mass Fraction/H2O.Molar Mass + \
                       N2.Mass Fraction/N2.Molar Mass) * \
```

```
                      R*T / Specific Volume
  dpdv t = - (H2.Mass Fraction/H2.Molar Mass + \
              O2.Mass Fraction/O2.Molar Mass + \
              H2O.Mass Fraction/H2O.Molar Mass + \
              N2.Mass Fraction/N2.Molar Mass) * \
             R*T / (Specific Volume^2)
  dpdt v = (H2.Mass Fraction/H2.Molar Mass + \
            O2.Mass Fraction/O2.Molar Mass + \
            H2O.Mass Fraction/H2O.Molar Mass + \
            N2.Mass Fraction/N2.Molar Mass) * \
           R / Specific Volume
END
```

It is your responsibility to ensure that the partial derivatives of density or pressure are consistent with the specific heat capacity definition. For details on consistency requirements, see General Equation of State in the *CFX-Solver Theory Guide*.

1.5.2. Molar Mass

For fixed composition, variable composition, and reacting mixtures, for which mixture molar mass is governed by the `Ideal Mixture` option, the inverse of the molar mass is computed as a mass-fraction-weighted average of the inverses of the component molar masses:

$$\frac{Y_{\mathrm{A}}}{W_{\mathrm{A}}}+\frac{Y_{\mathrm{B}}}{W_{\mathrm{B}}}+\ldots+\frac{Y_{N}}{W_{N}}=\frac{1}{W_{\mathrm{mix}}} \tag{1.35}$$

You can set a **Molar Mass** for the mixture in the **Material** details view instead, if you want.

1.5.3. Specific Heat Capacity

When the global option for **Mixture Properties** is set to `Non Ideal Mixture`, you can set the specific heat capacity for the mixture. The **Specific Heat Type** can be either `Constant Pressure` or `Constant Volume`. In addition to specific heat capacity, you also need to specify the specific enthalpy and the specific entropy.

CEL expressions may be used for defining the three parameters as functions of the mixture composition and the corresponding properties of the mixture components. The following quantities are available for use in the expressions:

- **Specific Heat Capacity**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component specific heat capacities
- **Specific Enthalpy**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component specific enthalpies
- **Specific Entropy**: pressure, temperature, component mass fractions, component mole fractions, component molar masses, component specific entropies

As an example, the text box below shows the CCL commands for defining a specific heat capacity for a mixture consisting of four components: H2, O2, H2O and N2. For demonstration purposes, the example resembles the ideal mixture rule.

```
SPECIFIC HEAT CAPACITY:
  Option = Mixture Specific Heat Capacity
  Specific Heat Type = Constant Pressure
  Specific Heat Capacity = H2.Mass Fraction*H2.Cp + \
                           O2.Mass Fraction*O2.Cp + \
```

```
                          H2O.Mass Fraction*H2O.Cp + \
                          N2.Mass Fraction*N2.Cp
  Specific Enthalpy = H2.Mass Fraction*H2.Static Enthalpy + \
                      O2.Mass Fraction*O2.Static Enthalpy + \
                      H2O.Mass Fraction*H2O.Static Enthalpy + \
                      N2.Mass Fraction*N2.Static Enthalpy
  Specific Entropy  = H2.Mass Fraction*H2.Static Entropy + \
                      O2.Mass Fraction*O2.Static Entropy + \
                      H2O.Mass Fraction*H2O.Static Entropy + \
                      N2.Mass Fraction*N2.Static Entropy
END
```

1.5.4. Electromagnetic Properties

The default mixture (Ideal Mixture) is also used for electromagnetic properties. The extensions are for Electrical Conductivity (no default) and Magnetic Permeability (the permeability of free space).

1.6. Efficiency Calculation

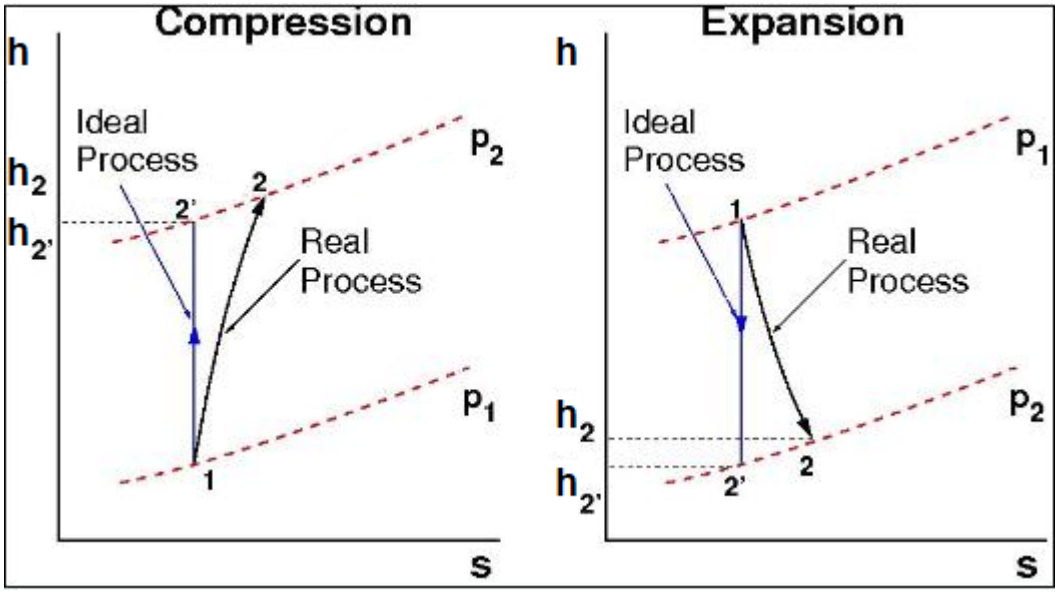
The following topics are discussed in this section:

- Isentropic Efficiency and Total Enthalpy (p. 101)
- Polytropic Efficiency (p. 103)
- Activating Efficiency Output (p. 105)
- Restrictions (p. 106)

1.6.1. Isentropic Efficiency and Total Enthalpy

The efficiency of any device is never perfect due to both fluid dynamic and thermodynamic losses within the entire system. The losses show up in a difference in stagnation properties from the values that would be expected if the process would be perfectly isentropic. Consider the processes drawn on the T-s diagrams shown in Figure 1.2: T-s Diagrams for Compression and Expansion Processes (p. 102) for a compression and expansion process.

Figure 1.2: T-s Diagrams for Compression and Expansion Processes



Two efficiency calculation options are possible for either process: **Total to Total** and **Total to Static**. The first option evaluates efficiency using stagnation conditions at points 1, 2 and 2′. The second option uses stagnation conditions at points 1 and 2 and static conditions at point 2′, so, the efficiency is evaluated without including the kinetic energy contribution at point 2. The **Total to Total** option is most useful for devices where work is either performed on the fluid (for example, a pump or compressor) or extracted from the fluid (for example, a turbine). The **Total to Static** option is useful for evaluating losses in devices that perform no work on the fluid (for example, diffusers and nozzles).

The isentropic work is of the form:

$$h_{\text{isen}} = h_{02'} - h_{01} \tag{1.36}$$

Work is generally input for a compression process such as gas compressors, so the **Total to Total** isentropic efficiency is defined as the ratio of the required isentropic work to the actual required work for fluid compression:

$$\eta_C = \frac{h_{\text{isen}}}{h_{02} - h_{01}} \tag{1.37}$$

Work is usually derived from expansion processes such as steam turbines, so the **Total to Total** efficiency is the ratio of the actual expansion work to the isentropic expansion work:

$$\eta_E = \frac{h_{02} - h_{01}}{h_{\text{isen}}} \tag{1.38}$$

where the total enthalpy at point 1 is the mass average value of the user-selected inlet boundary condition.

In addition, the isentropic total enthalpy, $h_{02'}$ is required. This is the fluid enthalpy evaluated at the entropy at point 1 and the total pressure at point 2. The entropy at point 1 is taken, by default, as the mass averaged value at a user-selected inlet boundary condition. The isentropic enthalpy is evaluated from the state equation:

$$h_{02',\text{node}} = f(\overline{s}_{\text{inlet}}, p_{0,\text{node}}) \tag{1.39}$$

The isentropic enthalpy is then used in Equation 1.37 (p. 102) or Equation 1.38 (p. 102) to evaluate compression or expansion efficiency. These quantities are useful for evaluating where losses are occurring in the device. Global efficiencies reported in the solver monitor are evaluated by mass averaging the field efficiencies at the user-selected outlet boundary condition.

The **Total to Static** efficiency is evaluated similarly to **Total to Total** efficiency. The **Total to Static** isentropic efficiency for a compression process is:

$$\eta_{\mathrm{C}}=\frac{h_{2'}-h_{01}}{h_{02}-h_{01}} \tag{1.40}$$

where

$$h_{2',\mathrm{node}}=f\left(\overline{s}_{\mathrm{inlet}},p_{\mathrm{node}}\right)$$

The **Total to Static** isentropic efficiency for an expansion process is:

$$\eta_{\mathrm{E}}=\frac{h_{02}-h_{01}}{h_{2'}-h_{01}} \tag{1.41}$$

Free stream detection is also applied to the efficiency values in a similar way to the high resolution advection blending. This is because in the free stream the isentropic enthalpy change and the local enthalpy change can both be small (that is, O(0)). A small number divided by another small number can result in a noisy looking efficiency field in inlet regions where the efficiency should be unity because no work has yet been performed on the fluid. Free stream damping blends the efficiency field to unity. The free stream damping is controlled with the expert parameter `fsdamp efficiency scale` with a default value of 0.01. Smaller values for this parameter will decrease the influence of the free stream damping, and reducing it to a value of 0.0 will switch off the free stream damping completely.

Device efficiency values reported by the CFX-Solver Manager and CFD-Post macros are calculated using a boundary averaged value of the damped efficiency field variable (for example Isentropic Efficiency). In some cases, this value may be different to one calculated from the classical formula using boundary-averaged values of enthalpy. The difference is due to the field damping, and is negligible in the vast majority of cases. However, in rare cases, for example where there is localized heating or cooling, the difference may be significant and you should check the reported values with the damping disabled.

1.6.2. Polytropic Efficiency

The isentropic efficiency has one significant drawback in that there is no way to separate the fluid dynamic losses from the total (fluid dynamic + thermodynamic) losses. This means that devices having different pressure ratios will have different isentropic efficiencies even though they may both be of similar fluid dynamic quality. An example would be two compressors of different pressure ratios. The higher pressure ratio compressor will have a lower isentropic efficiency because of thermodynamic losses. This property can make it difficult to comparatively evaluate different compressor designs. A similar argument applies to turbine design.

To work around this drawback, the assumed "ideal" path followed by the process does not have to be isentropic. Instead, one can evaluate the polytropic efficiency by following a path along a line of constant efficiency. Aungier [65] discusses how a constant efficiency path on a T-s diagram can be approximated by the following equation:

$$T\frac{ds}{dT}=A \tag{1.42}$$

This equation can be rearranged to give the following two relationships:

$$T\,ds=A\,dT, ds=\frac{A\,dT}{T} \tag{1.43}$$

The entropy change along this path is evaluated by integrating the last expression:

$$s_2-s_1=A\ln\left(T_{02}/\,T_{01}\right) \tag{1.44}$$

Rearranging this expression gives the value of the constant A along the given path:

$$A=\frac{s_2-s_1}{\ln\left(T_{02}/\,T_{01}\right)} \tag{1.45}$$

In order to evaluate the polytropic efficiency, evaluate the polytropic enthalpy change along the alternative path defined by the path equation. First, start with the second law:

$$T\,ds=dh-v\,dp \tag{1.46}$$

The $v\,dp$ term is the polytropic work, or enthalpy change, and is what must be solved. First, the $T\,ds$ term is substituted using Equation 1.43 (p. 104) to give:

$$A\,dT=T\,ds=dh-v\,dp \tag{1.47}$$

Integrating this equation along the polytropic path gives:

$$h_{\text{poly}}=(h_{02}-h_{01})-A(T_{02}-T_{01}) \tag{1.48}$$

Note:

The $v\,dp$ term has been renamed to h_{poly} to signify that the work is the polytropic enthalpy change.

The final form of the polytropic work is obtained by simply substituting for A:

$$h_{\text{poly}}=(h_{02}-h_{01})-\frac{(s_2-s_1)(T_{02}-T_{01})}{ln(T_{02}/\,T_{01})} \tag{1.49}$$

Now that you have an expression for this enthalpy change, you can compute the polytropic efficiency:

$$\eta_{\text{CP}}=\frac{h_{\text{poly}}}{h_{02}-h_{01}} \tag{1.50}$$

$$\eta_{\text{EP}}=\frac{h_{02}-h_{01}}{h_{\text{poly}}} \tag{1.51}$$

The polytropic efficiencies are also evaluated relative to a selected inlet boundary condition and output by the flow solver as both local (at every node in the flow) and global (overall device) quantities.

Similar to the isentropic efficiency, total to total and total to static versions of the polytropic efficiency are similarly evaluated by replacing stagnation quantities at point 2 with static quantities.

1.6.3. Activating Efficiency Output

Two options are available for efficiency analysis. You can monitor the overall device efficiency within the CFX-Solver Manager as well as post process field efficiencies in CFD-Post.

Efficiency output can be activated on the **Monitor** tab under **Output Control** in CFX-Pre, and enables you to monitor the device efficiencies in the CFX-Solver Manager.

1.6.3.1. CFX-Solver Manager Output Variables

To calculate the overall device efficiency requires that you specify both an inlet and outlet boundary condition corresponding to points 1 and 2 on the T-s diagrams shown in Figure 1.2: T-s Diagrams for Compression and Expansion Processes (p. 102). Only single boundary condition regions of type inlet and outlet are currently supported. The inlet entropy and total enthalpy are based on mass flow average values. The monitored efficiency values are based on mass average field efficiency at the selected outlet boundary condition.

For single phase cases or multiphase cases running the homogeneous model, the following variables are written to the CFX-Solver Output file:

- `Isentropic Compression Efficiency`
- `Isentropic Expansion Efficiency`
- `Polytropic Compression Efficiency`
- `Polytropic Expansion Efficiency`

For multiphase cases running the homogeneous mass and momentum model, but inhomogeneous energy equations (such as condensing/evaporation disperse droplets, disperse solids):

- <Continuous Phase>.`Isentropic Compression Efficiency`
- <Continuous Phase>.`Isentropic Expansion Efficiency`
- <Continuous Phase>.`Polytropic Compression Efficiency`
- <Continuous Phase>.`Polytropic Expansion Efficiency`

Compression and expansion efficiencies are calculated only for the continuous phase; all other dispersed phases are ignored.

For multiphase cases running the inhomogeneous mass and momentum model and inhomogeneous energy equations:

- <Continuous Phase *n*>.`Isentropic Compression Efficiency`
- <Continuous Phase *n*>.`Isentropic Expansion Efficiency`
- <Continuous Phase *n*>.`Polytropic Compression Efficiency`
- <Continuous Phase *n*>.`Polytropic Expansion Efficiency`

For each continuous phase, compression and expansion efficiencies are calculated.

All of the output variables are either **Total to Total** or **Total to Static** depending on which mode you have selected in CFX-Pre.

1.6.3.2. Results File Output Variables

When efficiency monitoring is activated field efficiencies are also written to the final results file, Standard backup and transient results files, and Selected Variables transient files where required. When efficiency monitoring is activated, the field variables can be used in CEL expressions, monitor points (callback and point monitors), and User Fortran.

In addition to the global efficiency fields, described in the previous section, the following field variables are written to results and backup files during a simulation:

For single phase or multiphase homogeneous mass and momentum with homogeneous energy equation model:

- `Isentropic Enthalpy`
- `Polytropic Enthalpy`

For homogeneous mass and momentum with inhomogeneous energy equation:

- <Continuous Phase>.`Isentropic Enthalpy`
- <Continuous Phase>.`Polytropic Enthalpy`

For inhomogeneous mass and momentum with inhomogeneous energy equation:

- <Continuous Phase *n*>.`Isentropic Enthalpy`
- <Continuous Phase *n*>.`Polytropic Enthalpy`.

All of the output variables are either Total to Total or Total to Static depending on which mode you have selected in CFX-Pre.

1.6.4. Restrictions

The following restrictions apply and are enforced by the CFX-Solver and CFX-Pre:

- The efficiency calculation is possible only if you use a non-isothermal heat transfer model with compressible fluids. For devices that involve a very small enthalpy change (or temperature change), ensuring that your material definition uses a small reference enthalpy might help to avoid inaccuracies.
- The boundary lists enable only single boundary conditions of type **Inlet** or **Outlet**.
- Multiple component efficiencies cannot be directly monitored using the existing interface. The best approach is to activate the field output and use the expression monitor option to monitor the efficiency change across the given component.
- **Total to Total** and **Total to Static** efficiency values cannot be evaluated at the same time. You must select one mode or the other.

1.7. Wall Condensation Model

The Wall Condensation Model has been developed for use in the nuclear containment industry, although it can be used in more general situations. In this model, the condensation of gaseous vapor in a variable composition mixture is driven by the concentration gradient at the wall, and internal boundary mass sources are used to model the removal of condensate from the system. The condensate film thickness, and heat transfer through it, are assumed to be negligible and are not modeled explicitly. This is appropriate for containment scenarios where the condensate does not adhere to the surface but is absorbed by it (for example, on ceramic/concrete walls).

The wall condensation model has two formulations: one for laminar flow and one for turbulent flow. The formulation for laminar flow has been developed for a specific case; it has a hard-coded saturation temperature threshold (623 in solver units of temperature: Kelvin by default) and requires a fine, uniform, and orthogonal, mesh near the wall (such as an inflation layer).

Note:

For laminar flow cases, you should avoid using the laminar flow formulation of the wall condensation model by modeling as a turbulent case if possible.

Detailed information about the Wall Condensation Model is available in Wall Condensation Theory in the *CFX-Solver Theory Guide*.

The following topics are discussed.

1.7.1. CFX-Pre Set-up

1.7.2. Condensation Mass Flux in CFD-Post

1.7.1. CFX-Pre Set-up

To set up a case, you require a variable composition mixture, the condensable component of which must be part of a Homogeneous Binary Mixture (HBM). The latter is not required to be a fluid in the domain, but must be part of the case definition. The variable composition mixture must consist of one condensable component and one or more non-condensable components.

Note:

There is a template model file (`wall_condensation.ccl`) to aid the set up of cases that use the wall condensation model.

The following topics are discussed:

1.7.1.1. Domain Fluid Model Specification

1.7.1.2. Boundary Condition Specification

1.7.1.3. Restrictions

1.7.1.4. Convergence Tip

1.7.1.1. Domain Fluid Model Specification

To use the wall condensation model, visit the Fluid Models tab of the domain, and set **Component Models** > **Wall Condensation Model** > **Option** to `Concentration Boundary Layer Model`.

Note that the wall condensation model is allowed only for multi-component fluids defined on fluid domains on which the Thermal Energy or Total Energy model is active. Also, there must exist a unique condensable component. It is necessary to indicate which component is condensable, and which are non-condensable. For the condensable component, it is necessary to name an associated homogeneous binary mixture that will be used to compute the required saturation material properties. The named homogeneous binary mixture must exist in the materials library, and must contain the condensable component as one of its components, and the condensate as the other component.

Note that the condensation model settings can be applied to only components that have the `Transport Equation` or `Algebraic Slip` option. In most applications, the condensable component will be vapor and the condensate will be liquid. However, the model is valid in other contexts, such as sublimation, so all thermodynamic states are permissible for the condensable component and the condensate.

1.7.1.2. Boundary Condition Specification

Wall condensation must be explicitly activated on those wall or fluid-solid boundaries for which it is required. The default condition on these boundaries, for all components, is Zero Flux.

Note that there are slight differences in the reporting of heat flux imbalances at a condensing fluid-solid interface, depending on whether the interface is one-to-one periodic or GGI. For a one-to-one periodic interface, condensation heat fluxes are computed internally in the same way that user boundary sources are computed, so the condensation heat fluxes are reported as boundary sources associated with the boundary condition. For a GGI interface, condensation heat fluxes are added to the volumetric heat sources, so the condensation heat fluxes appear as part of the heat source contribution from the domain.

1.7.1.3. Restrictions

The wall condensation model has the following restrictions:

- The wall condensation model is allowed only for multi-component fluids defined on fluid domains on which the Thermal Energy or Total Energy model is active.
- Only one condensable component is permitted. There is no restriction on the number of non-condensable components. The condensable component should either have Option = Transport Equation or Option = Algebraic Slip.
- Condensation boundary conditions are restricted to walls with specified temperature, and fluid-solid interfaces. The following further restrictions are imposed:
  - You cannot specify wall condensation on non-overlap boundaries.
  - Free slip and specified stress walls are allowed for laminar flows but are disallowed for turbulent flows.

- It is not possible to specify both wall condensation boundary conditions and additional user boundary mass sources on a given boundary surface.

1.7.1.4. Convergence Tip

At CHT boundaries adjacent to a low conductivity solid, convergence can be very slow or difficult to obtain. This can be helped significantly by applying explicit under-relaxation to both the condensation mass flux and condensation heat flux at the interface. This can be achieved by the following CCL:

```
LIBRARY:
  VARIABLE: condmflux
     Option = Definition
     Under Relaxation Factor = <real>
  END
  VARIABLE: condhflux
     Option = Definition
     Under Relaxation Factor = <real>
  END
END
```

Under-relaxation values between 0.01 and 0.1 are recommended.

1.7.2. Condensation Mass Flux in CFD-Post

On any boundary surface where condensation boundary conditions are set, the condensation mass flux is written as a boundary-only variable to CFD-Post under the variable name <Component>.Condensation Mass Flux.

Chapter 2: Boundary Condition Modeling

This chapter provides detail on how to use appropriate boundary conditions in CFX for basic models. Information on the mathematical representation of CFX boundary conditions is available in Boundary Conditions in the *CFX-Solver Theory Guide*. Extended information for more complex models is provided in the related modeling chapters. For example, for information on multiphase outlets, you should consult the multiphase modeling section for specific advice. For details, see Boundary Conditions in Multiphase Flow (p. 340).

The equations relating to fluid flow can be closed (numerically) by the specification of conditions on the external boundaries of a domain. It is the boundary conditions that produce different solutions for a given geometry and set of physical models. Hence, boundary conditions determine to a large extent the characteristics of the solution you obtain. Therefore, it is important to set boundary conditions that accurately reflect the real situation to enable you to obtain accurate results.

This chapter describes:

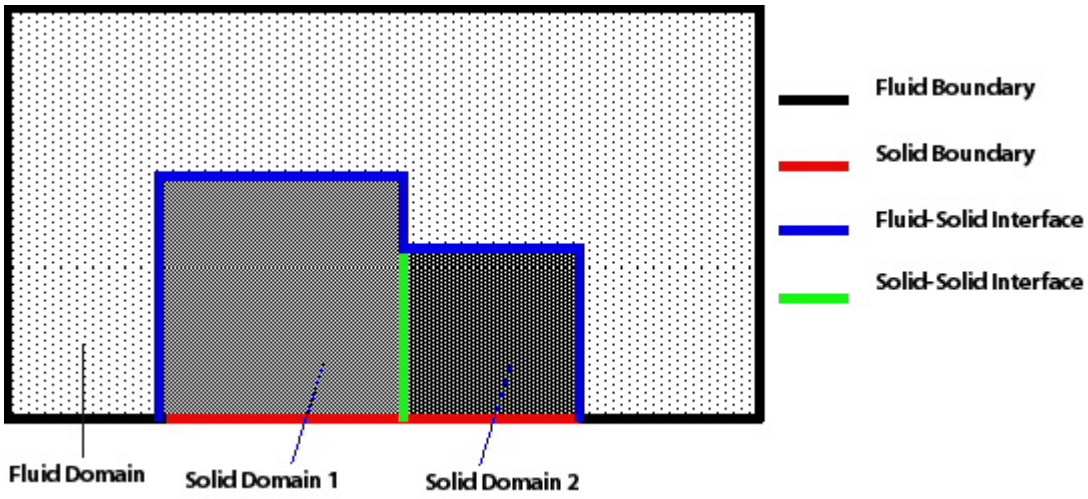
2.1. The Purpose of Boundary Conditions
2.2. Available Boundary Conditions
2.3. Using Boundary Conditions
2.4. Inlet
2.5. Outlet
2.6. Opening
2.7. Wall
2.8. Symmetry Plane
2.9. Profile Boundary Conditions
2.10. General Non-Reflecting Boundary Conditions
2.11. Limitations

2.1. The Purpose of Boundary Conditions

Boundary conditions are a set of properties or conditions on surfaces of domains, and are required to fully define the flow simulation. The type of boundary condition that can be set depends upon the bounding surface.

- A fluid boundary is an external surface of the fluid domain excluding surfaces where it meets other domains.
- A solid boundary is an external surface of the solid domain excluding surfaces where it meets other domains.
- A fluid-fluid interface is the interface between two fluid domains.

- A fluid-solid interface is the interface between a solid and fluid domain.
- A solid-solid interface is the interface between two solid domains.

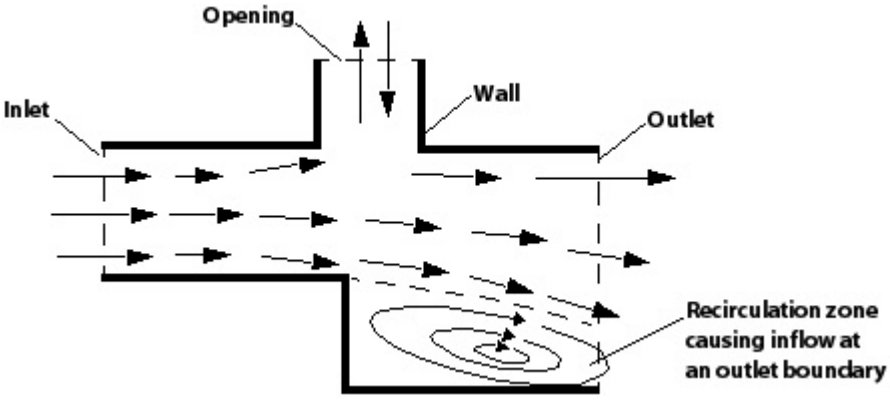


2.2. Available Boundary Conditions

The following boundary types are available in Ansys CFX:

- Fluid Boundaries (p. 112)
- Solid Boundaries (p. 113)

2.2.1. Fluid Boundaries



A fluid boundary is an external surface of a fluid domain and supports following boundary conditions:

- `Inlet` - Fluid predominantly* flows into the domain.
- `Outlet` - Fluid predominantly* flows out of the domain.

- `Opening` - Fluid can simultaneously flow both in and out of the domain. This is not available for domain with more than one fluid present.
- `Wall` - Impenetrable boundary to fluid flow.
- `Symmetry Plane` - A plane of both geometric and flow symmetry.

* When you define an area to be an "inlet," you are telling CFX-Solver that you anticipate that flow will be into a domain. You can set either velocity or pressure at that point, and CFX-Solver will calculate the other value. How CFX-Solver behaves during the flow calculations depends on which attribute you set:

- CFX-Solver allows both inflow and outflow due to the velocity specified conditions: "artificial walls" are not erected at inlets or inlets due to such conditions.
- If you set pressure at an inlet, you may accidentally create conditions that would lead to an outflow when the pressure inside the domain is greater than outside. Under this condition, CFX-Solver "builds" an artificial wall to prevent outflow. To disable this behavior, define the boundary to be an "opening," rather than an inlet.

For details, see Using Inlets, Outlets and Openings (p. 114).

2.2.2. Solid Boundaries

A solid boundary is an external surface of the solid domain and supports the following boundary conditions:

- `Wall` - Impenetrable boundary to fluid flow.
- `Symmetry Plane` - A plane of both geometric and flow symmetry.

2.3. Using Boundary Conditions

The following topics will be discussed:

2.3.1. Specifying Well-Posed Boundary Conditions
2.3.2. Recommended Configurations of Boundary Conditions
2.3.3. Using Inlets, Outlets and Openings
2.3.4. Using CEL Expressions With Boundary Conditions

2.3.1. Specifying Well-Posed Boundary Conditions

For a given computational domain, boundary conditions can be given that over-specify or under-specify the problem. This usually results in non-physical solutions or failure of the solution to converge. It is important, therefore, to understand the meaning of well-posed boundary conditions.

An example of an over-specified problem is a constant area duct, with a specified fluid velocity at the inlet, and a different velocity specified at the outlet. Clearly, both conditions cannot be physically satisfied in the absence of a mass source for an isochoric fluid.

An example of an under-specified problem is that of a closed box for which only heat flux boundary conditions were specified. In this case, the temperature level is not constrained and, while the solution may converge, the resultant temperature level would be unpredictable.

The best way to determine if the boundary conditions are well posed is to ask the question "Could the configuration I have set be physically recreated in a laboratory?" In the first example above, this is clearly not possible. In the second example, no matter how good the insulation of the boundary, there would eventually be some heat flow from or into the environment that would serve to set the temperature level.

For details, see Boundary Conditions in Multiphase Flow (p. 340).

2.3.2. Recommended Configurations of Boundary Conditions

The following combinations of boundary conditions are all valid configurations commonly used in Ansys CFX. They are listed from the most robust option to the least robust:

- **Most Robust**: Velocity/mass flow at an inlet and static pressure at an outlet. The inlet total pressure is an implicit result of the prediction.
- **Robust**: Total pressure at an inlet and velocity/mass flow at an outlet. The static pressure at the outlet and the velocity at the inlet are part of the solution.
- **Sensitive to Initial Guess**: Total pressure at an inlet and static pressure at an outlet. The system mass flow is part of the solution.
- **Very Unreliable**: Static pressure at an inlet and static pressure at an outlet. This combination is not recommended, as the inlet total pressure level and the mass flow are both an implicit result of the prediction (the boundary condition combination is a very weak constraint on the system).
- **Not Possible**: Total pressure cannot be specified at an outlet. The total pressure boundary condition is unconditionally unstable when the fluid flows out of the domain where the total pressure is specified.

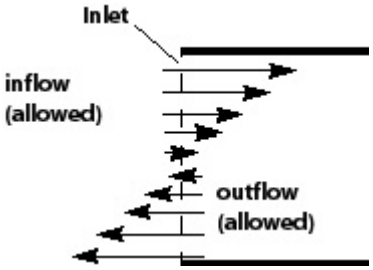
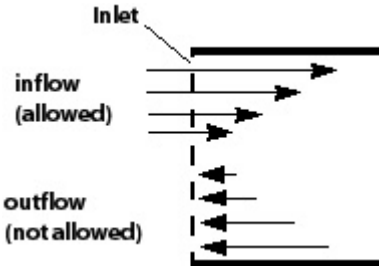
With more than two inflows or outflows, the other openings should be of the boundary condition type `Opening`. This is because the flow at these other boundaries could in general be in or out, and the direction will be part of the solution.

2.3.3. Using Inlets, Outlets and Openings

Define regions as inlets and outlets when you expect one-way flow boundary conditions; define them as openings when you want to support simultaneous inflow and outflow over a single region. Care should be taken specifying velocity components for these boundary conditions as you are allowed to specify flow in either direction at an inlet or outlet.

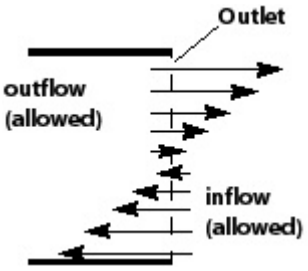
2.3.3.1. Inlets

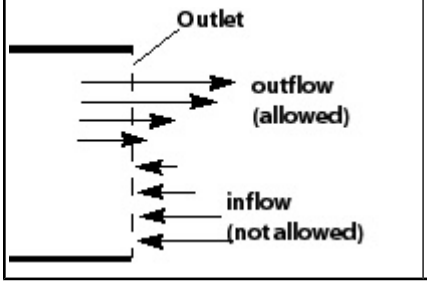
Inlets are used predominantly for regions where inflow is expected; however, inlets also support outflow as a result of velocity specified boundary conditions.

Artificial walls for Inlets	
Velocity specified condition 	CFX-Solver allows both inflow and outflow, and therefore artificial walls are not erected when the specified velocity is out of the domain.
Pressure and mass flow condition 	CFX-Solver enforces inflow by erecting artificial walls on those faces where the flow attempts to exit the domain.

2.3.3.2. Outlets

Outlets are used predominantly for regions where outflow is expected; however, outlets also support inflow as a result of velocity specified boundary conditions.

Artificial walls for Outlets	
Velocity specified condition 	CFX-Solver allows both inflow and outflow, and therefore artificial walls are not erected at inlets when the specified velocity is into the domain.
Pressure and mass flow condition	CFX-Solver enforces outflow by erecting artificial walls on those faces where the flow attempts to enter the domain.

Artificial walls for Outlets	
	

2.3.3.3. Openings

Define a region as an "opening" when the full prescription of information at that location is not readily available; for example, if the value of pressure is known, but the direction of flow is unknown. It is also important to note how Ansys CFX treats the value of pressure that you specify at an opening. If, during the course of the solution, flow is directed out of the domain, then the value is treated as static pressure. If, instead, the flow is entering the domain, the value is taken to be total pressure, from which the static pressure is calculated.

A large number of flow simulations can make use of the `Static Pressure` option to describe an outlet boundary where flow is out of the domain, and the `Total Pressure` option to specify an inlet boundary where flow is into the domain. As the solution progresses, inflow may occur at an outlet boundary, or outflow at an inlet boundary, due to either:

- The behavior of the solver
- A physical phenomenon, such as a recirculation zone, close to the boundary.

In either case, the CFX-Solver will try to enforce outflow by erecting artificial walls on the boundary mesh faces to prevent inflow occurring. If the full inlet/outlet is walled off and it was the only boundary condition that set the pressure level, then the artificial walls create a serious problem: temporarily, there is no pressure level felt by the code. This usually causes a linear solver failure. If you check the CFX-Solver Output file, a message like the following should be present:

```
+--------------------------------------------------------------------+
|                     ****** Notice ******                           |
|  A wall has been placed at portion(s) of an OUTLET                 |
|  boundary condition (at   31.4%  of the faces, 12.3% of the area)  |
|  to prevent fluid from flowing into the domain.                    |
|  The boundary condition name is: exit.                             |
|  The fluid name is: AirIdeal.                                      |
|  If this situation persists, consider switching                    |
|  to an Opening type boundary condition instead.                    |
+--------------------------------------------------------------------+
```

If you suspect that the problem is a numerical one, then you overcome this by temporarily replacing the inlet/outlet boundary condition with a pressure-specified opening. Artificial walls are not erected with the opening type boundary, as both inflow and outflow are allowed (although you may have to specify information that is used if the flow becomes locally inflow). Restart the solver, and once the solution is further converged, the original boundary condition can be restored.

If you suspect that a real phenomenon is causing the inflow, for example, a recirculation region near an outlet, then two options are available to you:

- Move the outlet to a region well away from this influence by, perhaps, extending the flow domain further downstream. A good rule of thumb is to place the boundary a distance downstream that is at least 10 times the height of the last obstacle.
- Alternatively, you may have reasons to want to maintain the location of that boundary, in which case, you can use an opening to describe subsonic flow where simultaneous inflow and outflow may occur. Generally, however, it should be noted that defining the boundary of your computational domain to cross a flow recirculation region is not a good idea. You are making a poor approximation of what is happening just outside the opening by not modeling it.

The diagram below, showing flow over a backward facing step, serves to illustrate the former approach. A recirculation region builds up downstream of the step as a result of the separated flow. An initial pressure boundary intersects this region with the result that flow re-enters the domain across it. Convergence problems will arise in this situation because the CFX-Solver will try to enforce an outward-flow-only condition, which is inconsistent with the physical nature of the flow. Although placing the boundary further downstream (beyond the re-attachment point) alleviates the inflow problem, the flow is still developing in the enlarged duct. A much better location would be at a point where the flow profile is not changing, well away from any zone of influence.

If, during the course of the solution, you do still experience convergence problems, the use of a small physical timestep may help to promote convergence.

Figure 2.1: Poor Location

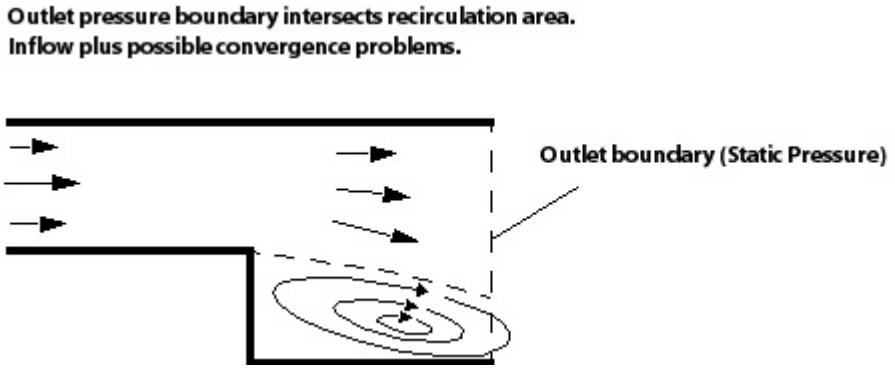


Figure 2.2: Better Location

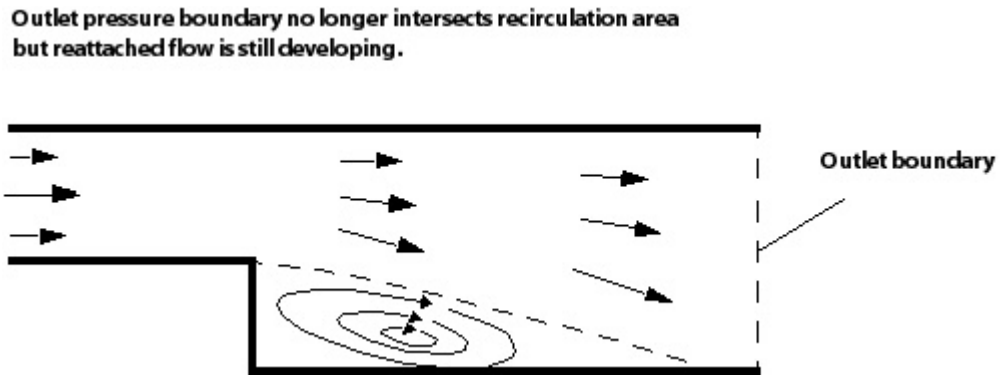
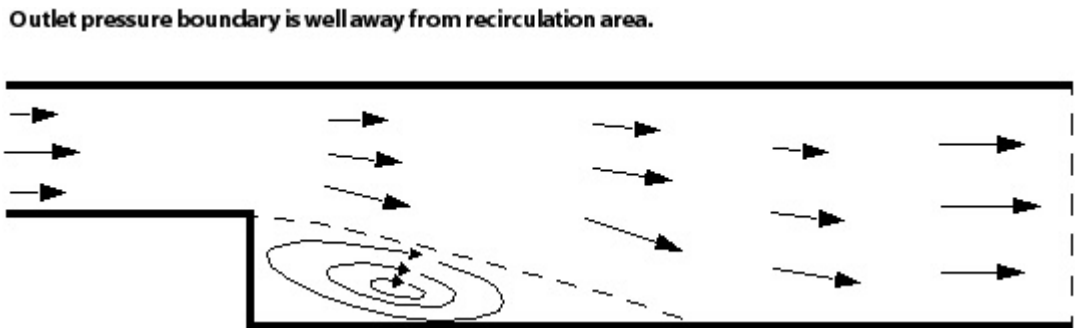


Figure 2.3: Ideal Location



2.3.3.4. Using Pressure Specified Boundaries with Buoyant Flows

When buoyancy is activated, the pressure calculated by the solver excludes the hydrostatic pressure gradient. This modified pressure is often called motion pressure because it is responsible for driving the flow. All boundary conditions are interpreted in terms of this modified pressure. For details, see Buoyancy and Pressure (p. 60).

Furthermore, if there is a free surface flow at the boundary due to variations in density, temperature or height change; specifying a pressure profile at the boundary may be required.

2.3.4. Using CEL Expressions With Boundary Conditions

Boundary condition values can be defined using any valid CEL expressions.

For details, see CEL Operators, Constants, and Expressions in the *CFX Reference Guide*.

2.4. Inlet

An inlet boundary condition is used where the flow is predominantly directed into the domain. The boundary condition can be set in a number of ways depending on how you want to specify the conditions, and what particular physical models you are using for the simulation.

There are many different types of inlet boundary condition combinations for the mass and momentum equations. For all other transport equations, the value is specified directly at the inlet, or specified in terms of a simple relationship that constrains the dependent variable.

Important:

If you accidentally set the velocity at an inlet to be flowing out of the domain, the inlet will behave as an outlet: *wall building will not be turned on*.

Information on setting Inlet values using CFX-Pre is available. For details, see Boundary Details: Inlet in the *CFX-Pre User's Guide*.

Information on the mathematical implementation of the inlet boundary condition is available.

- Inlet (Subsonic) in the *CFX-Solver Theory Guide*

- Inlet (Supersonic) in the *CFX-Solver Theory Guide*

2.4.1. Mesh Motion

Mesh motion can occur on inlet boundaries in domains where **Mesh Deformation** has been activated for the domain. See Mesh Deformation in the *CFX-Pre User's Guide* for information about activating mesh deformation for the domain. Note that solution errors will be introduced if any of the specified motion is normal to the boundary faces. For details, see Mesh Deformation (p. 42).

2.4.2. Inlet (Subsonic)

2.4.2.1. Mass and Momentum

Note:

Acoustic Reflectivity settings are available for subsonic inlet boundaries that are based on:

- Static Pressure
- Total Pressure
- Normal Speed / Velocity Components
- Mass Flow

For details, see General Non-Reflecting Boundary Conditions (p. 155).

2.4.2.1.1. Normal Speed

Specify the magnitude of the resultant normal velocity at the boundary. The value you specify is transferred from the fluid domain normal to each element face on that boundary during the execution of the CFX-Solver.

This option is especially useful for non-planar inlet boundaries (that is, curved surfaces).

If the domain is porous, the speed that you specify is interpreted as true, not superficial.

2.4.2.1.2. Cartesian Velocity Components

Specify the Cartesian components of velocity on the inlet boundary. The component values are relative to the selected coordinate frame.

If the domain containing the boundary condition is rotating, then the axis of the cylindrical co-ordinate system will automatically be set to the rotation axis. You can specify either relative or absolute velocity components by choosing **Frame Type**:

- If the **Frame Type** is `Rotating`, then your velocity components are relative to the local domain rotating frame of reference.

- If the **Frame Type** is `Stationary`, then your velocity components are relative to the absolute (stationary) frame of reference (at its initial orientation for transient cases).

If the domain is porous, the velocity components that you specify are interpreted as true, not superficial.

2.4.2.1.3. Cylindrical Velocity Components

Specify the r, theta, z components of velocity on the inlet boundary in cylindrical coordinates. The axis about which the velocity components are specified depends on the domain motion (rotating or stationary) and on whether a local rotation axis is specified with the boundary condition.

If the domain containing the boundary condition is stationary, you must specify an axis in one of two ways:

- You can select the axis of the cylindrical coordinate system for the boundary condition (called the **Rotation Axis**) as an axis of the global coordinate frame, Coord 0.
- You can specify **Rotation Axis From** and **Rotation Axis To** points. These are the two end points of the axis. The global coordinate system must be used to specify these points. The positive z component of the cylindrical coordinates is in the direction of the vector from the **Rotation Axis From** point to the **Rotation Axis To** point.

If the domain containing the boundary condition is rotating, then the axis of the cylindrical coordinate system will automatically be set to the rotation axis. You can specify either relative or absolute velocity components by choosing **Frame Type**:

- If the **Frame Type** is `Rotating`, then your velocity components are relative to the local domain rotating frame of reference.
- If the **Frame Type** is `Stationary`, then your velocity components are relative to the absolute (stationary) frame of reference (at its initial orientation for transient cases).

If the domain is porous, the velocity components that you specify are interpreted as true, not superficial.

2.4.2.1.4. Mass Flow Rate

When **Mass and Momentum** > **Option** is set to `Mass Flow Rate`, you must set **Mass and Momentum** > **Mass Flow Rate**. A positive value represents mass flow through the boundary in the specified flow direction. For details, see Flow Direction (p. 122).

You must also choose one of two settings for **Mass Flow Rate Area**:

- `As Specified`

  Mass Flow Rate corresponds to the modeled sector, and is applied directly to the boundary. This setting is selected by default.

- `Total for All Sectors`

  This setting is available only for single phase cases that have rotational periodicity and that have a rotational periodic interface. If no rotational periodic interface is present, `Total for`

`All Sectors` is identical to `As Specified`. For details on interface models with rotational periodicity, see Interface Models (p. 234).

Mass Flow Rate corresponds to the full geometry. The pair of rotational periodic interfaces (or, in the case of multiple pairs existing in the domain, one such pair) is used to calculate the sector angle (the angle that the modeled sector spans in the full geometry). **Mass Flow Rate** is multiplied by the ratio between the sector angle and 360° before being applied to the boundary.

Note:

Only one sector angle is calculated per domain. If you have defined multiple pairs of rotational periodic interfaces in the same domain, and they spanned different sector angles, you should check the CFX-Solver Output file to see which sector angle was used.

2.4.2.1.5. Total Pressure (Stable)

If you want to specify the total pressure at an inlet, you can use the `Total Pressure` option. You must specify the **Relative Total Pressure** and a flow direction. For details, see Flow Direction (p. 122).

The boundary mass flow is an implicit result of the flow simulation. This option is available for both single and multiphase simulations. When running multiphase simulations, the boundary condition is applied as if all phases are incompressible. For details, see Multiphase Total Pressure in the *CFX-Solver Theory Guide*.

2.4.2.1.6. Stationary Frame Total Pressure (Stable)

This is the same as the `Total Pressure` condition in a stationary domain. In a rotating domain, the total pressure is based on stationary frame conditions.

2.4.2.1.7. Static Pressure

If you want to specify the static pressure at an inlet, you can use the `Static Pressure` option. You must specify the **Relative Static Pressure** and a flow direction. For details, see Flow Direction (p. 122).

The boundary mass flow is an implicit result of the flow simulation. For multiphase simulations, the static pressure is the same for all phases and the flow direction on a fluid dependent basis.

This setting is less stable than the total pressure condition because it weakly constrains the momentum that passes through the boundary. It is constrained only by transverse stresses. It is therefore a useful condition to obtain a fully developed velocity profile but is also less robust than the total pressure condition.

2.4.2.1.8. Fluid Velocity

This option is only available for an inhomogeneous multiphase simulation. When this option is selected, the **Mass and Momentum** information is set on a per fluid basis using `Normal Speed`, `Cartesian/Cylindrical Velocity Components` or `Mass Flow Rate`.

If the domain is porous, the velocity components that you specify are interpreted as true, not superficial.

2.4.2.2. Flow Direction

The flow direction can be specified as normal to the boundary, in terms of Cartesian components or in terms of cylindrical direction components. The direction constraint for the normal to boundary option is the same as that for the Normal Speed option.

If the flow direction is set as normal to the boundary, a uniform mass influx is assumed to exist over the entire inlet boundary. Also, if the flow direction is set using Cartesian or cylindrical components, the component normal to the boundary condition is ignored. Again, a uniform mass influx is assumed.

When using cylindrical direction components, a rotation axis must be set for stationary domains and can optionally be set for rotating domains. The axis used follows the same rules as when setting cylindrical velocity components. For details, see Cylindrical Velocity Components (p. 120).

For static pressure inlets, the flow direction can also be set to `Zero Gradient`, which implies that the velocity gradient perpendicular to the boundary is zero. This is the most appropriate option for fully developed flow. For best accuracy with this option, the mesh elements should be close to orthogonal perpendicular to the interface.

2.4.2.3. Turbulence

You should set reasonable values of either the turbulence intensity, or k and ε at an inlet boundary. Several options exist for the specification of turbulence quantities at inlets. However, unless you have absolutely no idea of the turbulence levels in your simulation (in which case, you can use the `Medium (Intensity = 5%)` option), you should use well chosen values of turbulence intensities and length scales.

Nominal turbulence intensities range from 1% to 5% but will depend on your specific application. The default turbulence intensity value of 0.037 (that is, 3.7%) is sufficient for nominal turbulence through a circular inlet, and is a good estimate in the absence of experimental data. The allowable range of turbulence intensity specification for an inlet boundary is from 0.001 to 0.1 (that is, 0.1% to 10%), corresponding to very low and very high levels of turbulence in the flow, respectively.

If you know roughly the level of incoming turbulence (that is, the intensity), you can specify this together with an appropriate length scale. For internal flows, a fraction of the inlet hydraulic diameter is usually a good approximation for the length scale. If you want to specify values of k and ε directly, then you can approximate incoming levels for internal flow by using the following relationships:

$$k=\frac{3}{2}\left(I U\right)^2 \tag{2.1}$$

where I is the specified turbulence intensity. ε can then be approximated using:

$$\varepsilon=\frac{k^{3/2}}{0.3\,D_h} \tag{2.2}$$

where D_h is the hydraulic diameter of the inlet. Strictly, these relationships are only applicable for a small inlet to a large domain and should be used with caution. In cases where the hydraulic dia-

meter is not appropriate, and no experimental data is available, you should use uniform profiles for the turbulence quantities, based on mean flow characteristics.

For external flows, such as flow over an airfoil, relationships based on the geometric scale of the inlet boundary may not be appropriate. Therefore, you should consider using a length scale that is based on the size of the object over which flow is moving.

In turbulent simulations, you should also consider the nature of the flow upstream of the region of interest, and determine whether it would increase or decrease the magnitude of the turbulence intensity in the region you want to model.

The options available for turbulence at an inlet are:

2.4.2.3.1. Default Intensity and Autocompute Length Scale

The default turbulence intensity of 0.037 (3.7%) is used together with a computed length scale to approximate inlet values of k and ε. The length scale is calculated to take into account varying levels of turbulence. In general, the autocomputed length scale is not suitable for external flows.

2.4.2.3.2. Intensity and Autocompute Length Scale

This option enables you to specify a value of turbulence intensity but the length scale is still automatically computed. The allowable range of turbulence intensities is restricted to 0.1%-10.0% to correspond to very low and very high levels of turbulence accordingly. In general, the auto-computed length scale is not suitable for external flows.

2.4.2.3.3. Intensity and Length Scale

You can specify the turbulence intensity and length scale directly, from which values of k and ε are calculated.

2.4.2.3.4. Low (Intensity = 1%)

This defines a 1% intensity and a viscosity ratio μ_t / μ equal to 1.

2.4.2.3.5. Medium (Intensity = 5%)

This defines a 5% intensity and a viscosity ratio μ_t / μ equal to 10.

This is the recommended option if you do not have any information about the inlet turbulence.

2.4.2.3.6. High (Intensity = 10%)

This defines a 10% intensity and a viscosity ratio μ_t / μ equal to 100.

2.4.2.3.7. Specified Intensity and Eddy Viscosity Ratio

Use this feature if you want to enter your own values for intensity and viscosity ratio.

2.4.2.3.8. k and Epsilon

Specify the values of k and ε directly.

2.4.2.3.9. Zero Gradient

You can use this option for fully developed turbulence conditions provided that you specify the correct fully developed velocity profile. However, for robustness it is recommended that, instead of using the `Zero Gradient` option, you use the fully developed profiles for the turbulence model variables.

2.4.2.4. Heat Transfer

The inlet specification for the energy equation requires a value for the fluid temperature. You should note that ALL temperatures in Ansys CFX are interpreted as absolute values.

2.4.2.4.1. Static Temperature

Specify the **Static Temperature** (thermodynamic) at an inlet.

2.4.2.4.2. Total Temperature

Specify the **Total Temperature** at an inlet. The static temperature is calculated from this value together with the specifications in the **Mass and Momentum** section.

2.4.2.4.3. Stat. Frame Total Temperature

This is the same as the **Total Temperature** condition, except for rotating domains. Total temperature is based on stationary frame velocities.

2.4.2.4.4. Total Enthalpy

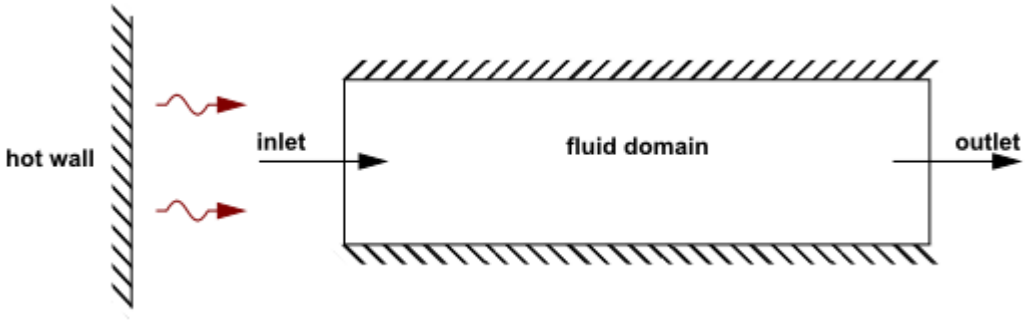
Specify the **Total Enthalpy** at an inlet. The value of total enthalpy is considered to be the absolute value, including the reference state.

2.4.2.4.5. Stationary Frame Total Enthalpy

For rotating frames the **Stationary Frame Total Enthalpy** is specified at an inlet.

2.4.2.5. Thermal Radiation

The `P1`, `Discrete Transfer` and `Monte Carlo Radiation` models enable you to account for heating of a fluid due to thermal radiation from a boundary. For the `P1` model, consider the case below where a hot wall that is outside of the fluid domain results in a radiation flux into the domain through the inlet boundary.



To model the radiation in this case, you would need to specify radiation at the inlet due to the hot wall using one of the three options described below. **Radiative Flux** into the domain for the `Monte Carlo` and `Discrete Transfer` models is specified on the **Sources** tab. For details, see Sources (Discrete Transfer and Monte Carlo models) (p. 125).

2.4.2.5.1. Radiative Heat Flux (P1 Model)

If this option is selected, you should specify the heat flux through the boundary as a result of thermal radiation.

2.4.2.5.2. Radiation Intensity (P1 Model)

The radiation intensity is the isotropic mean radiation intensity at the boundary.

2.4.2.5.3. External Blackbody Temperature

This is the effective blackbody temperature of the boundary, for details see External Blackbody Temperature (p. 474). In the example above, you might set this value to the temperature of the hot wall. In this case, you would be assuming that the hot wall can be treated as a blackbody and that there is no absorption of radiation in the gap between the hot wall and the inlet.

2.4.2.5.4. Local Temperature

For details, see Local Temperature (p. 475).

2.4.2.5.5. Sources (Discrete Transfer and Monte Carlo models)

Radiative Flux into the domain for the `Discrete Transfer` and `Monte Carlo` radiation models is specified via the **Sources** tab for the boundary condition in CFX-Pre. The `Monte Carlo` model allows directional and isotropic radiation sources to be specified: the `Discrete Transfer` model only allows isotropic radiation sources.

2.4.2.6. Transported Additional Variables at an Inlet

You must specify the values of any transported Additional Variables at an inlet.

For porous domains, if you specify a concentration for an Additional Variable, the concentration is with respect to the volume of fluid, not the total volume.

2.4.3. Inlet (Supersonic)

Supersonic inlet boundaries can be specified where the local Mach number is everywhere greater than unity. Supersonic boundary conditions can only be imposed if the `Total Energy` model is employed, and the fluid is either an ideal gas, a real fluid, or a general fluid whose density is a function of pressure.

2.4.3.1. Mass and Momentum

At a supersonic inlet boundary, the required settings depend on the choice of **Mass and Momentum** > **Option**, as shown in Table 2.1: Supersonic Inlet Settings (p. 126):

Table 2.1: Supersonic Inlet Settings

Mass and Momentum Option	Required Information
Normal Speed & Pressure	• Relative static pressure • Normal speed • Static or total temperature
Normal Speed & Total Pressure	• Relative total pressure • Normal speed • Static or total temperature or total enthalpy
Cart. Vel. & Pressure	• Relative static pressure • Cartesian velocity components • Static or total temperature
Cart. Vel. & Total Pressure	• Relative total pressure • Cartesian velocity components • Static or total temperature or total enthalpy
Cyl. Vel. & Pressure	• Relative static pressure

Mass and Momentum Option	Required Information
	• Cylindrical velocity components • Static or total temperature
Cyl. Vel. & Total Pressure	• Relative total pressure • Cylindrical velocity components • Static or total temperature or total enthalpy
Static and Total Pressure	• Relative static pressure • Relative total pressure • Total temperature

Note:

For rotating domains, the total pressure is interpreted with respect to the stationary frame of reference.

2.4.4. Inlet (Mixed Subsonic-Supersonic)

The mixed inlet boundary condition allows both subsonic and supersonic inflow within a single boundary condition. The blending between the two flow regimes is based on the user-specified inflow velocity components—total pressure and total temperature, or static pressure and static temperature. In either case, the flow solver evaluates the local Mach number from the user-specified values.

The Mach number calculation can be based on the velocity component normal to the boundary condition or the full velocity components. By default the flow solver uses the component normal to the boundary condition. You can select which calculation you want via the parameter **Blend Mach Number Type**.

The following types of mixed inlets are supported. In both cases, the velocity type depends on the **Frame Type** parameter (**Stationary** or **Rotating**, described in Cartesian Velocity Components (p. 119) and Cylindrical Velocity Components (p. 120)):

- Velocity, static pressure, and static temperature are specified:

  - For Ma < 1 (subsonic flow), this would result into the standard 'velocity specified' inlet. The user-specified pressure information is not used, but is computed as part of the iteration process.
  - For Ma > 1 (supersonic flow), all user-specified data is used as it is done for the supersonic inlet (p. 126).

- Velocity, total pressure, and total temperature/total enthalpy are specified:
  - For Ma < 1 (subsonic flow) this would result into the standard "total pressure and direction" inlet. The inflow direction is evaluated from the specified velocity components.
  - For Ma > 1 (supersonic flow) all user-specified data is used as it is done for the supersonic inlet (p. 126).
  - The total pressure and total temperature depend on the frame type (that is, total pressure can be in either the stationary or the local frame, and the same applies to total temperature).

  When you specify velocity, total pressure, and total temperature, the flow solver evaluates the static pressure and temperature directly from those values. The total enthalpy boundary condition is available only if the total energy is solved. In this case, no transformation of the specified energy is required.

Note:

Static and total conditions for pressure and temperature cannot be mixed for real gases or gas mixtures with real gas components. This restriction does not apply to gases and gas mixtures that obey the Ideal Gas law.

2.4.4.1. Supported Material Types

Mixed inlets can only be imposed if the `Total Energy` model is employed, and if the fluid is compressible. This includes the ideal gas model, NASA materials, Redlich Kwong model, RGP file materials and materials with a CEL equation of state, where density is a function of pressure.

Flows that use the wet real gas model (homogeneous binary mixtures) are currently not supported.

2.4.4.2. Mass and Momentum

At a mixed inlet, all quantities must be specified. `Static` or `Total Pressure`, velocity components and `Static` or `Total Temperature` or `Total Enthalpy` are required. Whenever the `Total Pressure` option is selected, the total pressure is in the stationary frame for rotating domains.

The following options are supported:

2.4.4.2.1. Cartesian Velocity Components and (Total) Pressure

For the mixed flow regime inlet boundary, the Cartesian components of velocity, together with a value for the **Relative Static** or **Relative Total Pressure**.

2.4.4.2.2. Cylindrical Velocity Components and (Total) Pressure

For the mixed flow regime inlet boundary, the cylindrical components of velocity, together with a value for the **Relative Static** or **Relative Total Pressure** (Relative to the domain reference pressure).

2.4.4.2.3. Normal Speed and (Total) Pressure

For the mixed flow regime inlet boundary, the normal speed, together with a value for the **Relative Static** or **Relative Total Pressure**. The flow solver evaluates the velocity direction based on the local normal to the boundary condition.

2.4.4.3. Heat Transfer

2.4.4.3.1. Static Temperature

The `Static Temperature` is specified at the mixed inlet boundary.

2.4.4.3.2. Total Temperature

The `Total Temperature` is specified at the mixed inlet boundary. The static temperature is computed from the specified `Total Temperature`. The flow solver will apply stationary frame total temperature if the domain is rotating.

2.4.4.3.3. Total Enthalpy

The `Total Enthalpy` is specified at the mixed inlet boundary. The flow solver will apply stationary frame total enthalpy if the domain is rotating.

The total enthalpy boundary condition is available only if the total energy is solved. The total enthalpy boundary condition is a 'natural' boundary condition for mixed subsonic-supersonic flow as you directly specify the total energy (that is, total enthalpy) value at the inlet. Static temperature and total temperature, on the other hand, have to be converted to total enthalpy before they can be used as a boundary condition.

2.4.4.4. Initial Guess Recommendation

Special care is required to specify the initial guess for a simulation that involves a mixed flow regime inlet. It is strongly suggested to use the `Automatic with Value` option rather than the `Automatic` option, especially if the flow near the mixed inlet is known to be locally supersonic, which locally imposes a large difference between the static and total conditions.

Flow solver robustness and convergence will be much better behaved if the flow field near the inlet is initialized as if it would be totally supersonic.

2.5. Outlet

An outlet boundary condition is used where the flow is predominantly directed out of the domain. The hydrodynamic boundary condition specification (that is, those for mass and momentum) for a subsonic

outlet involves some constraint on the boundary static pressure, velocity or mass flow. For all other transport equations, the outlet value of the variable is part of the solution.

Important:

If you accidentally set the velocity at an outlet to be flowing into the domain, the outlet will behave as an inlet: *wall building will not be turned on*.

Information on setting outlet values in CFX-Pre is available. For details, see Boundary Details: Outlet in the *CFX-Pre User's Guide*.

Information on the mathematical implementation of an outlet boundary condition is available.

- Outlet (Subsonic) in the *CFX-Solver Theory Guide*
- Outlet (Supersonic) in the *CFX-Solver Theory Guide*

2.5.1. Mass and Momentum

Information on the mathematical treatment of mass and momentum for an outlet boundary condition is available in the Mass and Momentum subsection of Outlet (Subsonic) in the *CFX-Solver Theory Guide*.

Note:

Acoustic Reflectivity settings are available for subsonic outlet boundaries that are based on:

- Average Static Pressure / Static Pressure
- Normal Speed / Velocity Components

For details, see General Non-Reflecting Boundary Conditions (p. 155). The **Acoustic Reflectivity** settings are not available for outlet boundaries that are based on mass flow.

2.5.1.1. Static Pressure

The **Relative Pressure** is maintained at a fixed specified value over the outlet boundary. The flow direction is an implicit result of the computation.

2.5.1.2. Normal Speed

Specify the magnitude of the flow velocity at the outlet. The direction of the flow is taken to be locally normal to the outlet boundary surface.

2.5.1.3. Cartesian Velocity Components

The boundary velocity components are specified. Care should be taken to ensure that a non-zero resultant is directed out of the domain.

2.5.1.4. Cylindrical Velocity Components

The components and axis are specified in the same way as for an inlet. For details, see Cylindrical Velocity Components (p. 120). The resultant flow must be directed out of the domain.

Note:

The last three options can cause severe robustness problems if the problem set-up causes a significant departure from the specified velocity distribution anywhere the flow is approaching the outlet. In this case, you may have no choice but to use one of the other options.

2.5.1.5. Average Static Pressure

When this option is selected, the static pressure is allowed to locally vary on the outlet boundary such that the average pressure is constrained in a specified manner. In all cases, the flow direction at the outlet is an implicit result of the computation. For details, see Boundary Conditions in the *CFX-Solver Theory Guide*.

The following averaging options are available:

2.5.1.5.1. Average Over Whole Outlet

This is the most commonly used option. The average constraint is applied by comparing the area weighted pressure average over the entire outlet to the user-specified value. The pressure profile at the outlet is shifted by this difference such that the new area weighted pressure average will be equal to the user-specified value. The flow direction is an implicit result of the computation.

Note that the specified average pressure may be a pressure profile. If it is, the pressure profile will be averaged and it is this profile average which will be enforced.

2.5.1.5.2. Average Above Specified Radius

The area weighted average of pressure above the specified radius is used as the comparison to the user-specified value (instead of the area weighted average over the entire outlet). The pressure profile is then shifted by the difference between these two values. The radius is calculated as the distance from the local axis of rotation (specified on the boundary for stationary domains) or the domain rotation axis for rotating domains.

The area weighted average of pressure at the outlet in the desired region may not be identical to the specified average static pressure, but it should be quite close. If the flow solver has built artificial walls on the boundary condition the actual average may differ more from the specified value than when no artificial walls are built.

2.5.1.5.3. Average Below Specified Radius

This is the same as **Average Above Specified Radius**, except that the region below the specified radius is used to obtain the area weighted average.

2.5.1.5.4. Circumferential

In this case, the average pressure is constrained within circumferential bands defined by either a local rotation axis or the domain rotation axis. The circumferential bands are either radial or axial depending on the geometry. The average pressure profile can be set as a constant, CEL analytic expression, profile function or user-defined function.

There is no restriction on how the pressure profile can spatially vary. The number of bands is automatically determined by the flow solver using the maximum number of bands that the underlying CFD mesh on the outlet will enable. Every band contains enough mesh to calculate a circumferential average. The flow solver will print a diagnostic telling you how many bands it is using. It is possible to use fewer bands than the value determined by the flow solver, but not more. You can modify the **Maximum Number of Circumferential Bands** setting.

The value of **Pres. Profile Blend** is used to blend between the specified pressure profile and a floating pressure profile where only the average is constrained. For a value of zero, the specified pressure is used only to enforce the average pressure. This allows a transverse pressure profile to develop according to upstream influences, which is much less reflective than specifying the pressure profile itself. For some flows, setting only the average pressure does not constrain the flow enough; in this case, a small amount of blending, the default value of 0.05, or 5%, may be appropriate.

2.5.1.5.5. Radial Equilibrium

In the case of Radial Equilibrium, the averaged static pressure is constrained within radial circumferential bands defined by either a local rotation axis or the domain rotation axis. The band-averaged pressure satisfies radial equilibrium between the radial pressure gradient and the centrifugal force calculated using the band-averaged density and circumferential velocity. Integrating this relationship gives pressure values for each band and requires a user-specified static pressure at a radial reference position as a starting point for the integration. Three different options are available for specifying the radial reference position:

- **Option** = Specified Radius (you must specify the **Specified Radius** parameter)
- **Option** = Minimum Radius (the radius is automatically calculated by the solver)
- **Option** = Maximum Radius (the radius is automatically calculated by the solver).

The solver generates the circumferential bands the same way as for the circumferential averaged pressure option (see Circumferential Pressure Averaging (p. 136)), except that, in this case, bands are allowed to be oriented only in the radial direction. As for all pressure averaging options, pressure profile blending can be applied to this boundary condition.

2.5.1.6. Mass Flow Rate (Bulk Mass Flow Rate for Multiphase)

When **Mass and Momentum** > **Option** is set to `Mass Flow Rate`, you must set **Mass and Momentum** > **Mass Flow Rate**. A positive value represents mass flow through the boundary in the specified flow direction. For details, see Flow Direction (p. 122).

Note:

The local velocity across the outlet boundary is part of the solution.

You must also choose one of two settings for **Mass Flow Rate Area**:

- `As Specified`

  Mass Flow Rate corresponds to the modeled sector, and is applied directly to the boundary. This setting is selected by default.

- `Total for All Sectors`

  This setting is available only for single phase cases that have rotational periodicity and that have a rotational periodic interface. If no rotational periodic interface is present, `Total for All Sectors` is identical to `As Specified`. For details on interface models with rotational periodicity, see Interface Models (p. 234).

  Mass Flow Rate corresponds to the full geometry. The pair of rotational periodic interfaces (or, in the case of multiple pairs existing in the domain, one such pair) is used to calculate the sector angle (the angle that the modeled sector spans in the full geometry). **Mass Flow Rate** is multiplied by the ratio between the sector angle and 360° before being applied to the boundary.

  Note:

  Only one sector angle is calculated per domain. If you have defined multiple pairs of rotational periodic interfaces in the same domain, and they spanned different sector angles, you should check the CFX-Solver Output file to see which sector angle was used.

The `Mass Flow Outlet Constraint` option should be chosen from one of three methods depending on the type of flow at the outlet. For details, see Mass Flow Outlet Constraint (p. 134).

Information on the mathematical treatment of mass flow rate for an outlet boundary condition is available in Mass Flow Rate: Scale Mass Flows, Mass Flow Rate: Shift Pressure with or without Pressure Profile and Mass Flow Rate: Shift Pressure with Circumferential Pressure Averaging in Outlet (Subsonic) in the *CFX-Solver Theory Guide*.

2.5.1.7. Exit Corrected Mass Flow Rate

The `Exit Corrected Mass Flow Rate` outlet boundary option adjusts the mass flow rate to total conditions at the outlet, maintaining a constant exit corrected mass flow rate. The main application of this boundary option is rotating machinery. The `Exit Corrected Mass Flow`

`Rate` boundary option enables you to sweep through the complete machine operational range, including machine operating points from choked flow to stall conditions.

This outlet boundary option allows you to specify the equivalent mass flow, based on similar criteria, corrected to a specified reference temperature and pressure. By default, the reference pressure and reference temperature are set to Standard Atmosphere Sea Level Static conditions of 15 [°C] and 1 [atm] respectively. Information on the mathematical treatment of exit corrected mass flow rate as an outlet boundary condition is available in Exit Corrected Mass Flow Rate in the *CFX-Solver Theory Guide*.

For this outlet boundary condition you are required to specify:

- **Exit Corrected Mass Flow Rate**

  For an ideal gas, the exit corrected mass flow rate is calculated as:

$$\dot{m}_{cor} = \dot{m} \frac{\sqrt{\tilde{T}_{o1} / T_{ref}}}{\tilde{P}_{o1} / P_{ref}} \tag{2.3}$$

  where, $\tilde{P}_{o1}$ and $\tilde{T}_{o1}$ are mass averaged values of total pressure and temperature in the stationary frame at the outlet. P_{ref} and T_{ref} are the reference conditions, which are constants in the equation. It should be noted that the numerical behavior of the boundary condition is not affected by the choice of reference conditions. Rather, the reference conditions simply provide a dimensional meaning to the otherwise non-dimensional mass flow rate.

Because the specified corrected mass flow and reference conditions are constant, you can observe that the resulting mass flow rate must be proportional to the exit total pressure and inversely proportional to the square root of the exit total temperature (or equivalently, inversely proportional to the stagnation speed of sound) in order to maintain a constant exit corrected mass flow. This allows the boundary condition to adapt dynamically to varying operating conditions, while remaining stable as the flow develops from the initial guess.

Currently, the `Exit Corrected Mass Flow Rate` outlet boundary condition is only available for ideal gas materials with the thermal energy or total energy options enabled. For more details on using the **Exit Corrected Mass Flow Rate** boundary condition, see Computing Speedlines for a Machine in the *CFX Reference Guide*.

Along with **Mass Flow Rate**, you can set **Mass Flow Rate Area**. Because cases that use `Exit Corrected Mass Flow Rate` usually feature rotational periodicity, the default **Mass Flow Rate Area** option for new cases is `Total for All Sectors`. For details on **Mass Flow Rate Area** options, see Mass Flow Rate (Bulk Mass Flow Rate for Multiphase) (p. 133).

The `Mass Flow Outlet Constraint` option should be chosen from one of three methods depending on the type of flow at the outlet. For details, see Mass Flow Outlet Constraint (p. 134).

2.5.1.8. Mass Flow Outlet Constraint

For the **Mass Flow Rate** and **Exit Corrected Mass Flow Rate** options you can specify one of three `Mass Flow Outlet Constraint` methods, depending on the type of flow:

2.5.1.8.1. Pressure Shape Unconstrained

This is the default option in Ansys CFX. The mass flow distribution is a function of the mass flow just upstream of the outlet boundary, and the specified mass flow is enforced at each timestep. The pressure distribution which results is an implicit part of the simulation and is not constrained by the boundary condition.

2.5.1.8.2. Uniform Mass Flux

This specifies a constant mass flow distribution (ρU_{normal} is the same across the entire outflow boundary). This option is most useful when the flow is highly tangential relative to the outlet. Under these circumstances, it is difficult to determine the mass flow distribution when the `Pressure Shape Unconstrained` or `Pressure Shape Constrained` options are used. This is because the mass distribution just upstream of and normal to the outlet is a small component relative to the tangential component of the mass flow. Conversely, the `Uniform Mass Flux` option should not be used if the mass flow at the outlet is not highly tangential. This would adversely affect the flow upstream of the boundary where the mass flow distribution is not uniform.

2.5.1.8.3. Pressure Shape Constrained

This option allows you to constrain the pressure profile over the entire boundary. Like `Pressure Shape Unconstrained`, the mass flow distribution is a function of the mass flow just upstream of the outlet boundary, and the specified mass flow is enforced at each timestep. However, the solver also computes the pressure shift between your specified profile and the implicit (floating) profile that enforces the specified mass flow rate.

The `Pressure Shape Constrained` option is generally useful if:

- The pressure profile at the outlet is known.
- The pressure shape is not known, but the implicit pressure profile shape computed using the `Pressure Shape Unconstrained` option is not acceptable.
- The model converges poorly using other **Mass Flow Outlet Constraint** methods.

If you do not constrain the profile, then the static pressure profile floats and becomes an implicit result of the solution. The shift value works out to the average pressure that enforces the specified mass flow rate. The solver still computes the necessary pressure shift, but otherwise behaves as if the **Mass Flow Outlet Constraint** setting is `Pressure Shape Unconstrained`.

To use `Pressure Shape Constrained`, you must set the following values:

Pres. Profile Shape

This is a value or expression that describes the pressure profile at the outlet and is entered in units of pressure. Because the pressure level at the outlet is part of the solution, only the relative variation in the profile is relevant. For example, specifying a constant pressure profile of 0 Pa is the same as specifying a constant pressure profile of 101325 Pa.

Pres. Profile Blend

This is a measure of enforcement of the specified pressure profile to the outlet boundary. For any blend value, θ between 0 and 1, the pressure is equal to:

$$\theta P_{\text{profile}} + \left(1-\theta\right) P_{\text{floating}} \tag{2.4}$$

where P_{profile} is the pressure value calculated from the user-specified profile and P_{floating} is derived from the value just upstream of the outflow boundary.

By default **Pres. Profile Blend** is set to 0.05. A blend value of 0.0 specifies no enforcement of the pressure profile. In this case, the behavior of the `Pressure Shape Unconstrained` method is recovered. A blend value of 1.0 fully imposes the pressure profile shape (but not the level) as well as the mass flow leaving the domain.

2.5.1.8.3.1. Circumferential Pressure Averaging

For cases that use the `Pressure Shape Constrained` option, a circumferential pressure averaging option is available. This option creates a meridional pressure profile shape based on any specified spatially varying pressure values, and then allows the local relative static pressure distribution to vary within circumferential bands (as done for average pressure outlets) based on that meridional pressure profile shape. This also makes it easy to switch between including or not including circumferential averaging. Note that with circumferential pressure averaging, there is no restriction on circumferential pressure variations in a given band (that is, only the average for a given band is constrained based on the specified pressure profile shape).

2.5.1.9. Degassing Condition (Multiphase only)

Degassing boundary conditions are used to model a free surface from which dispersed bubbles are permitted to escape, but the liquid phase is not. They are useful for modeling flow in bubble columns.

When **Degassing Boundary Condition** is selected as the **Flow Specification** of an outlet, the continuous phase and any dispersed solid phases that may be present see this boundary as a free-slip wall and do not leave the domain. Dispersed fluid phases and Lagrange particles see this boundary as an outlet. However, the outlet pressure is not specified. Instead, a pressure distribution is computed on this fixed-position boundary, and can be interpreted as representing the weight of the surface height variations in the real flow.

Because the **Degassing Boundary Condition** does not specify a pressure value, a fixed pressure reference point will be set automatically for a domain with an incompressible flow with degassing boundaries, but no pressure boundaries.

2.5.1.10. Fluid Velocity (Multiphase only)

This option is only available for an inhomogeneous multiphase simulation. When this option is selected, the **Mass and Momentum** information is set on a per fluid basis using `Cartesian` or `Cylindrical Velocity Components`.

2.5.1.11. Supercritical (Multiphase only)

Supercritical free surface flow means that the liquid velocity exceeds the local wave velocity, and nothing needs to be set at the outlet (analogous to the supersonic condition for compressible flow). However, for stability purposes, **Relative Pres. in Gas** (that is, Relative Pressure in Gas: the pressure in the gas phase above the free surface interface) must be set. Note that transcritical free surface flows often have two solutions: one subcritical at the outlet and the other supercritical. Because of

this, you may sometimes need to start with a `Static Pressure` or `Average Static Pressure` outlet condition based on a hydrostatic condition which forces the elevation into the supercritical regime.

2.5.2. Turbulence, Heat Transfer, and Additional Variables

For scalar quantities, the CFX-Solver does not require any variable values to be specified. The variables are obtained implicitly from the solution information on the boundary nodes.

2.5.3. Thermal Radiation

This is the same as specifying **Thermal Radiation** at an inlet. For details, see Thermal Radiation (p. 124).

2.5.4. Mesh Motion

Mesh motion can occur on outlet boundaries in domains where **Mesh Deformation** has been activated for the domain. (See Mesh Deformation in the *CFX-Pre User's Guide* for information about activating mesh deformation for the domain.) Note that solution errors will be introduced if any of the specified motion is normal to the boundary faces. For details, see Mesh Deformation (p. 42).

2.5.5. Outlet (Supersonic)

For a supersonic outlet, the flow is not influenced at all by downstream conditions. All values of dependent variables, with the exception of radiation intensity, are extrapolated from upstream, and the constant gradient constraint is not imposed. The specification of a supersonic outlet boundary condition, therefore, requires no accompanying values unless a **Thermal Radiation** model is used. For details, see Thermal Radiation (p. 124).

Ansys CFX will behave unpredictably if part or all of fluid exiting the region assigned a supersonic outlet is subsonic. The CFX-Solver might fail to converge or diverge in this situation. To correct this, you should move the outlet to a location where the flow is supersonic over the entire boundary, or move it to a subsonic location and specify a different outlet boundary condition.

2.6. Opening

An opening can be used at a boundary where the flow is into and/or out of the domain. The opening boundary condition type is only available for subsonic boundaries.

Information on setting the values for an opening in CFX-Pre is available. For details, see Boundary Details: Opening in the *CFX-Pre User's Guide*.

Information on the mathematical implementation of an opening boundary condition is available in Opening in the *CFX-Solver Theory Guide*.

2.6.1. Mass and Momentum

Note:

Acoustic Reflectivity settings are available for opening boundaries that are based on:

- Static/Opening Pressure and Direction
- Entrainment
- Velocity Components

For details, see General Non-Reflecting Boundary Conditions (p. 155).

2.6.1.1. Cartesian Velocity Components

The Cartesian components of the flow velocity at the opening are specified in the same way as for an inlet. For details, see Cartesian Velocity Components (p. 119). The components can be specified such that both inflow and outflow occur over different regions of the boundary.

2.6.1.2. Cylindrical Velocity Components

The components and axis are specified in the same way as for an inlet. For details, see Cylindrical Velocity Components (p. 120). The components can be specified such that both inflow and outflow occur over different regions of the boundary.

2.6.1.3. Opening Pressure and Direction

When the flow direction is into the domain, the pressure value is taken to be total pressure based on the normal component of velocity. When it is leaving the domain, it is taken to be relative static pressure.

This option also requires that the **Flow Direction** is specified, and is used if the flow is into the domain. For details, see Flow Direction (p. 122).

This is the most robust and stable setting for a pressure specified opening because it puts a constraint on the momentum transported through the boundary condition.

This option is available for both single and multiphase simulations. When running multiphase simulations, the total pressure is applied as if all phases are incompressible. For details, see Multiphase Total Pressure in the *CFX-Solver Theory Guide*.

2.6.1.4. Static Pressure and Direction

The pressure value is taken to be static pressure for both the inflow and outflow. This option also requires that the **Flow Direction** is specified, and is used if the flow is into the domain. For details, see Flow Direction (p. 122).

If the flow is coming into the domain, this setting is the least stable because it does not constrain the momentum allowed to flow through the boundary. In this case, any momentum flow satisfies

the static pressure condition, which can cause severe robustness problems. Always consider a pressure-specified opening before using this option.

2.6.1.5. Entrainment

When `Entrainment` is selected, the **Pressure Option** check box becomes available. When the latter is selected, you can set the sub-option to `Static Pressure` or `Opening Pressure`.

- Entrainment with the `Static Pressure` option is similar to the `Static Pressure and Direction` option, except that the direction is obtained by enforcing the velocity gradient perpendicular to the boundary to be zero.
- Entrainment with the `Opening Pressure` option has a similar behavior to a pressure-specified opening and has similar stability and robustness but does not require that a flow direction is specified. Instead, the flow solver locally calculates the flow direction based on the direction of the velocity field. When the flow direction is into the domain, the pressure value is taken to be total pressure based on the normal component of velocity. When it is leaving the domain, it is taken to be relative static pressure.

When the **Pressure Option** check box is not selected, the `Static Pressure` sub-option takes effect by default.

The `Entrainment` option can be quite useful for situations in which the main flow tends to pull fluid through the boundary where the flow direction is unknown.

2.6.1.6. Fluid Velocity

This option is only available for an inhomogeneous multiphase simulation. When this option is selected, the **Mass and Momentum** information is set on a per-fluid basis using `Normal Speed`, `Cartesian Velocity Components` or `Cylindrical Velocity Components`.

2.6.2. Loss Coefficient

You can optionally specify a **Loss Coefficient** which can be used to model the pressure drop across a screen or other planar resistance at the opening. The pressure drop is calculated using:

$$\Delta p_{\text{loss}} = \frac{1}{2} f \rho U_n^2 \tag{2.5}$$

where f is a specified **Loss Coefficient** and U_n is the normal component of velocity at the opening boundary location. When the opening behaves as an inlet, the pressure drop is applied *before* flow enters the domain. When the opening behaves as an outlet, the pressure drop is deemed to occur *after* the flow has left the domain. This results in the following behavior:

2.6.2.1. For a Pressure-Specified Opening

For outflow, the boundary static pressure is the specified **Relative Pressure**, plus a pressure loss, if specified.

For inflow, the boundary static pressure is the specified **Relative Pressure** converted from **Total** to **Static Pressure** using the normal component of the velocity, minus a pressure loss, if specified.

2.6.2.2. For a Static-Pressure-Specified Opening

For outflow, the boundary static pressure is the specified **Relative Pressure**, plus a pressure loss, if specified.

For inflow, the boundary static pressure is the specified **Relative Pressure**, minus a pressure loss, if specified.

2.6.3. Heat Transfer

Specify the **Heat Transfer** as for an inlet boundary condition. This value is only used for the inflow portion of the opening. You can specify either:

- Static Temperature
- Opening Temperature

  The opening temperature is similar to the `Total Temperature`, but the dynamic component is based only on the normal component of velocity.

2.6.4. Turbulence

Specify the turbulence quantities as for an inlet boundary condition. For details, see Turbulence (p. 122). These values are only used for the inflow portion of the opening.

2.6.5. Thermal Radiation

This is the same as specifying **Thermal Radiation** at an inlet. For details, see Thermal Radiation (p. 124).

2.6.6. Additional Variables

Specify the values of any Additional Variables as for an inlet boundary condition. These values are only used for the inflow portion of the opening. For flow out of the domain, the CFX-Solver calculates the values of Additional Variables from the solution field.

2.6.7. Mesh Motion

Mesh motion can occur on opening boundaries in domains where **Mesh Deformation** has been activated for the domain. (See Mesh Deformation in the *CFX-Pre User's Guide* for information about activating mesh deformation for the domain.) Note that solution errors will be introduced if any of the specified motion is normal to the boundary faces. For details, see Mesh Deformation (p. 42).

2.7. Wall

Walls are solid (impermeable) boundaries to fluid flow. Walls allow the permeation of heat and Additional Variables into and out of the domain through the setting of flux and fixed value conditions at wall boundaries.

Walls are the default boundary condition in CFX-Pre for fluid-world and solid-world regions; any of these regions that are not part of an existing boundary condition will remain in a default wall boundary when

the CFX-Solver input file is written. For details, see Default Boundary Condition in the *CFX-Pre User's Guide*.

The treatment of wall boundary conditions for turbulent flow is the same as for laminar flow, except for no-slip conditions. For details, see Wall Function in the *CFX-Pre User's Guide*.

Information on setting boundary conditions at walls in CFX-Pre is available in Boundary Details: Wall in the *CFX-Pre User's Guide*.

Additional mathematical information on the wall boundary condition is available in Wall in the *CFX-Solver Theory Guide*.

Additional information on wall boundaries in multiphase flow is available in Wall Boundaries in Multiphase (p. 341).

2.7.1. Mass and Momentum

The following options are available for modeling the influence of a wall boundary on mass and momentum:

- No Slip Wall (p. 141)
- Free Slip Wall (p. 142)
- Finite Slip Wall (p. 142)
- Specified Shear (p. 143)
- Counter-rotating Wall (p. 143)
- Rotating Wall (p. 143)

For additional details, see Wall in the *CFX-Solver Theory Guide*.

2.7.1.1. No Slip Wall

This is the most common type of wall boundary condition implementation. The fluid immediately next to the wall assumes the velocity of the wall, which is zero by default. Non-zero wall velocities are created in two ways: by explicitly setting a wall velocity and/or by activating mesh deformation.

Cartesian or cylindrical wall velocity components can be set to emulate wall motion even when mesh motion has not been activated. When using `Cartesian` or `Cylindrical Velocity Components`, the velocity is always in relation to the local (relative) frame of reference (that is, relative to the rotating frame in a rotating domain). When using `Cylindrical Velocity Components`, an axis for the cylindrical coordinate system must be specified in one of two ways:

- You can select the axis of the cylindrical coordinate system for the boundary condition (called the **Rotation Axis**) as an axis of the global coordinate frame, Coord 0.
- You can specify **Rotation Axis From** and **Rotation Axis To** points. These are the two end points of the axis. The global coordinate system must be used to specify these points. The positive z component of the cylindrical coordinates is in the direction of the vector from the **Rotation Axis From** point to the **Rotation Axis To** point.

For rotating domains, the axis will default to the domain axis of rotation.

Note:

Only the component of the specified wall velocity that is tangent to the wall is applied. Mesh motion should be used if the desired wall velocity is normal to the wall, or has a component normal to the wall.

Mesh motion can occur on wall boundaries in domains where **Mesh Deformation** has been activated for the domain. (See Mesh Deformation in the *CFX-Pre User's Guide* for information about activating mesh deformation for the domain.) If this is the case, the wall velocity (as set above) can be made relative to the boundary frame or to the mesh motion occurring on the boundary. When the wall velocity is made relative to the boundary frame, only the specified wall velocity is assumed by the flow. When the wall velocity is made relative to the mesh motion, the velocity due to mesh motion is super-imposed on the specified wall velocity. The default behavior is that the wall velocity is relative to the boundary frame when `Parallel to Boundary` or `Surface of Revolution` mesh motion boundary conditions are used, and relative to the mesh motion for all other boundary conditions types. For details, see Mesh Deformation (p. 42).

For example, in a piston-cylinder simulation, a zero velocity, no slip condition would be applied to all walls. The wall velocity would typically be made relative to the mesh motion for the moving piston boundary, and relative to the boundary frame for the cylinder side-walls. This would ensure that the fluid is properly affected by the motion of the piston, and that it is not dragged by the motion of the mesh on the cylinder side-walls.

Note:

In a multiphase case, if one fluid uses a No Slip Wall, other fluids must see the same wall velocity or use the Free Slip condition.

2.7.1.2. Free Slip Wall

Free-slip, where the shear stress at the wall is zero ($\tau = 0$), and the velocity of the fluid near the wall is not retarded by wall friction effects. In a multiphase case, if one fluid uses a `Free Slip` wall, other fluids can use any wall influence condition.

2.7.1.3. Finite Slip Wall

The Finite Slip Wall option is available only for laminar flows. This option causes the fluid to "slip" at the wall when the wall shear stress is greater than a critical stress, τ_c. A typical use of the finite slip wall option is to simulate the flow of non-Newtonian fluid. When the wall shear stress is greater than the critical stress, the wall velocity at the edge of the shear-thinning boundary layer is computed without resolving it directly. Ansys CFX simulates the slip by using a moving wall with the wall speed computed as follows:

$$\begin{gathered} U_w = 0, \tau \le \tau_c \\ U_w = U_s \left[\frac{\tau - \tau_c}{\tau_n} \right]^m \exp\left(-\frac{BP}{\tau_n} \right), \tau > \tau_c \end{gathered} \tag{2.6}$$

where U_s is the slip speed, τ_n is a normalizing stress, m is a positive power, B is a pressure coefficient, and P is the pressure.

2.7.1.4. Specified Shear

The `Specified Shear` option enables you to specify the shear stress components on the fluid. The free-slip option is the case of a specified shear equal to 0. The current implementation removes any normal component of the user-specified shear stress.

For turbulent flows, the specified shear is applied directly; no considerations about turbulent wall functions are taken into account.

2.7.1.5. Counter-rotating Wall

For domains specified with a **Rotating Frame of Reference**, a `Counter-rotating Wall` can be specified. The wall boundary is assumed to be stationary with respect to the stationary frame, essentially in counter-rotation with the rotating fluid and uses a no slip condition. In a multiphase case, if one fluid uses a `Counter-rotating Wall`, other fluids must use the same condition or the `Free Slip` condition.

2.7.1.6. Rotating Wall

This option applies to both stationary and rotating domains and enables the wall to rotate with a specified angular velocity. The angular velocity is always in relation to the local (relative) frame of reference (that is, relative to the rotating frame in a rotating domain). An axis must be specified in a stationary domain and can optionally be specified in a rotating domain. Axis specification follows the same rules as for cylindrical velocity component inlets. For details, see Cylindrical Velocity Components (p. 120).

2.7.2. Wall Roughness

For simulations using turbulence models, you can describe the wall surface as being smooth or rough. For rough walls, the equivalent sand-grain roughness is required as input parameter. An additional modeling option, `High Roughness (Icing)`, is available when using the SST turbulence model. The details of the rough wall treatment for turbulence models based on the ε and ω – equation can be found in Treatment of Rough Walls in the *CFX-Solver Theory Guide*.

This option is not available for the `Free Slip` wall boundary condition, or when a turbulence model is not used (that is, Laminar).

2.7.3. Wall Contact Model

The **Wall Contact Model** setting controls how the wall area fraction is calculated for the purpose of partitioning the wall values of total shear stress, and other variables, into variables for the phase-specific contributions.

The options are:

- `Use Volume Fraction`

  For the momentum equation, the volume fraction of a given phase is multiplied by the total wall shear stress to determine the component of wall shear stress that is due to that phase. For energy and other scalar equations, an analogous procedure is applied.

- `Specify Area Fraction`

  For the momentum equation, the area fraction of a given phase is multiplied by the total wall shear stress to determine the component of wall shear stress that is due to that phase. For energy and other scalar equations, an analogous procedure is applied.

  This option is suitable, for example, in the case of a thin liquid film whose thickness is smaller than the nearest node distance from the wall. In this case, the liquid volume fraction may be very small, but the area fraction of the liquid is one. You may set area fractions of one or zero to model films.

  For a case where the wall is partially covered by liquid film, the liquid area fraction may be a complicated function of liquid volume fraction. In such a case, you can use CEL to implement a complex model for the wall contact area fraction.

For details on the area contact model, see Area Contact Model (p. 341).

Note:

If the RPI boiling model is used then the **Wall Contact Model** option is used for the momentum equation but not the energy equation. For details, see Using a Wall Boiling Model (p. 328).

2.7.4. Wall Adhesion

The **Wall Adhesion** option is available for free surface flows when surface tension forces are modeled. For details, see Surface Tension (p. 347).

2.7.5. Heat Transfer

2.7.5.1. Adiabatic

The heat flux across the wall boundary is zero.

When radiation is included, the total heat flux is zero:

$$q_w = 0 = q_{\text{rad}} + q_{\text{cond}} \tag{2.7}$$

For details, see CFX-Solver Output File (Radiation Runs) in the *CFX-Solver Manager User's Guide*.

For details on the related results file variable, see Wall Temperature (T_w) (p. 146).

2.7.5.2. Fixed Temperature

The wall boundary is fixed at a specified temperature T_w. The heat flux into the domain is calculated for laminar flows from the temperature gradient at the wall, and for turbulent flows by:

$$q_w = h_c \left(T_w - T_{nw} \right) \tag{2.8}$$

where T_{nw} is the near-wall temperature and h_c involves the use of turbulent wall functions. For details, see Heat Flux in the Near-Wall Region in the *CFX-Solver Theory Guide*.

For details on the related results file variable, see Wall Temperature (T_w) (p. 146).

2.7.5.3. Heat Flux and Wall Heat Flux

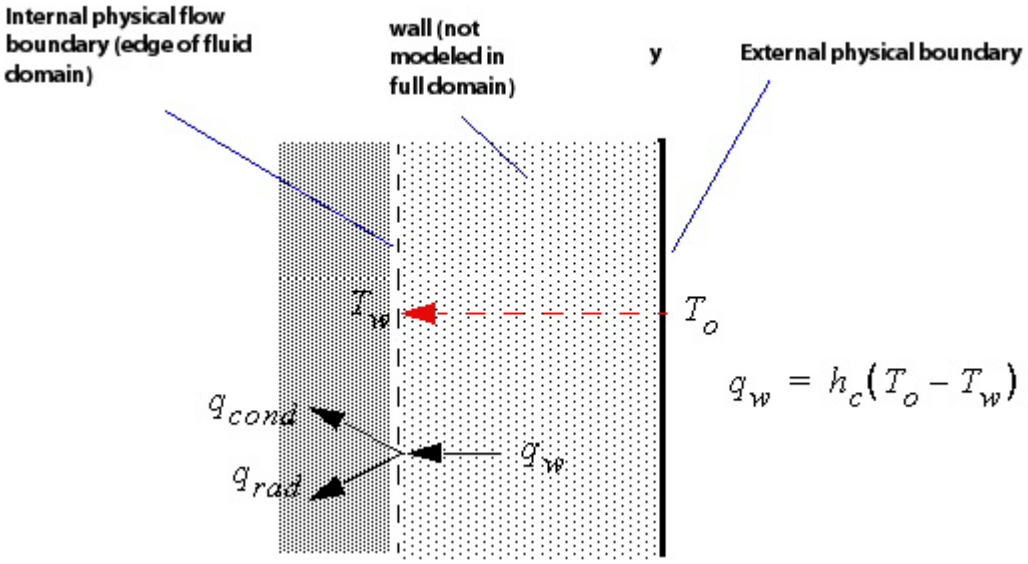
A heat flux is specified across the wall boundary. A positive value indicates heat flux into the domain. For multiphase cases, when the bulk heat flux into both phases is set, this option is labeled `Wall Heat Flux` instead of `Heat Flux`. When set on a per fluid basis, this option is labeled `Heat Flux`.

For details on the related results file variables, see Wall Heat Flux and Heat Flux (q_w) (p. 147).

2.7.5.4. Heat Transfer Coefficient and Wall Heat Transfer Coefficient

In this case, the heat flux at a wall boundary is implicitly specified using an external heat transfer coefficient, h_c, and an outside or external boundary temperature, T_o. This boundary condition can be used to model thermal resistance outside the computational domain, as in the diagram below:

Figure 2.4: Heat Transfer



The heat flux at the **Heat Transfer Coefficient** wall is calculated using:

$$q_w = h_c(T_o - T_w) = q_{\text{rad}} + q_{\text{cond}} \tag{2.9}$$

where T_o is the specified outside or external boundary temperature. T_w is the temperature at the wall (edge of the domain), which for turbulent flows is calculated from a surface energy balance and for laminar flows is the boundary temperature field calculated by the solver. The heat flux q_w is the total heat flux from conduction and radiation when radiation is modeled. For details, see CFX-Solver Output File (Radiation Runs) in the *CFX-Solver Manager User's Guide*.

For details on the related results file variables, see Wall Heat Transfer Coefficient (h_c) and Wall Adjacent Temperature (T_{nw}) (p. 147) and Wall External Heat Transfer Coefficient and Wall External Temperature (p. 148).

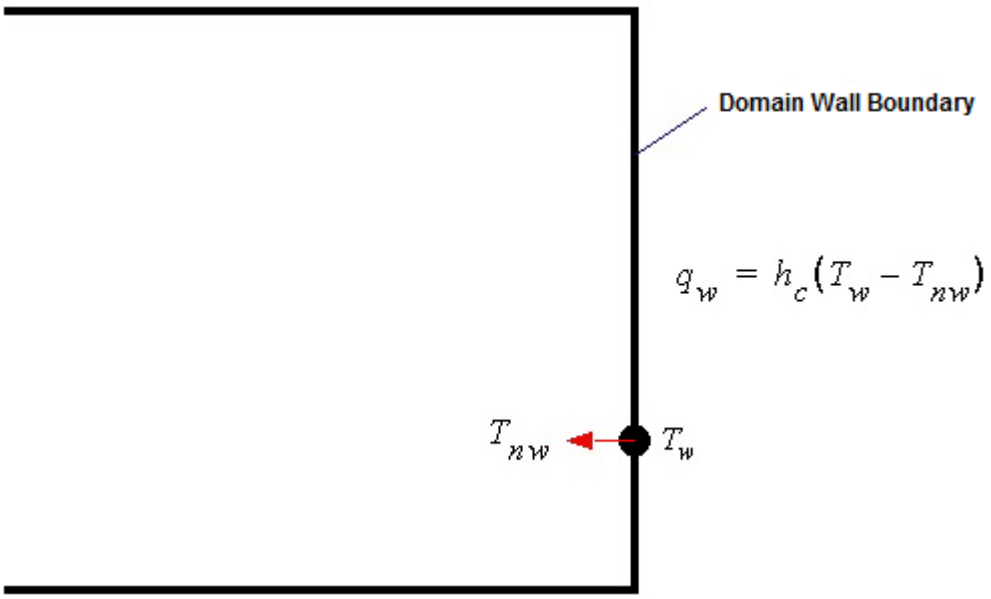
2.7.5.5. System Coupling

Temperature data is set according to System Coupling. For details, see Coupling CFX to an External Solver: System Coupling Simulations in the *CFX-Solver Modeling Guide* (p. 529).

2.7.5.6. Results File Variables for Postprocessing

The results file contains several wall-only variables when the Thermal or Total Energy heat transfer models are active. Figure 2.5: Variables at a Wall Boundary (p. 146) depicts the variables that can be written at a wall boundary condition.

Figure 2.5: Variables at a Wall Boundary



The following sections provide details about these wall-only results file variables:

2.7.5.6.1. Wall Temperature (Tw)

2.7.5.6.2. Wall Heat Flux and Heat Flux (qw)

2.7.5.6.3. Wall Heat Transfer Coefficient (hc) and Wall Adjacent Temperature (Tnw)

2.7.5.6.4. Wall External Heat Transfer Coefficient and Wall External Temperature

2.7.5.6.1. Wall Temperature (T_w)

This is the hybrid temperature field at a wall boundary condition. For Fixed Temperature walls, the wall temperature is the specified value T_w. For all other heat transfer boundary conditions, the wall temperature is calculated from turbulent wall functions when running a turbulent flow model. For laminar flow modeling the wall temperature is just the local fluid temperature at the vertex adjacent to the wall.

When running the inhomogeneous heat transfer model, this variable is written per fluid. Similarly, for a porous solid, this variable is written for the solid phase as well.

2.7.5.6.2. Wall Heat Flux and Heat Flux (q_w)

`Wall Heat Flux` is the total heat flux into the domain, including convective and radiative contributions. There are also variables called `Wall Convective Heat Flux` and `Wall Radiative Heat Flux` for the separate convective and radiative contributions.

When running the inhomogeneous heat transfer model, this variable is written per fluid. Similarly, for a porous solid, this variable is written for the solid phase as well.

`Heat Flux` is also the total heat flux into the domain, including convective and radiative contributions. This variable is different from the `Wall Heat Flux` in several ways:

- There are no equivalent separate variables for the convective and radiative components.
- It can be plotted local to a specific boundary condition. `Wall Heat Flux` contains contributions from adjacent boundary conditions. For example, edge values of `Heat Flux` on a heat-transfer coefficient boundary are not affected by being adjacent to an adiabatic wall.
- It is computed on boundary vertices by the post processor directly from the energy flows written to the results file by the CFX-Solver, and the geometric area of the control volume face at the boundary vertex. This is sometimes advantageous over the `Wall Heat Flux` variable in that it eliminates the arithmetic averaging procedure used by the CFX-Solver to compute `Wall Heat Flux` at each boundary vertex from the faces adjacent to each vertex. For porous domains, the use of the geometric area introduces another difference to the `Wall Heat Flux` variable since the CFX-Solver uses the phasic area (geometric area * fluid/solid volume porosity) instead of the geometric area at the boundary face. `Heat Flux` should be used in preference to `Wall Heat Flux` when this is possible.

In the CFX-Solver the heat flux variables available for use in expressions are just `Wall Heat Flux` and `Wall Convective Heat Flux`. These variables are based on energy flows and so are equivalent to the variable `Heat Flux` and its convective component in the post-processor.

2.7.5.6.3. Wall Heat Transfer Coefficient (h_c) and Wall Adjacent Temperature (T_{nw})

These two variables are calculated as part of the convective heat transfer at a wall.

Laminar flow model:

- Wall Heat Transfer Coefficient, h_c, is calculated by rearranging the expression for the convective heat flux in Equation 2.8 (p. 144) and setting the wall temperature as described above.
- Wall Adjacent Temperature, T_{nw}, is the average temperature in the element adjacent to the wall.

Turbulent flow model:

- Wall Heat Transfer Coefficient is given by the thermal wall functions. The theory of thermal wall functions can be found at Heat Flux in the Near-Wall Region in the *CFX-Solver Theory Guide*.
- For turbulent flow without viscous work active, the Wall Adjacent Temperature is the conservative (solved for) temperature in the control volume adjacent to the wall. On the

other hand, if viscous work is active, Equation 2.267 in the *CFX-Solver Theory Guide*, can be rearranged as:

$$q_w = \frac{\tau_w c_p}{\mathrm{Pr}_t U}\left(T_w - T_f - \frac{\mathrm{Pr}_t U^2}{2 c_p}\right) \tag{2.10}$$

Therefore, with viscous work active, the Wall Adjacent Temperature becomes:

$$T_{nw} = T_f + \frac{\mathrm{Pr}_t U^2}{2 c_p} \tag{2.11}$$

For details of the treatment of heat transfer at a wall for turbulent flows, refer to the discussion of Heat Flux in the Near-Wall Region in the *CFX-Solver Theory Guide*.

2.7.5.6.4. Wall External Heat Transfer Coefficient and Wall External Temperature

These variables are written for boundaries that have a specified external heat transfer coefficient and a specified external boundary temperature.

Additional information for multiphase cases is available. For details, see Wall Boundaries in Multiphase (p. 341).

2.7.6. Mesh Motion

All options for **Mesh Motion** are available for wall boundaries in domains where **Mesh Deformation** has been set to `Regions of Motion Specified`. For details, see Mesh Deformation (p. 42).

Note:

Unlike all other boundary condition values in Ansys CFX, which are set on mesh faces, **Mesh Motion** boundary values are applied to all *nodes* on the boundary region. Care is therefore required to ensure that consistent **Mesh Motion** boundary conditions are applied at nodes that are common to adjacent boundary regions. For example, a `Specified Displacement` condition on one boundary region must produce no displacement for nodes that are common to adjacent boundary regions where `Stationary` conditions have been applied.

2.7.7. Thermal Radiation

2.7.7.1. Opaque

Set an **Emissivity**. A blackbody exhibits perfect emission and has an **Emissivity** value of `1`. For details, see Opaque (p. 467).

2.7.7.2. Sources

For the `Discrete Transfer` and `Monte Carlo` models, radiation sources can be specified at wall boundaries. For details, see Sources (Discrete Transfer and Monte Carlo models) (p. 125).

2.7.8. Equations Governing Additional Variables

The equations governing the behavior of Additional Variables are analogous to those for heat transfer, and the specification of boundary conditions at the wall are, therefore, very similar. For details, see Heat Transfer (p. 144).

2.8. Symmetry Plane

Often a physical problem has the property that all aspects of the flow are symmetric about some physical (flat) plane. A problem is symmetric about a plane when the flow on one side of the plane is a mirror image of flow on the opposite side. By definition, a symmetry boundary condition refers to a planar boundary surface.

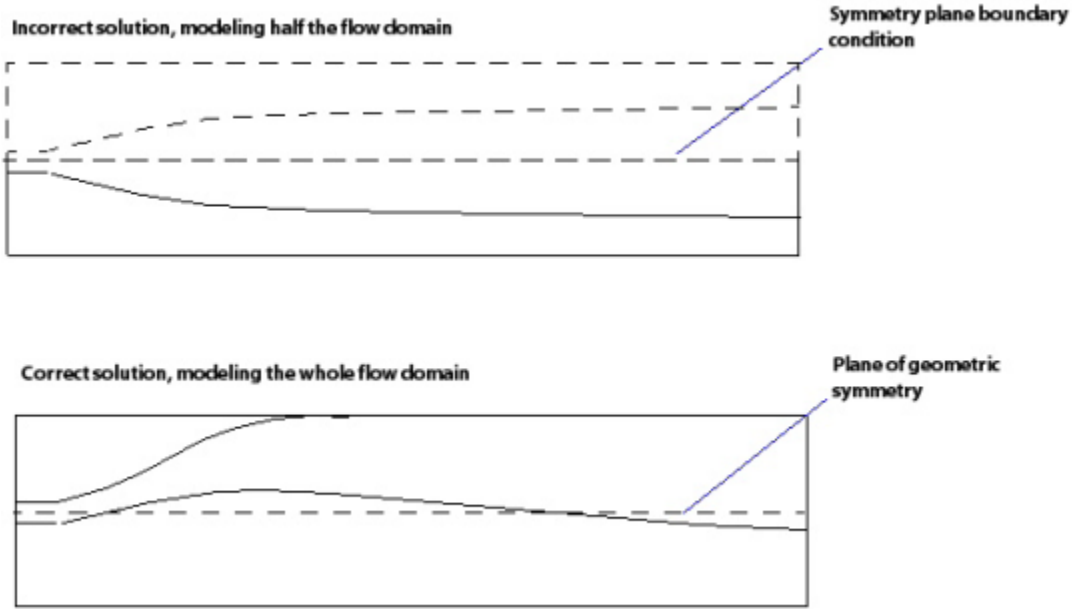
A particular case may have more than one plane of symmetry. For example, the flow in a square duct may exhibit symmetry about two independent planes (at 90° to one another). In this case, the physical problem has four-way symmetry. The computational domain can therefore be limited to one quarter of the duct, using two **Symmetry Plane** boundaries.

If you have two surfaces that meet at a sharp angle, and both are symmetry planes, you should set each surface to be a separate named boundary condition, rather than combine them into a single one.

When using an SMC turbulence model, you should position your model so that symmetry planes are perpendicular to coordinate axes if possible. This improves the performance of the CFX-Solver.

Symmetric geometry does not necessarily imply that the flow field is also symmetric. For example, a jet entering at the center of a symmetrical duct will tend to flow along one side above a certain Reynolds number. This is known as the Coanda effect. If a symmetry plane is used in this situation, an incorrect flow field will be obtained.

Figure 2.6: Symmetry Plane



A symmetry plane should only be used when you have reason to believe that the flow field is also symmetrical.

A description of the mathematical details of the Symmetry Plane boundary condition is available in Symmetry Plane in the *CFX-Solver Theory Guide*.

2.8.1. Mesh Motion

Stationary and unspecified boundary conditions are available for symmetry boundaries in domains where **Mesh Deformation** has been set to `Regions of Motion Specified`. (See Mesh Deformation in the *CFX-Pre User's Guide* for information about activating mesh deformation for the domain.) For details, see Mesh Deformation (p. 42).

2.9. Profile Boundary Conditions

It is possible to specify a boundary condition based on the interpolated values from a data file. This is useful to use the results of a previous simulation or experimental results as a boundary condition for the current simulation. CFX-Pre will generate CEL expressions that refer to the imported data, using interpolation functions. This data is automatically generated when creating a boundary condition using the Profile method.

To implement profile boundary conditions, follow these steps:

1. Create a boundary condition profile file. This can be accomplished by using the **Export** feature of CFD-Post.

   For details, see Type in the *CFD-Post User's Guide*.

   The data file format supports face and element definitions. Although not used in CFX-Pre and CFX-Solver, CFD-Post reads this data and displays values using the face/element information. A list of valid variable names that can be used in profile files is available. For details, see Standard Variable Names (p. 152).

   Alternatively, create your own data file (you may want to use an exported boundary condition profile file from CFD-Post as a template).

   For details, see Profile Data Format in the *CFX-Pre User's Guide*.

2. Initialize (read in) the profile in CFX-Pre.

   Do this by selecting **Tools** > **Initialize Profile Data** and selecting the profile file and data to load. You can load multiple profile files and each file can contain profile data for more than one locator.

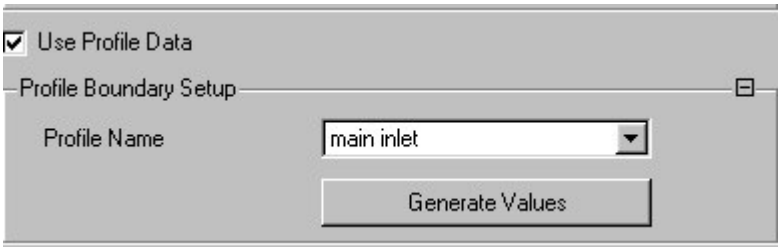
   For details, see Initialize Profile Data in the *CFX-Pre User's Guide*.

3. Optionally edit the profile function (which appears in the **Outline** tree view under `Simulation` > `Expressions, Functions and Variables` > `User Functions`) to associate value fields of the profile data with CEL parameters. This association facilitates the automatic configuration of boundary condition settings when profile data is used to create a boundary condition. For details, see Associating Profile Data with CEL Parameters in the *CFX-Pre User's Guide*.

4. Assign the profile data to a boundary condition.

   Do this by editing the boundary condition. On the **Basic Settings** tab, select **Use Profile Data** For details, see Use Profile Data in the *CFX-Pre User's Guide*.

Profile boundary conditions are always evaluated with reference to the local coordinate frame. Select the appropriate coordinate frame from the **Coordinate Frame** drop-down list. Additional information on coordinate frames is available in Coordinate Frames in the *CFX-Pre User's Guide*. For details, see Coordinate Frames (p. 76).



5. Select the appropriate profile from the drop-down list, and then click **Generate Values**. The **Boundary Details** tab will be modified to use the profile data. These changes to the **Boundary Details** tab will not be applied unless you click **Apply**. If you later modify profile values using CEL, and you would like to re-apply the profile data that came from the profile file, click **Generate Values** again.

The profile boundary condition (as well as other boundary conditions) can be visualized in CFX-Pre by using the **Plot Options** tab in the **Boundary Conditions** details view. You can create a **Boundary Contour** or a **Boundary Vector** plot of the profile data. For details, see Boundary Plot Options Tab in the *CFX-Pre User's Guide*.

The profile data is read into the CFX-Solver each time the solver is started/restarted (that is, the profile file can be edited between solver runs without returning to CFX-Pre).

2.9.1. Using a Profile From One Location at Another Location

There are a number of approaches that can be used to apply a profile from one location at another.

1. For locations that both have a surface normal vector of X, Y or Z, export the data as a 2D Profile (for example, for two boundaries with a normal in the Z direction, export X and Y profile data). The data from the first boundary can then be used at the second with no need for modifications to the data.

2. Create a custom coordinate frame so that the locations exported from one boundary will map correctly to a new boundary. Bear in mind that the coordinate frame you select will also be used by the solver to interpret the U, V, W Cartesian components you enter on the **Boundary Details** tab. Also bear in mind that any expression used to specify these quantities will be interpreted with x, y, z as being in the local coordinate frame, and all other variables as being in the global coordinate frame. For details, see Coordinate Frames in the *CFX-Pre User's Guide*. For details, see Coordinate Frames (p. 76).

3. Edit the data in your profile file directly to map the locations from the first boundary to the second.

4. Edit the arguments in CFX-Pre to the U, V, W expressions. As an example, the following modification is permitted in CFX-Pre: `myprofile.Velocity u(x,y,(z + 1[m]))`.

2.9.2. Standard Variable Names

The following table gives a list of valid variable names that can be used in profile files.

Variable	Variable Name(s) in Profile File
Epsilon	ed, Turbulence Eddy Dissipation
Fixed Temperature	T, Temperature
Heat Flux in	Qwall, Wall Heat Flux
Heat Transfer Coefficient	htc, Wall Heat Transfer Coefficient
k	ke, Turbulence Kinetic Energy
Mass Fraction	mf, Mass Fraction
Normal Speed	Velocity
Outside Temperature	T, Temperature
Pressure Profile Shape	p, Pressure, ptot, Total Pressure, ptotstn, Total Pressure in Stn Frame
Relative Pressure	p, Pressure, ptot, Total Pressure, ptotstn, Total Pressure in Stn Frame
Relative Static Pressure	p, Pressure
Static Temperature	T, Temperature
Stationary Frame Total Temperature	T, Temperature, Ttot, Total Temperature, Ttotstn, Ttotal Temperature in Stn Frame
Total Temperature	T, Temperature, Ttot, Total Temperature, Ttotstn, Total Temperature in Stn Frame
U	u, Velocity u, Velocity in Stn Frame u
Unit Vector X Component	veldir u, Velocity Direction u
Unit Vector Y Component	veldir v, Velocity Direction v
Unit Vector Z Component	veldir z, Velocity Direction z
V	v, Velocity v, Velocity in Stn Frame v
Velocity Axial Component	velaxial, Velocity Axial, w, Velocity w, velstn w, Velocity in Stn Frame w
Velocity r Component	velradial, Velocity Radial, u, Velocity u, velstn u, Velocity in Stn Frame u
Velocity Theta Component	velcirc, Velocity Circumferential, v, Velocity v
Volume Fraction	vf, Volume Fraction
W	w, Velocity w, Velocity in Stn Frame w
Wall U	u, Velocity u, Velocity in Stn Frame u
Wall V	v, Velocity v, Velocity in Stn Frame v
Wall W	w, Velocity w, Velocity in Stn Frame w
Wall Velocity r Component	u, Velocity u, Velocity in Stn Frame u
Wall Velocity Theta Component	v, Velocity v, Velocity in Stn Frame v

Variable	Variable Name(s) in Profile File
Wall Velocity Axial Component	w, Velocity w, Velocity in Stn Frame w

Some variables require a prefix to include the material name (for example, air.vf corresponds to the volume fraction of air). For details, see System Variable Prefixes in the *CFX Reference Guide*.

2.9.3. Non-Standard Variable Names

If you have non-standard variable names in your profile file, they can still be used in CFX-Pre. Such names must be typed in as an expression directly in CFX-Pre, because they will not be picked up automatically. For example, your file may contain the variable `uvel` for velocity in the U direction. On the **Boundary Details** tab, you could enter `<profile>.uvel(x,y,z)` (where `<profile>` is the name of your profile) to pick up your `uvel` values.

2.9.4. Custom Variables

You may also type your own expressions to use non-standard interpolation variables, as below:

`<prof name>.<var name>`(x,y,z)

2.9.5. Using r-Theta Profiles

Using r-theta profiles in Ansys CFX is possible, but the following points should be noted to ensure that the profiles are set up as intended.

1. The r-theta coordinates specified in the profile are used to calculate X and Y positions in the Cartesian space. The local coordinate frame (selected on the **Basic Settings** tab for each boundary condition in CFX-Pre) is used to evaluate the X and Y directions. When the r-theta and X-Y coordinates are defined with respect to the same coordinate frame, the points (r=1, theta=0) correspond to (X=1, Y=0). For details, see Coordinate Frames (p. 76).

2. When specifying cylindrical velocity components, a rotation axis is used on the **Boundary Details** tab to specify the velocity component directions. Note that this axis is independent of the axis used to evaluate the r-theta coordinate positions specified in the profile file.

3. Profiles specified using r-theta positions and U, V, W velocity components may not be usable unless:

   - The U and V velocity component directions correspond with the X-Y coordinate directions that will be mapped from your r-theta coordinates by CFX-Pre (outlined in point 1, above).

   - You understand that using a custom coordinate frame to achieve the correct X-Y mapping from your r-t theta positions will also change the U, V, W velocity component directions, which you enter on the **Boundary Details** tab (and this may not be what you intend).

2.9.6. Data Interpolation Method

For 1D discrete profiles, the topology of the data can be determined by ordering the raw data based on the given single spatial coordinate. Linear interpolation is performed between the ordered raw data points. The data is sorted so that the order of specification is not important.

For 2D and 3D discrete profiles, a cloud of points algorithm is used to perform the interpolation. The process involves a fast lookup of the three nearest raw data points to the evaluation point, and then application of an inverse distance weighted averaging procedure. If a raw data point lies precisely at the evaluation point location, the raw data value at that point will be used.

Note:

NOTE: For 2D and 3D discrete profiles that have fewer than three data points defined, only those data points will be used in the nearest point search. For profiles with two data points, the inverse distance weighted averaging procedure is still applied, although the results could appear misleading in regions away from those points. For profiles with just one data point, the exact data value from that point will be returned with no spatial variation.

During the solution process, the solver requires values at various locations. For example, at integration points, nodes and face center locations, as required by the specifics of the discretization and numerical integration process. In all instances, the required location is determined and the raw data is interpolated to that location.

2.9.7. Extracting Profile Data from Results Files

When profiles have been read into the CFX-Solver, a summary of the files is written to the CFX-Solver Output file under the section **Profile Data**. All profile data is written to the results file and can be extracted using the command line utility `cfx5dfile`. Depending on your preference setting for internal or external CSV data storage, profile data might be read into the solver each time the solver is started/restarted, in which case the results file contains the profile data that was most recent at the time.

`cfx5dfile` can be run from the command line and is located in the *<CFXROOT>*`/bin/` directory.

The remainder of this section describes the `cfx5dfile` command. You can also get help from the command line by using the `-help` switch.

To find out which profile data is stored in the current results file, enter:

```
cfx5dfile <results file name> -list-profile-data
```

This outputs a list of all profile data stored in or referenced by the results file, one profile per line. The data for any of the listed profiles can be extracted with the `-read-profile-data` switch.

Enter:

```
cfx5dfile <results file name> -read-profile-data -function <function name>
```

to print the profile data corresponding to the named function to standard output (for example, your terminal window), or to the file specified with the `-output` switch.

Enter:

```
cfx5dfile <results file name> -extract-profile-data -function <function name>
```

to write the profile data corresponding to the named function to the current directory to a file with an automatically-determined file name. The file will be named according to the function name (for

internally-stored profile data) or based on the original file name (for external profile data). If this file already exists in the current directory, it will not be overwritten. The function name is the CCL object name.

This is equivalent to using `-read-profile-data -function <function name>`, and using `-output` to set the destination file name.

Enter:

```
cfx5dfile <results file name> -write-profile-data -function <function name> <filename>
```

to insert the contents of the specified file into the results file as profile data to be used for the specified function. Note that this profile data is used when restarting a run from the results file.

For any file referenced in the results file, enter:

```
cfx5dfile <results file name> -read-profile-data -filename <filename>
```

to print the profile data that was originally initialized from the CSV file `<filename>` to standard output (for example, your terminal window), or to the file specified with the `-output` switch. This is only applicable in this form for externally-stored profile data. For internal CSV data, the `-filename` argument behaves the same way as the `-function` argument.

Alternatively, enter:

```
cfx5dfile <results file name> -extract-profile-data -filename <filename>
```

to write the profile data that was originally initialized from the CSV file `<filename>` to the current directory in a file of the same name. This is equivalent to using `-read-profile-data -filename <filename>`, and using `-output` to set the destination file name. This is only applicable in this form for externally-stored profile data. For internal CSV data, the `-filename` argument behaves the same way as the `-function` argument.

Enter:

```
cfx5dfile <results file name> -extract-all-profile-data
```

to save all the profile data contained in the results file to a set of files in the current directory. If any of the files exist locally, they will not be overwritten.

Note:

If you use `cfx5cmds` to update CCL, and/or you use `cfx5dfile` to update the profile data or table data stored within the case, then you must, in addition to updating an `mdef` file, also perform the same update to any associated `.def` or `.cfg` files.

2.10. General Non-Reflecting Boundary Conditions

Information about general NRBCs is provided in the following sections.

2.10.1. Overview

2.10.2. Restrictions and Limitations

2.10.3. Theory
2.10.4. Acoustic Reflectivity Settings in CFX-Pre

2.10.1. Overview

The general acoustic reflectivity option for boundary conditions in Ansys CFX is based on characteristic wave relations derived from the Euler equations. The non-reflecting boundary conditions are available only for subsonic flow regimes on inlet, outlet and opening boundary types. The reformulated Euler equations are solved on the boundary of the domain in an algorithm similar to the flow equations applied to the interior of the domain.

2.10.2. Restrictions and Limitations

- The acoustic reflectivity boundary condition option is available only for transient and transient blade row analyses of compressible single phase flow.
- The flow regime cannot become supersonic at a non-reflecting boundary at any time during the simulation.

2.10.3. Theory

The acoustic reflective boundary conditions available in Ansys CFX are derived by first recasting the Euler equations in a local orthogonal coordinate system (x_1, x_2, x_3) such that one of the coordinates, x_1, is normal to the boundary Figure 2.7: The Local Orthogonal Coordinate System onto which Euler Equations are Recasted for the General NRBC Method (p. 157).

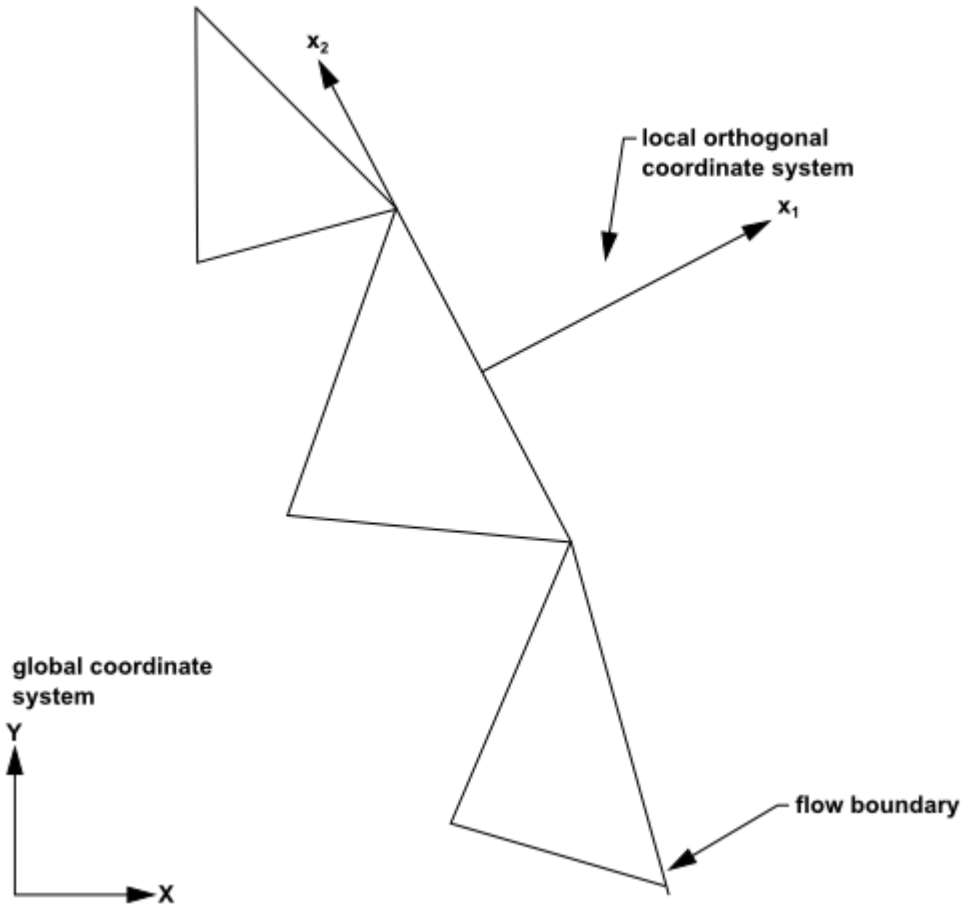
$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{1}{c^2}\left[L_2 + \frac{1}{2}\left(L_5 + L_1\right)\right] &= 0 \\
\frac{\partial \rho}{\partial t} + \frac{1}{2}\left(L_5 + L_1\right) &= 0 \\
\frac{\partial u_1}{\partial t} + \frac{1}{2\rho c}\left(L_5 - L_1\right) &= 0 \\
\frac{\partial u_2}{\partial t} + L_3 &= 0 \\
\frac{\partial u_3}{\partial t} + L_4 &= 0
\end{aligned} \tag{2.12}$$

Where L_i's are the incoming and outgoing characteristic wave amplitudes.

Important:

Note that a transformation between the local orthogonal coordinate system (x_1, x_2, x_3) and the global Cartesian system (X, Y, Z) must be defined on each face on the boundary to obtain the velocity components (u, v, w) in a global Cartesian system.

Figure 2.7: The Local Orthogonal Coordinate System onto which Euler Equations are Recasted for the General NRBC Method



From characteristic analyses, the wave amplitudes, L_i, are given by:

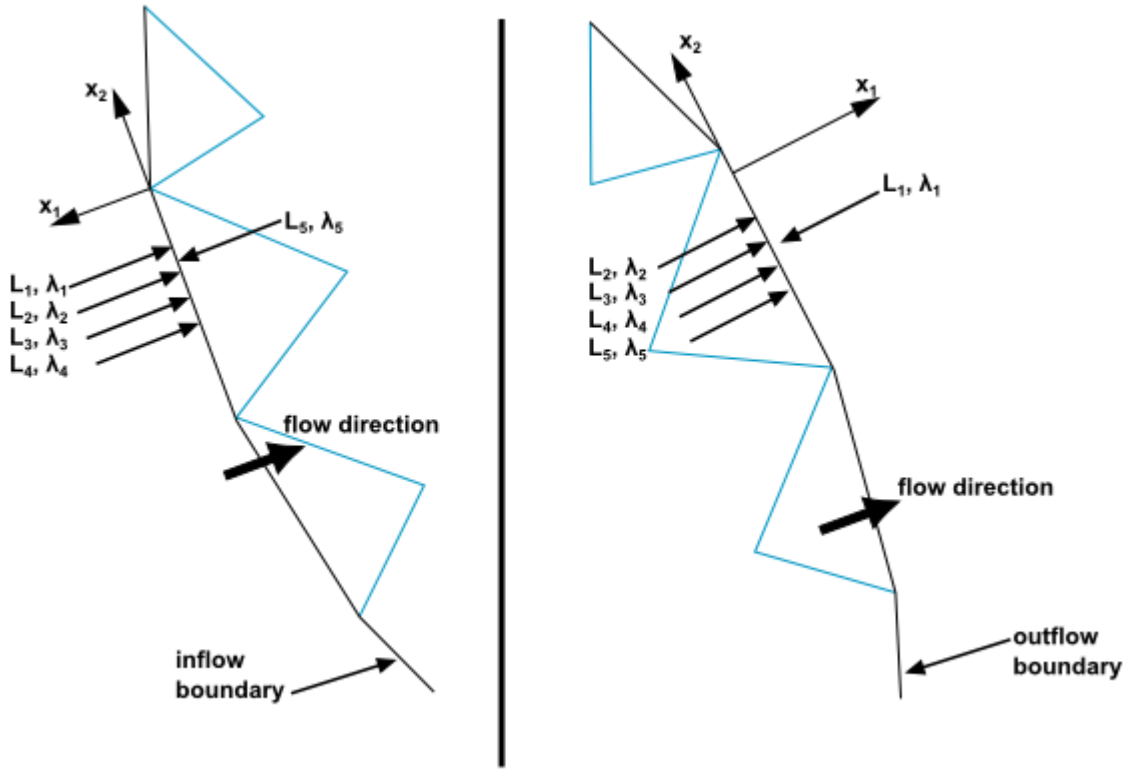
$$
\begin{aligned}
L_1 &= \lambda_1\left(\frac{\partial P}{\partial x_1} - \rho c \frac{\partial U_1}{\partial x_1}\right) \\
L_2 &= \lambda_2\left(c^2 \frac{\partial \rho}{\partial x_1} - \frac{\partial P}{\partial x_1}\right) \\
L_3 &= \lambda_3 \frac{\partial U_2}{\partial x_1} \\
L_4 &= \lambda_4 \frac{\partial U_3}{\partial x_1} \\
L_5 &= \lambda_5\left(\frac{\partial P}{\partial x_1} + \rho c \frac{\partial U_1}{\partial x_1}\right)
\end{aligned}
\tag{2.13}
$$

The equations above are solved on non-reflecting boundaries, along with the interior governing flow equations, using similar time stepping algorithms to obtain the values of the primitive flow variables.

The outgoing and incoming characteristic waves are associated with the characteristic velocities of the system (i.e eigenvalues), λ_i, as seen in Figure 2.8: Waves Leaving and Entering a Boundary Face on Inflow and Outflow Boundaries. The Wave Amplitudes are Shown with the Associated Eigenvalues for a Subsonic Flow Condition (p. 158). These eigenvalues are given by:

$$\lambda_1 = U_1 - c$$
$$\lambda_2 = \lambda_3 = \lambda_4 = U_1 \tag{2.14}$$
$$\lambda_5 = U_1 + c$$

Figure 2.8: Waves Leaving and Entering a Boundary Face on Inflow and Outflow Boundaries. The Wave Amplitudes are Shown with the Associated Eigenvalues for a Subsonic Flow Condition



For subsonic flow leaving a boundary, four waves leave the domain (associated with positive eigenvalues λ_2, λ_3, λ_4, and λ_5) and one enters the domain (associated with negative eigenvalue λ_1). For subsonic flow entering a boundary, four waves enter the domain (associated with the negative eigenvalues λ_1, λ_2, λ_3, λ_4) and one leaves the domain (associated with a positive eigenvalue λ_5).

To solve Equation 2.12 (p. 156) on a boundary, the amplitudes of the incoming and outgoing waves must first be determined. The amplitudes of outgoing waves are computed from Equation 2.13 (p. 157) by using values of flow derivatives $\frac{\partial P}{\partial x_1}$, $\frac{\partial \rho}{\partial x_1}$, $\frac{\partial U_1}{\partial x_1}$, $\frac{\partial U_2}{\partial x_1}$, and $\frac{\partial U_3}{\partial x_1}$ from inside the domain. The amplitudes of the incoming waves are computed as follows. The amplitudes of the incoming waves are computed from the Linear Relaxation Method (LRM) of Poinsot 107. The LRM method sets the value of the incoming wave amplitude to be proportional to the differences between the local primitive variable on a boundary and the imposed boundary value.

The reflection relaxation factor, K is given by:

$$K = \sigma_1 \left(1 - M_{\max}{}^2\right) \frac{c}{L} \tag{2.15}$$

where c is the acoustic speed, L is the characteristic length, $M_{\max}$ is a representative Mach number scale in the domain (default is 0), and σ_1 is the under-relaxation factor (default value is 0.25).

The expression for the incoming amplitudes depends on the boundary condition type as shown below:

1. **Static Pressure Specified**

$$L_1 = K(P - P_{\text{specified}}) \tag{2.16}$$

where $P_{\text{specified}}$ is the imposed pressure at the boundary.

2. **Total Pressure Specified**:

$$L_1 = K(P - P_{\text{specified}}) \tag{2.17}$$

where the specified static pressure, $P_{\text{specified}}$ is computed from the specified Total Pressure.

3. **Mass Flow Specified**:

$$L_1 = -K\rho c(U_1 - U_{1,\text{specified}}) \tag{2.18}$$

where the imposed specified velocity and density are computed from specific mass flux and temperature.

4. **Velocity Inlet Specified**:

$$L_1 = -K\rho c(U_1 - U_{1,\text{specified}}) \tag{2.19}$$

where the imposed velocity, $U_{1,\text{specified}}$, is specified at the boundary and the density is computed from the specified boundary temperature and pressure.

2.10.4. Acoustic Reflectivity Settings in CFX-Pre

The **Acoustic Reflecticity** settings are available for some subsonic inlets, subsonic outlets, and openings.

With reference to Theory (p. 156):

- **Reflection Factor** is K.
- **Reflection Length Scale** is L.
- **Reflection Mach Scale** is $M_{\max}$.

2.11. Limitations

- For specified flux or heat transfer coefficient boundary conditions, the boundary solution data at integration points corresponds to conservative values instead of hybrid values. As a result, postprocessing evaluations of flux integrals and gradients of the solution variable over such boundaries may be less accurate. This limitation does not affect solution data on nodes.

  The most affected cases are those that involve specified flux or heat transfer coefficient boundaries having any of the following features:

  – Heat transfer for laminar flows

  – Heat transfer for solid domains

  – Additional Variables

– Electric potential

Chapter 3: Initial Condition Modeling

Initial values for all solved variables need to be set before the solution can begin.

If your calculation is a steady-state calculation, the initial variable values serve to give the CFX-Solver a flow field from which to start its calculations. Convergence is more rapidly achieved if sensible initial values are provided. However, converged results should not be affected by the initialization.

If your calculation is a transient (unsteady) calculation, then initial values provide the flow field at the time when the CFD calculation starts. The values specified should be the actual flow field present at the beginning of the time of the simulation.

There are three options available for setting the initial values:

- automatically generated default values
- explicitly specified values
- read from an existing results file.

The first two options are specified by setting initialization parameters for the analyses (global) or domains in CFX-Pre. For details, see Initialization in the *CFX-Pre User's Guide* and Initialization Tab in the *CFX-Pre User's Guide*, respectively). If you want to read initial conditions from an initial values file (an existing results file from a previous run) then this is done on the **Run Definition** tab for a simulation execution control or a configuration in CFX-Pre or the **Run Definition** tab of the **Define Run** dialog box in the CFX-Solver Manager. For details, see Run Definition Tab in the *CFX-Pre User's Guide*, Run Definition Tab in the *CFX-Pre User's Guide* and Run Definition Tab in the *CFX-Solver Manager User's Guide*, respectively. If you chose to specify an initial values file then the initial conditions read from that file will override any settings made in CFX-Pre.

Automatically generated default values are generally only recommended for steady-state runs. They are specified by choosing the `Automatic` option on the **Initialization** tabs in CFX-Pre. In this case, the CFX-Solver will calculate its own initial values, based on the physics and boundary conditions in the set-up. Details of how the initial values are calculated can be found in Initialization Parameters (p. 163). You can set the initial conditions explicitly by choosing the `Automatic with Value` option on the **Initialization** tabs in CFX-Pre. Both of these options are discussed in Setting the Initial Conditions in CFX-Pre (p. 162).

Reading results from an existing results file is useful in many cases, including:

- To restart a steady-state run after a physics change.

  When restarting a run with, for example, a new turbulence model, save time and resources by using a set of results from a previous run as the basis for an initial guess (even if the run was only partially converged). These initial guess values are likely to be far more accurate than using an automatic initial guess.

- To begin a transient analysis, using the results of a steady-state analysis as the initial conditions.

  Transient simulations require values to be set for all variables during initialization. Starting a transient run with results from a converged steady-state analysis can lead to good convergence and solution robustness, especially if it is difficult to provide an accurate estimate of the initial conditions.

- To continue a transient run with a new mesh.

  A transient moving mesh case might need to be stopped and restarted with a new mesh, if the mesh quality is poor. The run with the new mesh should continue using the values calculated by the solver when the run with the old mesh was stopped.

In addition, reading results from an existing results file is done implicitly whenever one configuration from a multi-configuration run is set to use initial values from a previous configuration (where the solver will use the results file produced by the previous configuration as its initial values file), and when a run is interrupted and restarted using the automatic remeshing capability. For details, see Remeshing Tab and Remeshing Guide.

The process and limitations for initializing a run with values from a previous results file is described in Reading the Initial Conditions from a File (p. 174).

This chapter describes:

- Setting the Initial Conditions in CFX-Pre (p. 162)
- Initialization Parameters (p. 163)
- Reading the Initial Conditions from a File (p. 174)
- Using the CFX-Interpolator (p. 182)

3.1. Setting the Initial Conditions in CFX-Pre

In CFX-Pre, you can set the initialization option for each solved variable to either the `Automatic` or `Automatic with Value`. Only solved (or principal) variables are initialized; if an initial field is required for other variables, it is derived from the solved variable initial fields.

You can set initial conditions on a per-domain basis (on the Initialization Tab in the *CFX-Pre User's Guide* for the domain when you check the **Enable Initial Conditions** toggle on the domain's **Basic Settings** tab), or globally (from the *Global Initialization*  icon).

Global initialization options apply to all domains in the simulation. However, domain initialization will take precedence over global initialization for any domain in which it is specified.

For details on initialization, see Initialization in the *CFX-Pre User's Guide*.

3.1.1. Automatic

Selecting the `Automatic` option allows the solver to automatically read the initial conditions from an initial values file if it can find one, or else use a solver default initialization field if no initial values file is provided. If the default initialization field is used, the CFX-Solver will calculate values for the solved variable at the start of the solution. You can select an initial values file for the CFX-Solver to

use from the CFX-Solver Manager or from the **Run Definition** tab for a simulation execution control or a configuration in CFX-Pre. For details, see Reading the Initial Conditions from a File (p. 174).

Choosing the `Automatic` option without providing an initial values file is only recommended for steady-state runs. For transient runs, you should provide initial conditions that correspond to the real flow field at the start of the calculation, and the CFX-Solver will stop by default if it has to use a solver default initialization field for a transient run for any quantities except turbulence-related variables.

If the automatic initial guess fails, it is often because an inappropriate initial velocity or static pressure field exists. Poor initial guesses are one of the most common reasons for the CFX-Solver to fail. If you use `Automatic` as the initial guess method for velocity, you must understand how the initial guess field is generated in CFX-Solver; this is described below. You should think about an initial guess and use the `Automatic with Value` option to set a constant value for velocity and pressure if the `Automatic` option is inappropriate. Better still, use CEL expressions or Cloud Interpolation to initialize the velocity and pressure fields.

An automatic initial guess is always acceptable for incompressible laminar cases if a sensible velocity field is produced, as described below. It is sometimes acceptable for incompressible turbulent cases, but the `Automatic with Value` method is often better.

Additional information on the way in which the `Automatic` option calculates values, and on sensible values for initial guesses, is available in Initialization Parameters (p. 163).

3.1.2. Automatic with Value

Selecting the `Automatic with Value` option allows the CFX-Solver to automatically read the initial conditions from an initial values file if it can find one, or else use the specified value or expression. This option is often better than using the `Automatic` method (assuming a sensible value is set), and is almost always required for compressible flow cases.

3.1.3. Using Expressions with Initial Conditions

The initial conditions can be defined using any valid CEL expression. However, in general, to avoid infinite recursion or problems with initialization on parallel runs, the expression may only contain the CFX System Variables `x`, `y`, `z`, `r`, and `theta`.

For details, see CFX Expression Language (CEL) in the *CFX Reference Guide*.

3.2. Initialization Parameters

The topics in this section include:

- Coordinate Frame (p. 164)
- Frame Type (p. 165)
- Velocity Type (p. 165)
- Cartesian Velocity Components (p. 166)
- Cylindrical Velocity Components (p. 168)

- Velocity Scale (p. 168)
- Velocity Fluctuation (p. 168)
- Static Pressure (p. 169)
- Temperature (p. 169)
- K (Turbulent Kinetic Energy) (p. 169)
- Epsilon (Turbulence Eddy Dissipation) (p. 170)
- Omega (Turbulence Eddy Frequency) (p. 171)
- Reynolds Stress Components (p. 171)
- Initialization of Additional Variables (p. 172)
- Component (p. 172)
- Volume Fraction (p. 172)
- Radiation Intensity (p. 172)
- Initialization of Solid Domains (p. 172)
- Initial Conditions for a Multiphase Simulation (p. 173)
- Initialization Advice (p. 173)

3.2.1. Coordinate Frame

When using domain initialization, you can optionally select a reference coordinate frame other than the default `Coord 0` frame in which to have the solver interpret your initialization values. If a coordinate frame is not specified, the domain coordinate frame is used (although it may be overridden, as described below). Any variables used in the specification of an initialization value are interpreted as follows:

- x, y, z

  local coordinate frame

- other variables

  global coordinate frame

You can select any predefined coordinate frame.

Information about making local coordinate frames is available in Coordinate Frames in the *CFX-Pre User's Guide*.

The coordinate frame used is local to initialization only and is used to interpret all x, y and z component values set on the **Initialization** panel.

Note:

The coordinate frame set here may be ignored in some situations. For details, see Velocity Type (p. 165).

3.2.2. Frame Type

If a domain is rotating, you can choose to apply the initialization values relative to each domain (**Frame Type** = `Rotating`) or to the absolute frame of reference (**Frame Type** = `Stationary`). If the **Frame Type** setting is not given, the initialization values apply relative to each domain. For stationary domains, the **Frame Type** setting has no effect, and initialization values apply to the absolute frame of reference.

3.2.2.1. Considerations for Multiple Domains and Global Initialization

When using the **Global Initialization** dialog box to set properties for **Frame Type**, the settings acquired by each domain in a multiple domain simulation are as follows:

- If one domain is rotating and the other is stationary, selecting a global rotating frame type will specify relative frame types for both domains. The rotating frame will therefore be initialized with its relative (rotating) frame and the stationary domain will be initialized with its relative (stationary) frame.
- If one domain is rotating and the other is stationary, selecting a global stationary frame type will specify initialization for both domains with an absolute (stationary) frame.

3.2.3. Velocity Type

The velocity type can be set to `Cartesian` or `Cylindrical`: There are some important considerations to note with respect to the way that coordinate frame types affect velocity types:

3.2.3.1. Cartesian Coordinate Frame, Cartesian Velocity Components

The `Cartesian Velocity Components` are applied relative to the Cartesian coordinate frame selected for initialization.

3.2.3.2. Cartesian Coordinate Frame, Cylindrical Velocity Components

A local axis is required, which overrides the definition of the Cartesian coordinate frame chosen for initialization. The location of theta = 0 is not important, so only the axis is required to define the cylindrical coordinate frame.

3.2.3.3. Cylindrical Coordinate Frame, Cartesian Velocity Components

The specified Cartesian velocity components X, Y and Z are applied in the cylindrical coordinate frame directions for r, theta and Z, respectively.

3.2.3.4. Cylindrical Coordinate Frame, Cylindrical Velocity Components

A local axis is required, which overrides the definition of the cylindrical coordinate frame chosen for initialization. The location of theta = 0 is not important so only the axis is required to define the cylindrical coordinate frame.

3.2.4. Cartesian Velocity Components

3.2.4.1. Automatic Values

For **Automatic Values**, the initial conditions are based on a linear evaluation. Evaluation of linear initial conditions requires two points in the solution domain and initial condition values at these points. These points and initial condition values are generated using a weighted average of boundary condition information from all inlets, openings and outlets. The two points may be considered as a globally representative inlet and outlet. For example, the first point (the globally representative inlet) is located at an average of the centroids of all inlet, opening and outlet boundaries where the inlet boundary information is weighted much more heavily than the information from other boundaries. The initial condition value(s) for this point are similarly evaluated. See Figure 3.1: Initial Condition Values (p. 167) for an illustration.

The direction and magnitude of velocity linear initial conditions are treated independently. This reflects the use of velocity-specified (the direction and magnitude specified) boundary conditions, and of direction-specified (the total pressure and normal direction) boundary conditions.

When direction information is available from boundary conditions, it is used. If no information is available, a boundary normal direction is assumed. Note that zero vectors are possible for the globally representative inlet and outlet direction vectors because of the direction averaging (for example, two similar inlets that face each other).

Magnitude information is similarly used when it is available from the boundary conditions (note that an approximate velocity is calculated for mass flow specified boundary conditions). If no velocity magnitude information is available from boundary conditions (such as total pressure inlet, static pressure outlet), then half of the average rotational velocity (that is, the angular velocity multiplied by the mean radius) over a given boundary is used. This gives a reasonable "velocity triangle" approximation for turbo machines, and results in a magnitude of zero for stationary domains.

Note that the linear initial condition reverts to a constant velocity magnitude (or pressure) if only one value can be derived from boundary condition information (for example, constant pressure for total pressure in, mass flow out).

Important:

When applying initial conditions, the shape of your domain(s) is an important consideration when applying an automatic initial guess. As Figure 3.2: Inaccurate Initial Conditions (p. 167) shows, a linearly varying prediction for velocity is not always appropriate. You can disable the automatic initial guess for velocity by setting the velocity scale to 0 m/s. For details, see Velocity Scale (p. 168).

Figure 3.1: Initial Condition Values

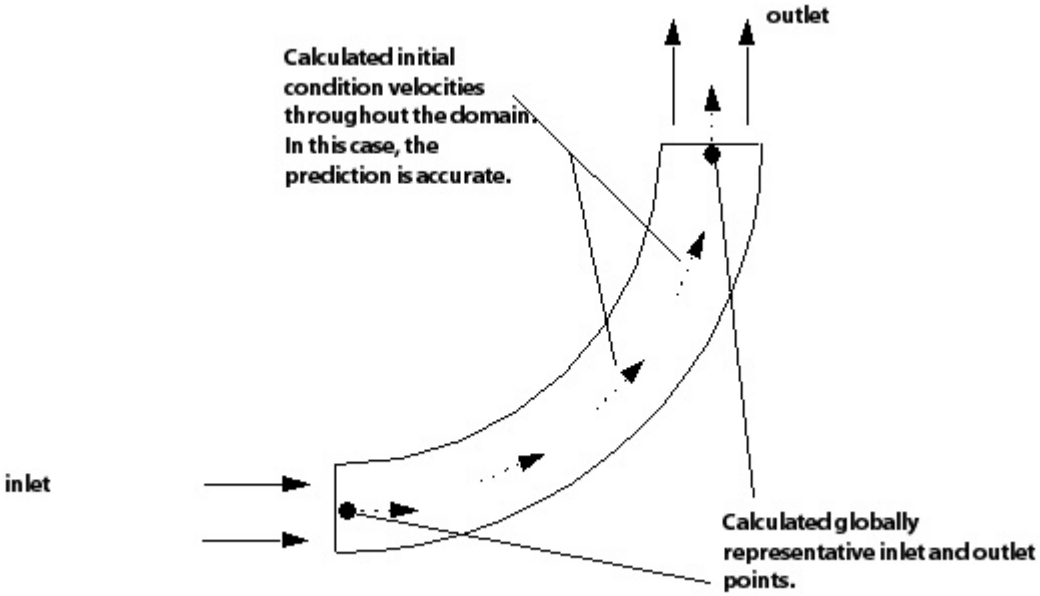
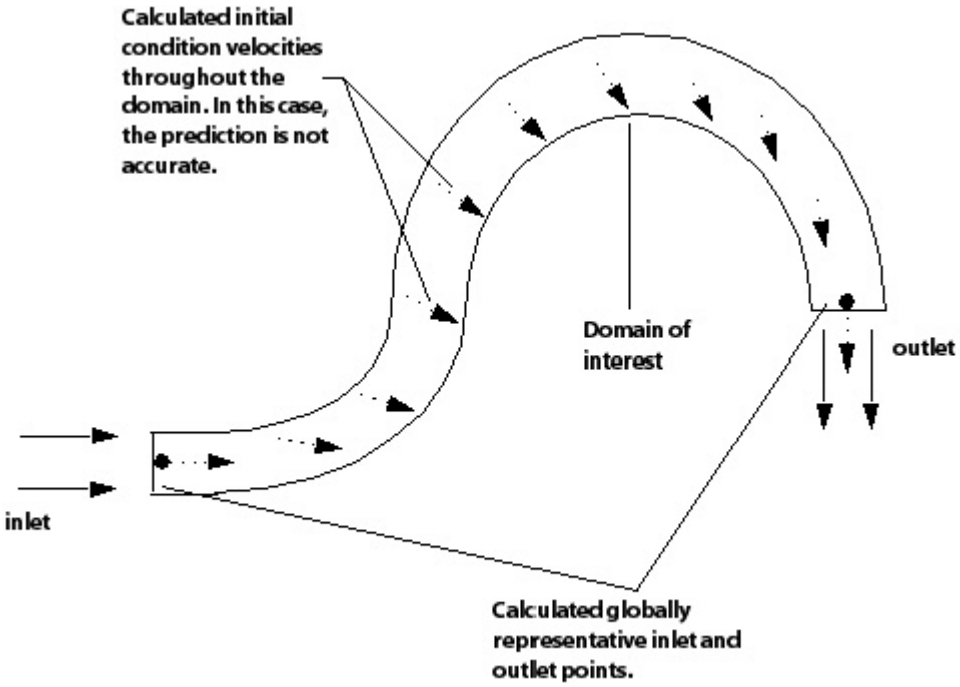


Figure 3.2: Inaccurate Initial Conditions



3.2.4.2. Recommended Values

If you suspect that the flow is predominantly in a particular direction through the domain (for example, in the direction of an Inlet specification), you can initialize the velocity components to this direction. You should not set a velocity higher than the inlet velocity.

If you predict that the `Automatic` method is unsuitable for your simulation, but do not want to specify Cartesian components, setting the velocity scale to zero imposes a zero velocity field throughout the domain. For details, see Velocity Scale (p. 168).

An explanation of how coordinate frames interact with velocity components is available in Coordinate Frame (p. 164).

Information on modeling multiphase flow is available in Initial Conditions for a Multiphase Simulation (p. 173).

3.2.5. Cylindrical Velocity Components

3.2.5.1. Automatic Values

The initial guess for velocity is calculated in the same way as for Cartesian velocity components. For details, see Automatic Values (p. 166).

3.2.5.2. Recommended Values

If you suspect that the flow is predominantly in a particular direction through the domain (for example, in the direction of an Inlet specification), you can initialize the velocity components to this direction. You should not set a velocity higher than the inlet velocity.

If you believe that the automatic prediction will be unsuitable for your simulation, but do not want to specify Cartesian components, setting the velocity scale to zero imposes a zero pressure and velocity field throughout the domain. For details, see Velocity Scale (p. 168).

An explanation of how coordinate frames interact with velocity components is available in Coordinate Frame (p. 164).

Information on modeling multiphase flow is available in Initial Conditions for a Multiphase Simulation (p. 173).

3.2.6. Velocity Scale

In some cases, the automatically generated initial guess for the velocity magnitude and/or direction may not be appropriate. For example, for large domains with small, high-speed inlets, the initial guess for velocity may be too high. You can optionally enforce a velocity scale (which is essentially the velocity magnitude used throughout the domain). The calculated velocity directions are unchanged. In particular, applying this velocity scale to zero direction vectors will still yield a zero velocity vector. For details, see Automatic Values (p. 166).

A zero velocity field (that is, a quiescent velocity field) results if the velocity scale is set to `0`.

3.2.7. Velocity Fluctuation

`Velocity Fluctuation` is an optional parameter that can be entered for large eddy simulations. For details, see The Large Eddy Simulation Model (LES) (p. 213).

3.2.8. Static Pressure

3.2.8.1. Automatic Values

The location of two globally representative inlet and outlet points is determined in the same way as for `Cartesian Velocity Components`. For details, see Automatic Values (p. 166).

In order to avoid the creation of walls at domain inlets and outlets, the globally representative inlet and outlet pressure values are respectively decreased and increased by 10% of the range of values. For example, globally representative inlet and outlet values of 100 kPa and 90 kPa would be updated to 91 kPa and 99 kPa, respectively.

3.2.8.2. Recommended Values

The initial condition for the pressure field should be the average of the highest value of pressure specified on any of the outlet boundaries and the lowest value of pressure specified on any of the inlet boundaries. This reduces the likelihood of spurious inflow at outlets, or outflow at inlets, during the course of the solution.

For a free surface calculation, you should set a hydrostatic pressure distribution in the heavy fluid, and a constant pressure equal to the free surface value in the light fluid. For details, see Modeling Advice for Free Surface Flow (p. 348).

3.2.9. Temperature

3.2.9.1. Automatic Values

For an isothermal simulation, the temperature field is set to the specified isothermal value.

For other heat transfer models, the temperature field is calculated from an area weighted average of the boundary temperatures (inlets, outlets, openings and walls). If you have used a function or expression to specify the boundary temperature, an area weighted average of the expression over the boundary will be used to calculate an average value for that boundary.

3.2.9.2. Recommended Values

A sensible initial guess for the temperature field is an average of the boundary condition temperatures.

3.2.10. K (Turbulent Kinetic Energy)

3.2.10.1. Automatic Values

The default turbulent kinetic energy is calculated as:

$$k=\frac{3}{2}\left[I_{def}\ \max\left(U_s,\left|\boldsymbol{U}_{IG}\right|,U_\omega\right)\right]^2 \tag{3.1}$$

where:

I_{def} is the default turbulent intensity of 5%.

U_s is a minimum clipping velocity of 0.01 m/s to avoid a result of zero for the turbulent kinetic energy when an initial velocity value of zero exists.

$\boldsymbol{U}_{IG}$ is the velocity initial guess.

U_ω is the product of the simulation average length scale and the rotation rate. This term is designed to produce a suitable velocity scale for rotating domains.

3.2.10.2. Recommended Values

Unless you have information about the turbulent kinetic energy for your application, use the `Automatic` initial guess.

3.2.11. Epsilon (Turbulence Eddy Dissipation)

3.2.11.1. Automatic Values

The default turbulence eddy dissipation is calculated as:

$$\varepsilon = \frac{C_\mu k^2}{\nu \cdot \left(\mu_T / \mu\right)} \tag{3.2}$$

where:

C_μ is a non-dimensional constant

k is the turbulent kinetic energy

ν is the kinematic viscosity

μ_T / μ is the eddy viscosity ratio, which is set to 10 by default

3.2.11.2. Recommended Values

Unless you have information about the turbulence eddy dissipation for your application, use the `Automatic` initial guess.

3.2.11.3. Manual Specification

Turbulence eddy dissipation can be specified as a function of:

- `Omega` (that is, turbulence eddy frequency)
- `Eddy Length Scale`

$$\varepsilon = \frac{k^{3/2}}{l} \tag{3.3}$$

- `Eddy Viscosity Ratio`

$$\varepsilon = \frac{C_\mu k^2}{\nu \cdot \left(\mu_T / \mu \right)} \tag{3.4}$$

Turbulence eddy dissipation can also be specified directly as a CEL expression.

3.2.12. Omega (Turbulence Eddy Frequency)

3.2.12.1. Automatic Values

The default turbulence eddy frequency is calculated as:

$$\omega = \frac{k}{\nu \cdot \left(\mu_T / \mu \right)} \tag{3.5}$$

where:

C_μ is a non-dimensional constant

k is the turbulent kinetic energy

ν is the kinematic viscosity

μ_T / μ is the eddy viscosity ratio, which is set to 10 by default

3.2.12.2. Recommended Values

Unless you have information about the turbulent frequency for your application, use the `Automatic` initial guess (which is to use an `Eddy Viscosity Ratio` of `10`).

3.2.12.3. Manual Specification

Turbulence eddy frequency can be specified as a function of:

- `Epsilon` (that is, turbulence eddy dissipation)
- `Eddy Length Scale`

$$\omega = \frac{k^{3/2}}{k\, C_\mu\, l} \tag{3.6}$$

- `Eddy Viscosity Ratio`

$$\omega = \frac{k}{\nu \cdot \left(\mu_T / \mu \right)} \tag{3.7}$$

Turbulence eddy frequency can also be specified directly as a CEL expression.

3.2.13. Reynolds Stress Components

Unless you have information about the Reynolds stress components for your application, use the `Automatic` initial guess method.

3.2.14. Initialization of Additional Variables

A recommended guess for Additional Variable levels is to set them equal to their values at the inlet.

3.2.15. Component

When a fluid is a variable composition mixture, set the initial mass fractions for each of the components in the fluid. If modeling combustion, setting a mass fraction of 0.232 for Oxygen, and then setting all other components to `Automatic` is a good initial guess. For details, see Combustion Modeling (p. 425).

For non-combusting reactions, `Automatic` is a sensible initial guess method if you do not know the initial concentrations of components at the start of the simulation.

Note:

When restarting a calculation, you can rename a defined material component, or add (define) a new material component in the solver input file. For the renamed and/or added material components, there is no initial condition information stored in the initial values file; therefore, these component mass fractions are initialized using user-specified values and/or automatically generated default values.

Since a combination of initialization options may be used to initialize all of the component mass fractions, it is likely that the component mass fractions (including the `Constraint` mass fraction) will no longer sum to 1. If this is the case, the solver scales the defined mass fractions and/or recalculates the `Constraint` mass fraction to enforce a component mass fraction sum of 1. The solver also issues a warning explaining that the initial component mass fractions that were produced may not be precisely those defined by the initial condition information in the initial values file.

3.2.16. Volume Fraction

For details on setting up initialization for multiphase cases, see Initial Conditions for a Multiphase Simulation (p. 173).

3.2.17. Radiation Intensity

A sensible value would be to assume that the fluid is in radiative equilibrium; that is, the radiation intensity is set to the initial blackbody intensity in the fluid:

$$\text{Radiation Intensity}_{\text{initial}} = \frac{\sigma T^4}{\pi} \tag{3.8}$$

where σ is the Stefan-Boltzmann constant.

3.2.18. Initialization of Solid Domains

For solid domains, you may need to supply information for the following:

- Coordinate Frame (p. 164)
- Frame Type (p. 165)

Initialization data is also required for all variables (for example, Temperature, radiation, and Additional Variables) associated with the models that are activated on the solids domain.

3.2.19. Initial Conditions for a Multiphase Simulation

For a multiphase simulation, the initial conditions for some variables must be the same for all fluids. For example, in CFX, the pressure is assumed to be common for all fluids. However, for other variables, the initial conditions can be specified for each fluid individually.

Initial conditions for parameters common to both fluids are set in the same way as for single-phase simulations. The initial conditions that are fluid specific are set on the **Fluid Specific Initialization** frame. For details, see Fluid Specific Initialization in the *CFX-Pre User's Guide*. As with single-phase simulations, the initial conditions can be set to `Automatic` or `Automatic with Value`. For details, see Modeling Advice for Multiphase Flow (p. 344).

3.2.19.1. Volume Fraction

If a CFX-Solver default value is used for the **Fluid Volume Fraction**, it is distributed evenly among the fluids for which a volume fraction has not been assigned. For example, for two phase flow, an initial **Fluid Volume Fraction** of `0.5` would be assigned to each fluid if the defaults are used. For many simulations, including those containing a dilute fluid, using the Solver default values is not recommended. Instead, set a sensible value for the initial **Fluid Volume Fraction** using `Automatic with Value`. Solver default values are used when the `Automatic` option is selected without an initial values file, or when `Default` is selected. If values are entered for the **Fluid Volume Fraction**, they must sum to 1.

3.2.19.2. Velocity

The initial velocity fields for each fluid should be slightly different to ensure a non-zero slip velocity. You should set the difference to a fraction of the expected terminal slip velocity.

3.2.20. Initialization Advice

If you are having problems getting your solution started, it will often be due to a poor initial guess. For many simple modeled flows, such as incompressible laminar, or incompressible turbulent with the `Zero Equation` model, a zero initial condition for all variables is sufficient for robust convergence.

Usually, however, better initial convergence rates can be achieved if the pressure and temperature initial values are set close to the boundary condition values, and if the velocity components are some fraction of an expected average value (that is, under estimating initial velocities is better than over estimating). For turbulent flow simulations using the *k-e* model, some initial velocity scale is strongly recommended.

For more difficult flows, such as compressible flow in a complex geometry, with the 2nd order discretization option (blend factor of 1.0), and with multiple pressure specified openings, you may find that the linear equation solver fails no matter what initial values are selected.

For difficult flows, you should first obtained a solution using more robust solver settings and models. For details, see Monitoring and Obtaining Convergence (p. 578).

Important:

When using an incompressible solution as the initial guess for a compressible case, you should make sure that the incompressible case had a sensible reference pressure set and that there are no negative values of pressure in the domain.

An incompressible case in which the reference pressure is zero is likely to contain negative pressures. This is not a problem for incompressible flow because only the pressure difference is of importance. For a compressible case, the absolute pressure is of importance and negative pressures are non-physical.

3.3. Reading the Initial Conditions from a File

Reading the initial conditions from one or more existing results files ("Initial Values Files") will often provide a flexible way to initialize a run and to provide initial conditions. See Initial Condition Modeling (p. 161) for some examples of circumstances when this may be useful. Most simulations will require the use of only one Initial Values File.

When you read initial conditions for a run from an existing results file, then there are various controls that affect how the results in the results file are used to provide the necessary initial conditions. These are described in the sections Continuing the History (p. 175), Using Multiple Files to Provide Initial Conditions (p. 179), Using the Mesh from the Initial Values File (p. 179), and Interpolation Mapping (p. 190).

If the Initial Values File contains particle data from a Lagrangian Particle Tracking Simulation, then some limitations apply. See Using an Initial Values File that Contains Particles (p. 180) for details.

Sometimes the new simulation will require the initialization of a variable that is not present in the Initial Values File. Wherever there is a variable missing from the Initial Values File(s), the CFX-Solver will fall back to the initial values specified in CFX-Pre on the Initialization panel - it will use the specified value if the `Automatic with Value` option was chosen in CFX-Pre, or a solver-calculated value if the `Automatic` option was chosen. For example, the Initial Values File might contain the results of a simulation with just a single fluid "Water", but the new simulation might have two fluids "Water" and "Air" in an inhomogeneous multiphase set-up. In this case, the CFX-Solver will initialize all the variables it can from the Initial Values File, including variables such as `Pressure` and `Water.Velocity`. The initial conditions for the remaining variables, including `Water.Volume Fraction`, `Air.Volume Fraction` and `Air.Velocity` (which are not present in the Initial Values Files), will all be read from the CFX-Pre settings.

When multiple fluids (or components of fluids) are present in the Initial Values File(s) and/or the new CFX-Solver Input file, especially if more than one Initial Values File is used, the logic used to decide which variables in the Initial Values File(s) map to which variables in the CFX-Solver Input file may not be obvious. This is described in detail in Using the CFX-Interpolator (p. 182).

The specification of an Initial Values Files is part of the **Run Definition** rather than part of the physics set-up. An Initial Values File can be specified in any of the following locations;

- The **Run Definition** tab of the **Define Run** dialog box of the CFX-Solver Manager (see Run Definition Tab in the *CFX-Solver Manager User's Guide*)

- The **Run Definition** tab of the **Execution Control** details view in CFX-Pre (see Run Definition Tab in the *CFX-Pre User's Guide*)
- The **Run Definition** tab of the **Configuration** details view in CFX-Pre, for individual configurations of a multi-configuration simulation (see Run Definition Tab in the *CFX-Pre User's Guide*).

Unless the setting **Use Mesh From** (described in Using the Mesh from the Initial Values File (p. 179) is set to `Initial Values`, then behind-the-scenes the CFX-Interpolator will be called at the start of any run using an Initial Values File in order to extract the appropriate initial values, and you will see a section in the CFX-Solver Output file headed

```
+--------------------------------------------------------------------+
|                                                                    |
|                  Interpolation of Initial Values                   |
|                                                                    |
+--------------------------------------------------------------------+
```

You should check this carefully to ensure that the diagnostic information presented represents what you might expect given your new run settings, your existing results file and any settings made to control how the initial values are used. Details on how the CFX-Interpolator works are given in Using the CFX-Interpolator (p. 182).

Visualization of the initial values used (typically using CFD-Post) is recommended for any situation where the initial conditions are important (for example, for a transient run) and you are not sure what to expect, or where the diagnostic output from the CFX-Interpolator is unexpected or reports a problem. The best way to visualize the initial conditions actually used by the CFX-Solver is to view the transient or backup file written before the first timestep or outer iteration of the new run begins. The `backup file at start` parameter controls whether a backup file is written at the start of every run.

3.3.1. Using Configuration Results to Provide Initial Values

If you are performing a multi-configuration simulation, then you can specify that Initial Values for a configuration are provided by `Configuration Results`. As far as the CFX-Solver is concerned, this behaves exactly the same as if the Initial Values are set to be read from a specified results file, and all the information about reading initial values from a file applies in exactly the same way. The only difference is that the actual location and exact name of the results file is not known until just before that configuration starts to run.

3.3.2. Continuing the History

In some circumstances where you choose to read initial conditions from a file, you need just an approximate initial guess for the main solved variables (particularly velocity and pressure) to get your run started. However, in many circumstances you might want to continue as smoothly as possible from a previous run, particularly if you are running a single transient case that has to be stopped and restarted, perhaps to enable a new mesh to be used, or to enable the geometry to change at a particular time (for example, when a valve opens).

The setting that controls this choice is the **Continue History From** check box. This setting can be made on any panel which allows the specification of an Initial Values file. (If the CFX-Solver is being started from the command line, then the `-continue-from-file` or the `-continue-from-configuration` can be used to turn on the **Continue History From** setting.)

If you want to use an initial guess for your new run, then you can choose to clear (turn off) the **Continue History From** check box. (If the CFX-Solver is being started from the command line then the **Continue History From** setting will be off if all the Initial Values Files are specified using the `-initial-file` and `-initial-configuration` arguments.)

If you want to have the smoothest possible run continuation from a previous run, then you should select (turn on) the **Continue History From** check box. If you have just a single Initial Values file, then the history must be continued from this file. If you have multiple Initial Values files, then the history can be continued from only one file, which you must select.

The setting of the **Continue History From** check box has various effects that are tabulated in more detail below. In general, the aim of this setting is to provide as smooth a restart as possible, given the contents of the existing results file and the setup of the new CFX-Solver Input file.

Continue History From check box	**Cleared (off)**[a]	**Selected (on)**
Variables	Excluding Mesh Displacement (described below), only principal variables and solver-internal derived variables that are under-relaxed or have a recursive definition are copied/interpolated. For example, mass flow integration point data will not be copied/interpolated. In general, this option should be switched off only if a rough initial guess is required.	All principal variables and dependent variables will be copied to make the run continuation as smooth as possible if the source domain and target domain have the same mesh. If the source domain and target domain have different meshes, only principal and solver-internal derived variables that are under-relaxed or have a recursive definition are interpolated. Particle database, particle tracks and particle sources will be copied/interpolated, if appropriate. Depending on the similarity between the mesh/topology/physics in the CFX-Solver Input file and Initial Values File(s), data on GGI interfaces, faceset data and element-set data may also be copied from the Initial Values File for use in the new run. More details can be found in Interpolating from a Single File (p. 183).
Initial Values File containing Mesh Displacements	If the **Mesh Deformation Option** (on the **Interpolator** tab; see Interpolator Tab in the *CFX-Pre User's Guide*) is set to `Automatic` or `Use Latest Mesh`, then the CFX-Interpolator will compare the mesh in the Solver Input File with the final mesh in the Initial Values File. If they are the same then the all relevant variables (not including mesh	If the **Mesh Deformation Option** (on the **Interpolator** tab; see Interpolator Tab in the *CFX-Pre User's Guide*) is set to `Automatic` or `Use Initial Mesh`, then the CFX Interpolator will compare the mesh in the Solver Input File with the initial mesh in the Initial Values File. If the meshes are the same, then the relevant variables (including mesh displacements and mesh velocities) are copied from the Initial

Continue History From check box	**Cleared (off)**[a]	**Selected (on)**
	displacement, mesh velocity or final mesh) are copied from the Initial Values File to the Solver Input File; otherwise the variables are interpolated. Setting the **Mesh Deformation Option** to `Use Initial Mesh` makes the CFX Interpolator check whether the mesh in the Solver Input File matches the initial mesh in the Initial Values File instead. If these meshes match, then the variables are copied (including mesh displacements and mesh velocities) from the Initial Values File rather than interpolated. This could result in unexpected behavior because it results in variables being copied from nodes in the Initial Values File to the equivalent nodes in the Solver Input File without taking proper account of their temporal derivatives in a second order backward Euler transient scheme. However, this can be the desired behavior for a minority of analyses such as mesh motion with `Specified Displacement` or `Specified Location`.	Values File to the Solver Input File; otherwise variables (not including mesh displacement or mesh velocity) are interpolated. Setting the **Mesh Deformation Option** to `Use Latest Mesh` makes the CFX-Interpolator check whether the mesh in the Solver Input File matches the final mesh in the Initial Values File instead. If these match then the variables (not including mesh displacement or mesh velocity) are copied from the Initial Values File rather than interpolated. However, this can be the desired behavior if a smooth restart is desired and the Solver Input File mesh matches the Initial Values File final mesh.
Iteration/Timestep Counter	The new run will start from an `Outer Iteration` (steady-state) or `Timestep` (transient) of zero. The CEL variables `Accumulated Timestep`, `Accumulated Coupling Step` and `Accumulated Iteration Number` will be equivalent to `Current Timestep`, `Current Coupling Step` and `Current Iteration Number`, respectively.	The new run will start from the `Outer Iteration` (steady-state) or `Timestep` (transient) at which the previous run finished. The CEL variables `Accumulated Timestep`, `Accumulated Coupling Step` and `Accumulated Iteration Number` will *not* be equivalent to `Current Timestep`, `Current Coupling Step` and `Current Iteration Number`, respectively, but instead will continue to accumulate from the previous run (in the Initial Values File).

Continue History From check box	**Cleared (off)**[a]	**Selected (on)**
Monitor Data	The new run does not contain monitor data from the previous run.	The new run contains the monitor data from the previous run. This data will be plotted by default whenever the new run is monitored in the CFX-Solver Manager (although it can be excluded by using the **Monitor Properties** dialog box and setting the **Timestep Range Option** to `This Run Only`).
Particle Tracks	The new run does not contain any Lagrangian particles (or their tracks) from the previous run, although the particle sources to the fluid(s) from the previous run will be available.	Particle tracking will continue from the particles in the previous runs, particle tracks will be available from the previous run, and particle sources to the fluid(s) will be available. Some limitations apply: for details, see Using an Initial Values File that Contains Particles (p. 180).
Run History	The new run does not contain any history relating to the existence of a previous results file nor its associated transient files.	If the results of the new run are loaded into CFD-Post with the setting **Load Complete History as a single case/separate cases** selected, then CFD-Post will be aware of the existence of the Initial Values File and all of its associated transient and files.
Transient Statistics	Transient statistics from the previous run are not used or available.	Transient statistics continue to accumulate from the Initial Values File into the new run.

[a] This does not apply if **Use Mesh From** is set to `Initial Values`. See Using the Mesh from the Initial Values File (p. 179) for details.

Note:

Regardless of whether the **Continue History From** check box is selected or cleared, the following rules apply:

- If the initial values file contains results from a time transformation run, and the target solver input file is not a time transformation run, then the interpolator will only copy or interpolate values reconstructed from the Fourier coefficients. Variables without Fourier data will be skipped. This rule applies regardless of whether there is an interpolation mapping object defined.
- If the initial values file contains Fourier data from a transient blade row simulation, and the source passage is replicated (that is, **Total Num. Instances** is greater than 1), the interpolator will only interpolate values that are reconstructed from the Fourier coefficients. Variables without Fourier data will be skipped.

3.3.3. Using Multiple Files to Provide Initial Conditions

You might want to use more than one Initial Values Files to provide initial conditions for your new run. This is most likely to be applicable if you have simulated multiple parts of a geometry (for example, multiple stages of a rotating machine) separately, and then want to use these existing results to initialize a simulation of the full geometry (for example, the full rotating machine). Another example would be where you run a transient simulation where at some point during the calculation, a valve opens and connects two previously-separate enclosures: you could perform a lengthy transient run simulating the complex physics in one part of the geometry up to the time where the valve opens, perform a quick steady-state simulation of the less-interesting conditions in the other part of the geometry, and then use both existing results files to provide the initial conditions for the run of the full geometry after the valve opens.

Multiple Initial Values Files can be specified by creating more than one Initial Values object on any of the panels where you can specify an Initial Values File.

The **Continue History From** check box setting (see above) can only be applied to one of the Initial Values Files selected. The monitor data, particle positions and tracks, and iteration/timestep counter will be read from that file. Particle positions and tracks and monitor data from the other file(s) will be ignored.

It is recommended that where practical, separate domains are created in the new CFX-Solver Input file for the region of space occupied by each of the Initial Values Files (that is, each new domain only overlaps with the geometry from one of the Initial Values Files), particularly when the Initial Values Files do not have identical physics and material names. This is because the CFX-Interpolator interpolates results domain by domain, and if each domain only picks up initial values from one Initial Values File, then the resulting mapping of variables is simpler. Where it is not possible to have one or more separate new domains for each existing Initial Values File, then it is recommended that you read the section Interpolating from Multiple Files (p. 188) to be aware of the limitations.

Where your Initial Values Files have geometry that overlap or do not fully cover the geometry of the new CFX-Solver Input file, it is recommend that you read the section Interpolating from Multiple Files (p. 188) to be aware of how the CFX-Interpolator calculates the initial values for these cases.

3.3.4. Using the Mesh from the Initial Values File

If you have specified only one Initial Values Files, then you can choose whether the new simulation should use the mesh (and topology) from the CFX-Solver Input file (default) or the Initial Values Files, by using the setting **Use Mesh From**. You should choose the `Initial Values` option only if the following conditions are satisfied:

- The number of boundary condition definitions in your current model and the results file is the same.
- The names and locations (that is, surface/face identifiers) of boundary conditions are the same.
- The mesh is the same in both your current model and the results file.

This setting is included mainly for backwards-compatibility with Release 11.0 and previous releases (where it was the default unless **Interpolate Initial Values onto Def File Mesh** was selected); however, it may occasionally be useful in other circumstances.

If the meshes in the CFX-Solver Input file and the Initial File are identical, then this setting should have no effect on the simulation if **Continue History From** is set to the single Initial Values File.

Note:

If **Use Mesh From** is set to `Initial Values`, then the CFX-Interpolator is not used at all. If **Use Mesh From** is set to `Solver Input File`, then the CFX-Interpolator will be used, but it will recognize if the mesh and topology in the Initial Values File is identical to the CFX-Solver Input file and simply copy all of the variables and other settings in this circumstance - no actual interpolation will be performed. The end result should be same regardless of which option is chosen if the mesh in the CFX-Solver Input file and Initial Values File is identical, but the CFX-Solver Output file will show a section for interpolation in the case where the `Solver Input File` setting was selected for **Use Mesh From**.

When **Use Mesh From** is set to `Initial Values`, then the **Continue History From** setting behaves differently to the description in the table in Continuing the History (p. 175). If the **Continue History From** check box is selected, then the description in the table is correct. If **Continue History From** has been cleared (turned off), then no monitor data and run history is available for the new run, and the timestep/iteration counter is reset. However, the particle data and the variable data is unaffected by the **Continue History From** check box being cleared (turned off).

3.3.5. Using an Initial Values File that Contains Particles

If an Initial Values File contains the results of a Lagrangian particle tracking simulation, then three types of data can be made available to the new run:

- particle database containing particle positions at the end of the run, which will be used to continue particle tracking calculations in the new run if appropriate
- particle tracks (the tracks that the particles have already followed), which can be used for postprocessing
- particle source terms in the fluid(s) equations.

If the **Continue History From** is cleared (turned off), or the new run is steady-state, then only the particle source terms in the fluid(s) equations will be made available for the new run. The particle database and existing particle tracks are not used in the new run. The only particles present in the new run will be those injected during the new run.

If **Continue History From** is set to an Initial Values File which contains particle data, and the new run is transient, then all three types of data may be made available in the new run. However, there are a number of limitations that determine the type of data made available for any given run. Three scenarios are possible, with the final scenario being the default:

- Only the particle sources are made available for the new run. In this case, the only particles present in the new run will be those injected during the new run. When postprocessing, the only particle tracks available will be those calculated in the new run.
- The particle sources and the particle database are made available for the new run, but the particle tracks are not. In this case, if the new run defines particles of the same type as those contained in the Initial Values File, then the active particles from the Initial Values File will continue to be tracked from their positions and attributes at the end of the previous run (and

new particles will be injected according to the settings in the new run). However, the tracks from the previous run will not be available in CFD-Post, so the history of how the particles from the Initial Values File arrived at those positions and attributes will be lost.

- All of the particle sources, particle database, and particle tracks are made available. In this case, particles from the Initial Values File continue to be tracked as described in the point above, and additionally the particle tracks postprocessed using the results from the new run will include the tracks from the Initial Values File.

Limitations associated with continuing particle calculations from an Initial Values File, and in making the particle data from the Initial Values file available for a new run, are detailed below:

- Only the particle database and particle tracks from the single Initial Values File specified by the "Continue History From" setting will be made available to the new run. Particle databases and tracks in other Initial Values Files will be ignored. It is not currently possible to combine the particle databases from two or more Initial Values Files for use in the new run.

- If the new run contains a different mesh to the Initial Values File, then the CFX-Interpolator is used to "relocalize" the particles from the Initial Values File onto the mesh in the CFX-Solver Input file. This means that for each particle in the Initial Values file, the CFX-Interpolator has to locate the element in the CFX-Solver Input file that now contains the particle, given its coordinates in the particle database from the Initial Values File. If the mesh has changed, then some particles in the particle database of the Initial Values File may appear to be outside the new run's mesh even if the geometry has not changed, if the underlying geometry has curved faces and the particles were very close to the edge of the mesh in the Initial Values File. In this case, the CFX-Interpolator will attempt to remap these initially-unmapped particles to the closest element, but will only succeed if the particles are close enough to that element, to within a tolerance. You can check whether particles were mapped successfully or not by referring to the section in the CFX-Solver Output file entitled "Particle Relocalization Information". If the geometry is highly curved and the mesh is coarse, then this tolerance might need to be adjusted to ensure that all the particles in the Initial Values File are successfully relocalized. See Adjusting the Bounding Box Tolerance (p. 193) for details on how to do this. If the geometry is significantly different between the Initial Values File and the CFX-Solver Input file, then any particles that cannot be relocalized onto the CFX-Solver Input file mesh will be lost from the particle database.

- By default, particle tracks are stored according to the domain that contains them. The particle tracks in this format from an Initial Values File can only be made available in the new run if the CFX-Interpolator is able to match the domain that contains the particles tracks to an equivalent domain in the CFX-Solver Input file. The tracks will only be made available if the geometry between the two equivalent domains is nominally the same; that is, 90% or greater of the nodes of the domain in the CFX-Solver Input file can be mapped to the equivalent domain in the Initial Values File and the bounding box volumes overlap by at least 90% between the two domains.

  – If the geometry is the same but due to numerical error or mesh faceting (due to geometrically curved surfaces) the 90% criteria is not met, then you can control the tolerance that the CFX-Interpolator uses to decide if the nodes are mapped or not. This is described in Adjusting the Bounding Box Tolerance (p. 193).

  – If each domain in the Initial Values File does not have the same geometry as a domain in the CFX-Solver Input file (because the geometry has changed or because the domains have been defined differently (for example, by combining two domains into one)), then you can set the expert parameter `pt zone specific tracks = f`. This stores the particle

tracks independent of the domain. This is the recommended setting for multi-configuration runs.

- Particle tracks in the new run can only be associated with a Boundary Condition if the Boundary Condition names remain the same in the CFX-Solver Input file as they were in the Initial Values File.
- Time-integrated particle boundary vertex fields (such as `Time Integrated Mass Flow Density`) are not accumulated from the Initial Values File into the new run unless the mesh in the Initial Values File is the same as the mesh in the CFX-Solver Input file.
- Wall impact variables on boundary patches are copied into the CFX-Solver input file if they are found in the initial values file. If the target mesh is different from the source mesh, the CFX-Interpolator maps each source boundary patch to its target boundary patch, and copies the wall impact variables to the CFX-Solver input file accordingly, in order to ensure a smooth restart.
- Time-integrated mass, momentum, and energy flows on particle injection regions are copied into the CFX-Solver input file if they are found in the initial values file. The CFX-Interpolator assumes that the particle injection regions are the same in the initial values file and CFX-Solver input file. If the particle injection regions are different, the CFX-Interpolator might not copy the time-integrated mass, momentum, or energy flows to the CFX-Solver input file correctly.

3.4. Using the CFX-Interpolator

The CFX-Interpolator allows the results from one or more results files (the "source files") to be used as initial conditions for a run defined in a single Solver Input File (the "target file"). Although it will often be used where interpolation of results from one mesh to another is required, it can also deal efficiently with cases where no interpolation is required (the source file(s) have the same mesh as the target file), and can also be used for cases where physics and other setup information is different between the source and target files.

The CFX-Interpolator can also be used to calculate the differences between two results files. This is discussed in Using the CFX-Interpolator to Calculate Difference Variables (p. 196). The rest of this section considers only the use of the CFX-Interpolator to provide initial conditions.

To be able to run, the CFX-Interpolator must be given the names of the source and target files, and it outputs data in two places:

- it writes variables and other appropriate data to the target file
- it writes diagnostic information to a text output.

The most common way to use the CFX-Interpolator is to choose to specify one or more Initial Values Files as set on the **Run Definition** tab, either in CFX-Pre or in the CFX-Solver Manager. Unless the `Use Mesh From` option is set to `Initial Values`, the CFX-Interpolator will always be used to extract the results from the Initial Values Files for use in the new run (you do not need to explicitly select to use it). In this case, the target file itself is never seen directly by you (it is written in the CFX-Solver working directory and deleted at the end of the run) and the text output is written to the CFX-Solver Output file.

If you want to manually use the interpolator to write variables into a specific CFX-Solver Input file (not as part of a Run Definition) then you can choose **Tools** > **Interpolate Results** from the CFX-Solver

Manager. In this case the specified Mesh File is used as the target file, and will be modified by the interpolation process. The text output is written into the CFX-Solver Manager's Interpolation window. You can then run the resulting target file in the CFX-Solver, using the variables written into the target file by the CFX-Interpolator as the initial conditions for the run.

You can also run the CFX-Interpolator manually using the `cfx5interp` command on the command line. See Using the Command Line to Interpolate Results in the *CFX-Solver Manager User's Guide* for details. In this case the target file is the file specified by the `-mesh` argument, and the text output is written to the command line window.

The following topics are discussed:

3.4.1. Interpolating from a Single File
3.4.2. Interpolating from Multiple Files
3.4.3. Interpolation Mapping
3.4.4. Adjusting the Bounding Box Tolerance
3.4.5. Interpolating Onto a Solver Input File with Results Fields
3.4.6. Miscellaneous Limitations of the CFX-Interpolator
3.4.7. Using the CFX-Interpolator to Calculate Difference Variables

3.4.1. Interpolating from a Single File

This section describes how the CFX-Interpolator deals with the most common case, where the results from a single source file are interpolated onto the target file.

The CFX-Interpolator can be run in two different modes: "Initial Guess" mode and "Run Continuation" mode. The aim of Run Continuation mode is to provide the cleanest restart possible, when starting the run defined in the target file using the results in the source file. The aim of Initial Guess mode is to provide a basic initial condition for the principal (solved) variables. The differences between these are discussed in detail in Continuing the History (p. 175). If the **Continue History From** check box is selected, then the CFX-Interpolator is run in Run Continuation mode when it is called as part of the run setup. If **Continue History From** check box is cleared, then the CFX-Interpolator is run in Initial Guess mode.

The basic outline of the interpolation process (conceptually) is as follows, when the CFX-Interpolator is run in Run Continuation mode.

- For each domain (which is not of type Immersed Solid) in the target file:

  1. Same Mesh Check.

     Check to see if any appropriate domain in the source file has the same mesh as the target domain under consideration. (See Mapping Data from the Source File to the Target File (p. 185) for a discussion of which types of domain can be used to interpolate onto any particular domain, and what is meant by "same mesh".) If it has, then copy all the data from that domain to the target domain, including vertex values, faceset values and element-set values. No more processing needs to be performed for this target domain.

  2. Determine mapped nodes.

1. Find the relevant domain in the source file that has the largest bounding box overlap with the target domain under consideration. For each node in the target domain, determine whether it can be mapped to an element in the source file domain under consideration.

2. Repeat the above step for the remaining relevant domains in the source file, in order of descending bounding box overlap.

3. If a node can be mapped to more than one source domain, then choose to map that node to the element in the source domain with most mapped destination nodes.

4. For all the target nodes that can be mapped to a source element, calculate the interpolated nodal values for the vertex variables (for example, `Pressure` or `Velocity`). These nodes need no further consideration.

3. Project unmapped target nodes that are only just outside the source geometry.

   If `Simulation Control` command language is used to set the bounding box tolerance (as described in Adjusting the Bounding Box Tolerance (p. 193)), and, if any unmapped target nodes are within the bounding box tolerance of any source elements, then map these nodes to the nearest face/element/node of the appropriate source element. Calculate interpolated nodal values for the vertex variables appropriately; these nodes need no further consideration. This situation applies to target nodes that are only just outside the source elements, perhaps because of numerical precision issues, or because of mesh faceting of curved geometric faces.

4. Extrapolate any remaining unmapped nodes.

   If any target nodes are still unmapped, then use extrapolation from the nearest mapped target nodes to provide nodal values for the vertex variables. These nodes are reported as being "unmapped" in the CFX-Interpolator diagnostic output. When unmapped nodes remain in the target domain, the interpolation will write a new variable `Interpolation Source Domain` onto the target file and you can plot this variable in CFD-Post to visualize the location of the unmapped nodes. The value of `Interpolation Source Domain` is zero on unmapped nodes and, otherwise, a real number corresponding to a solver-internal domain number.

5. All nodes in the target domain should now have appropriate vertex values.

- For each immersed solid domain in the target file:

  If there is an immersed solid domain in the source file with the same name as the target immersed solid domain, then copy the domain motion data from the source file to the target immersed solid domain.

- For each rigid body object in the target file:

  If there is a rigid body object in the source file with the same name as the target rigid body then copy the rigid body motion, force, torque, and convergence data from the source file to the target rigid body object.

- For each particle type in the target file:

If the same particle type exists in the source file, then copy the appropriate particle data, as described in Using an Initial Values File that Contains Particles (p. 180). For details on what is regarded as the "same" particle type, refer to Mapping Data from the Source File to the Target File (p. 185).

- For each Generalized Grid Interface (GGI) in the target file:

  If there exists an identical GGI interface in the source file, then copy the GGI data from that GGI onto the target file GGI. The criteria used to determine whether or not the GGIs are identical are discussed below in Mapping Data from the Source File to the Target File (p. 185).

If the CFX-Interpolator is run in Initial Guess mode, then some of these steps are skipped: only the principal (solved) variables, derived variables, and material properties are copied/interpolated from the source file to the target file; the other data (such as variables relating to immersed solids, GGI interfaces and particle tracking, non-derived dependent variables, and monitor data) is not considered at all (except that the particle sources to the fluids' equations are copied/interpolated).

When using the interpolator for cases involving mesh motion or remeshing, note that:

- When the CFX-Interpolator is run in Run Continuation mode, the same mesh check is performed by comparing the initial meshes from the source with the mesh in the target domains. This is done so that mesh morphing restarts are smooth. If the meshes are the same, the CFX-Interpolator will copy the mesh displacements and the latest mesh coordinates from the source domains.
- When the CFX-Interpolator is run in Initial Guess mode, the same mesh check is performed by comparing the latest meshes from the source with mesh in the target domains. Even if the meshes are the same, the CFX-Interpolator will ignore the mesh displacements and the mesh coordinates from the source domains. This is done for backwards-compatibility with releases prior to Release 12.0.
- If the meshes are different, the node mapping is performed using the mesh in the source file at the latest time.

3.4.1.1. Mapping Data from the Source File to the Target File

The CFX-Interpolator has to read the objects defined in the target file (for example, particle types, fluids, domains, and GGI interfaces) to determine what the corresponding objects in the source file are, in order to copy or interpolate the correct data. The criteria listed in the table below are used.

Table 3.1: Data-mapping Criteria

Object	Criteria to determine which is the corresponding object in the source file (if any)
Fluid and Porous domains	Data from Fluid and Porous domains in the source file can be interpolated or copied only to domains defined as being either `Fluid` or `Porous` in the target file. Domains which have a **Domain Motion** set to `Rotating` will only be used to interpolate onto other rotating domains, and domains that have the **Domain Motion** set to `Stationary` will only be used to interpolate onto other stationary domains (note that data in one rotating domain can be interpolated onto a domain that has a different rotation rate: you must determine whether or not this will give a sensible result given your problem setup and

Object	Criteria to determine which is the corresponding object in the source file (if any)
	how you are using the CFX-Interpolator). There is no requirement for Fluid and Porous domains to have the same name, mesh or geometry between the source and target files; only the geometrical overlap between the domains is important.
Solid domains	Data from a Solid domain in the source file can only be interpolated or copied to a domain of type Solid in the target file. There is no requirement for Solid domains to have the same name, mesh or geometry between the source and target files, only the geometrical overlap between the domains is important.
Immersed Solid domains	Immersed Solid domains are considered to be the same if they have the same name. There is no requirement to have the same mesh.
Same Mesh check	A domain in the source file is considered to have the same mesh as a domain in the target file if the following criteria are all met: • The bounding box volumes of the domains must be the same (to within a very small tolerance). • The topology of the domain meshes must be the same. • The mesh coordinates for each node must be the same (to within a very small tolerance) in order to copy the nodal solution data. This implies that meshes will *not* be considered to be the same if: – Node re-ordering has been used to change the order of the nodes (for instance, node 1 in one mesh does not have the same coordinates as node 1 in the other mesh) – The same boundary conditions (with the same boundary condition names) have not been defined on each mesh. If the CFX-Interpolator is run in Run Continuation mode, then: • The element-set data will be copied if the corresponding source element set is mapped to a target element set. • The faceset data will be copied if the corresponding source face set is mapped to a target face set. • The boundary condition data will be copied if the corresponding source boundary is mapped to a target boundary. Note that in some circumstances, the faceset order in the source file may not match that in the target file, even if the mesh is expected to be identical. This change in order may occur when:

Object	Criteria to determine which is the corresponding object in the source file (if any)
	• CFX-Pre has imported a Solver Input File or Results File when creating the target file from the source file (rather than reading data from a `.cfx` file) • The mesh data in the source or target files was created with CFX-Pre from a release prior to Release 12.1. In this circumstance, the meshes are regarded as being the same, but the CFX-Interpolator will not copy any faceset data. If this data is important to obtain a smooth restart, then you may be able to select **Use Mesh From** > **Initial Values** to retain it; see Using the Mesh from the Initial Values File (p. 179) for information on when this option can be used.
Fluid	Where the source file and the target file each have a single fluid, then all variables that are stored on a per-fluid basis (such as `Velocity`) will be interpolated so that the fluid in the target file uses the results for the fluid in the source as initial conditions, regardless of the Fluid Definition name or material. If either of the source file or target file has two or more fluids, then the CFX-Interpolator has to ensure that fluid-specific variables in the source file (such as `Water.Velocity`) are applied to the appropriate fluid in the target file (so `Water.Velocity` in the source file is used to provide the initial condition for `Water` in the target file, and not `Air`, for example). The CFX-Interpolator will regard two fluids as being the same if they have the same Fluid Definition name. If the fluids do not have the same Fluid Definition names, then they will not be regarded as being the same. For example, if the source file contains `Air` and `Water`, and the target file contains `Fluid Definition 1` and `Fluid Definition 2`, then no fluid-specific variables will be interpolated from the source file to the target file; only the variables not specific to a fluid (such as `Pressure`) will be used to provide initial conditions. There is one exception to this. As a special case, when there is only one fluid defined in the source file, but there is more than one fluid defined in the target file, the CFX-Interpolator will check to see if the fluid defined in the source file has the same name as any fluid defined in the target file. If there is no match, then the CFX-Interpolator will apply the initial conditions from the single fluid in the source file to the first (alphabetically) fluid in the target file. Note that material names are not considered; only fluid names are considered.
Solid	The logic for determining which Solid variables (for example, `Copper.Temperature`) in the source file should be mapped to which Solid variables in the target file is exactly the same as for Fluid variables, as described above (using the Solid Definition name instead of the Fluid Definition name).

Object	Criteria to determine which is the corresponding object in the source file (if any)
Particle Data	The logic for determining which particle data in the source file should be mapped to which particle data in the target file is exactly the same as for Fluid variables, as described above.
Fluid Component	If a fluid is defined as a Variable Composition Mixture (for details, see Multicomponent Flow (p. 64)) in the source file, and the CFX-Interpolator determines that this fluid should provide the initial conditions for a Variable Composition Mixture fluid in the target file, then the CFX-Interpolator needs to determine which component data from the source file should provide the initial conditions for each component in the target file. The components are considered the same if they have the same name. If a component in the fluid from the target file does not have the same name as a component in the corresponding fluid in the source file, then it will not pick up initial conditions from any results in the source file.
Generalized Grid Interfaces (GGIs)	The CFX-Interpolator will only copy GGI data from the source file to the target file if it finds an Interface in the source file that is identical to the interface in the target file. For the GGI interface to be considered identical, the following criteria must all be met: • The source and destination domain interfaces are both GGIs. • Both GGIs have the same physics type (for example, Fluid Fluid, or Fluid Porous). • Both GGIs have the same frame change model (for example, Frozen Rotor). Both the rotational offsets must be the same if the Frozen Rotor model is set for the GGI. • Both the associated domains have the same mesh (and boundary topology) in the target and source files.

3.4.2. Interpolating from Multiple Files

The CFX-Interpolator can be used with more than one source file. When this might be useful is described in Using Multiple Files to Provide Initial Conditions (p. 179). Interpolation proceeds in a very similar manner to that described in Interpolating from a Single File (p. 183), but searching all the source files for matching domain or other data.

For each Fluid, Porous or Solid domain in the target file, the following checks are made to determine which source file and which domain the initial conditions are taken from:

- Check to see if any of the source files has a domain that has the same mesh as the target domain under consideration. If it has, then copy all the data from that domain to the target domain, including vertex values, face set values and element set values. No more processing needs to be performed for this target domain. If more than one source file contains a domain with the same mesh as the target domain, then the CFX-Interpolator will choose one of matching domains to provide the initial conditions, depending upon the order in which the

source files were specified, and it will note which domain was used in the diagnostic information (in the CFX-Solver Output file if the interpolation is being done as part of a solver run).

- If there is no domain that passes the same mesh check, the CFX-Interpolator will do a "100% overlap check". If one of the source files has a domain that leads to no unmapped nodes, then the CFX-Interpolator will use only that domain to interpolate from. If more than one source file contains a domain that passes the 100% overlap check, then the CFX-Interpolator will choose one of matching domains to provide the initial conditions, depending upon the order in which the source files were specified, and it will note which domain was used in the diagnostic information.

- If none of the source domains pass the "100% overlap check", the CFX-Interpolator will identify which domains overlap, extract a list of variables that are common to those domains, and interpolate only those variables. The diagnostic information gives the node mapping information, and the list of common variables is also given. The same rules described in Mapping Data from the Source File to the Target File (p. 185) apply in determining which data in the sources files maps to which data in the target file.

It is recommended that where practical, the domains in the target file are set up so that each will overlap only with domains from one source file. This will enable more flexibility in the variables available in the target file. For example, consider the situation where the target file contains Water and Air. One source file also contains Air and Water but the other source file contains just Water:

- If the target file contains just one large domain (covering all the domains from both source files), then the CFX-Interpolator will interpolate only variables like `Water.Velocity` and `Pressure` onto the target file, missing all the variables relating to Air (for example, `Air.Velocity` and `Air.Volume Fraction`) as well as `Water.Volume Fraction` even in regions where these would exist. This is because the CFX-Interpolator will only interpolate variables common to both source files onto this single domain. The CFX-Solver will then just provide initial conditions for the missing variables from the Initialization set in CFX-Pre. So there will be no use made of any volume fraction results or any variables relating to Air from the source files at all.

- However, if the target file contained two domains, one covering the region of space occupied by one source file and the other covering the region of space occupied by the other source file, then the CFX Interpolation will produce better results. In the domain that corresponds to the region of the source file that contains Water and Air, it will interpolate all the variables like `Air.Velocity` and `Water.Volume Fraction` onto the target file, rather than just the variables relating to Water and the variables that are not fluid-specific. All the information provided by the two source files will be used to set the initial conditions.

When there are multiple source files, containing (between them) multiple fluids, and a target file that may also contain multiple fluids, then it is not always possible for the CFX-Interpolator to be able to match the correct fluids in the source files with the correct fluid in the target files, unless consistent Fluid Definition names are maintained across all the source and target files. It is recommended that you always keep the Fluid Definitions (and Solid Definitions) consistent. If you do have to change the Fluid Definition names, then it is recommend that you always inspect the initial conditions used by the solver to check that the correct initial conditions are being used. This can best be done by visualizing (in CFD-Post) a transient or backup file written before the first timestep/iteration of the new run.

The following extra information on how the CFX-Interpolator determines whether fluids (or solids or particles) are the same applies when multiple source files are present:

- If each source file contains just a single fluid, then the fluid in second and subsequent source files will be treated the same as the fluid in the first source file, regardless of the Fluid Definition names. (You can override this behavior, and have the fluid from each source file mapped to the appropriate target file fluid using its Fluid Definition name, by turning on the **Enforce Strict Name Mapping for Phases** setting, which is described in Interpolator Tab in the *CFX-Pre User's Guide*.)

- If each source file contains just a single fluid, then the fluid in second and subsequent source files will be treated the same as the fluid in the first source file, regardless of the Fluid Definition names.

- If the first source file has more than one fluid, and the second or subsequent source file has only one, the fluid in the second source file will be treated as the same as the first fluid in the first source file, if no Fluid Definition names match.

- In contrast, if the first source file has only one fluid, and a second or subsequent source file has more than one fluid, then only fluids that have the same Fluid Definition name will be considered equivalent by the CFX-Interpolator.

The "first source file" will always be the one selected in **Continue History From** (or the equivalent command line argument), if this is selected. If the **Continue History From** check box is cleared, then the first source file is the one which appears first in the CFX Command Language (CCL) for the run, or is given first on the command line if the CFX-Interpolator is being run outside of a run definition. The "first fluid" is the one that appears first in the CCL for the run.

If any of the above cases apply, you should carefully check the diagnostic output and visualize the initial conditions using CFD-Post to determine that the fluids have been used as expected.

3.4.3. Interpolation Mapping

Data from a results file can be used to initialize a simulation whose mesh is positioned, oriented, or scaled differently from the corresponding source mesh. In this case, it may be necessary to align and/or replicate results file data in order to use it for initialization. For example, you may want to use the results from several different parts of a machine that were simulated in isolation in order to initialize a simulation that involves more than one of those parts. In this case, the solution data from the results file may have to be geometrically rotated, translated, and/or scaled in order to be aligned with the corresponding part of the mesh of the simulation being initialized. As another example, you may want to use solution data from a blade passage simulation in order to initialize a blade row simulation. In this case, it is necessary to replicate the single-passage solution data in order to initialize all the blade passages of the blade row simulation. You can use *interpolation mapping* to align and replicate solution data as needed. You can also use interpolation mapping to selectively apply data from certain domains within a results file.

The basic procedure for setting up and using interpolation mapping is:

1. Define any transformations required for aligning results data with the mesh of the simulation being initialized.

   These transformations may include translations, rotations, and geometric scaling.

2. For the results file used for initialization, create one or more Interpolation Mapping objects as required in order to select, position, and/or replicate solution data.

   Each Interpolation Mapping object can make use of alignment transformations that you defined in the previous step. Each Interpolation Mapping object can specify that the data is replicated with uniform rotations and/or translations between each copy.

The following subsections describe each of these steps in more detail.

Note:

If sequential transformations are required to move data into position, use a list of transformations within a single Interpolation Mapping object. Do *not* specify multiple Interpolation Mapping objects, each with one of the required transformations.

Note:

Any solution data that is, after any applicable interpolation mapping, positioned outside the mesh of the case being initialized is ignored.

Note:

When using interpolation mapping, all target domains must be interpolated using interpolation mapping objects. Each interpolation mapping object contains only one target domain.

The following topics are discussed.

3.4.3.1. Defining Transformations for use in Interpolation Mapping Alignment Transformations

3.4.3.2. Creating Interpolation Mapping Objects

3.4.3.1. Defining Transformations for use in Interpolation Mapping Alignment Transformations

You can create transformations that can be used for positioning data from a results file as part of initialization.

To add a new transformation:

1. In CFX-Pre, in the tree view, right-click the **Transformations** object and select **Insert** > **Transformation**.

2. In the **Insert Transformation** dialog box, specify a name for the transformation and click **OK**.

   The **Transformation** details view appears.

3. Specify settings for the transformation and click **OK**.

The details view settings depend on the **Transformation** option:

- **Rotation**

These settings work in the same way as the corresponding mesh transformation settings, described in Transformation: Rotation in the *CFX-Pre User's Guide*, except that the `Full Circle` option is not available and that the global coordinate frame is assumed.

- **Translation**

  These settings work in the same way as the corresponding mesh transformation settings, described in Transformation: Translation in the *CFX-Pre User's Guide*, except that the global coordinate frame is assumed.

- **Scale**

  These settings work in the same way as the corresponding mesh transformation settings, described in Transformation: Scale in the *CFX-Pre User's Guide*, except that the global coordinate frame is assumed.

Once the required **Transformation** objects have been defined, you are ready to use them in creating Interpolation Mapping objects.

3.4.3.2. Creating Interpolation Mapping Objects

Interpolation Mapping objects can be used for selecting, positioning, and/or replicating data from a results file as part of initialization. These objects are defined on the **Initial Values** tabs in these locations:

- In CFX-Solver Manager in the **Define Run** dialog box
- In CFX-Pre in the **Configuration** details view
- In CFX-Pre in the **Execution Control** details view

To add a new Interpolation Mapping object:

1. In the details view, on the **Initial Values** tab, in the **Interpolation Mapping** section, click *Add new item*  to the right of the list, type a name for the Interpolation Mapping object, and click **OK**.
2. Specify settings for the Interpolation Mapping object and click **OK**.

The Interpolation Mapping object settings are described here:

- **Source Location**: the name of the domains in the results file that require interpolation mapping
- **Target Location**: the name of the domain in the simulation being initialized onto which data from the source location is to be mapped
- **Alignment Transformation**: a list of one or more predefined transformations

  To select multiple transformations, hold the **Ctrl** key while selecting with the mouse button. The order in which you specify the transformations is important; transformations will be applied in the order you specify.

- **Replication Control**: a collection of settings for replicating the results data using repeated rotations or translations

  The `Rotation` replication control option works in the same way as the corresponding mesh transformation settings, described in Transformation: Rotation in the *CFX-Pre User's Guide*, except that:

  – The `Full Circle` option is not available.

  – There are two additional settings, **Total Num. Instances** and **Theta Offset**, that control the number of replications in, and the rotational offset for, the set of replicated solutions, respectively.

  The `Translation` replication control option works in the same way as the corresponding mesh transformation settings, described in Transformation: Translation in the *CFX-Pre User's Guide*, except that there is one additional setting, **Total Num. Instances**, that controls the number of replications in the set of replicated solutions.

  The `Turbo Rotation` replication control option involves settings similar to the `Rotation` replication control option, except that the angle of rotation between replications is specified indirectly: by setting **Pass. in Component** to the number of passages in the component mesh (the **Source Location**), and by setting **Passages in 360** to the number of passages that reach 360° around the machine axis.

Note:

If sequential transformations are required to move data into position, use a list of transformations within a single Interpolation Mapping object. Do *not* specify multiple Interpolation Mapping objects, each with one of the required transformations.

For each Interpolation Mapping object, the **Alignment Transformation** and **Replication Control** settings are applied in the following order:

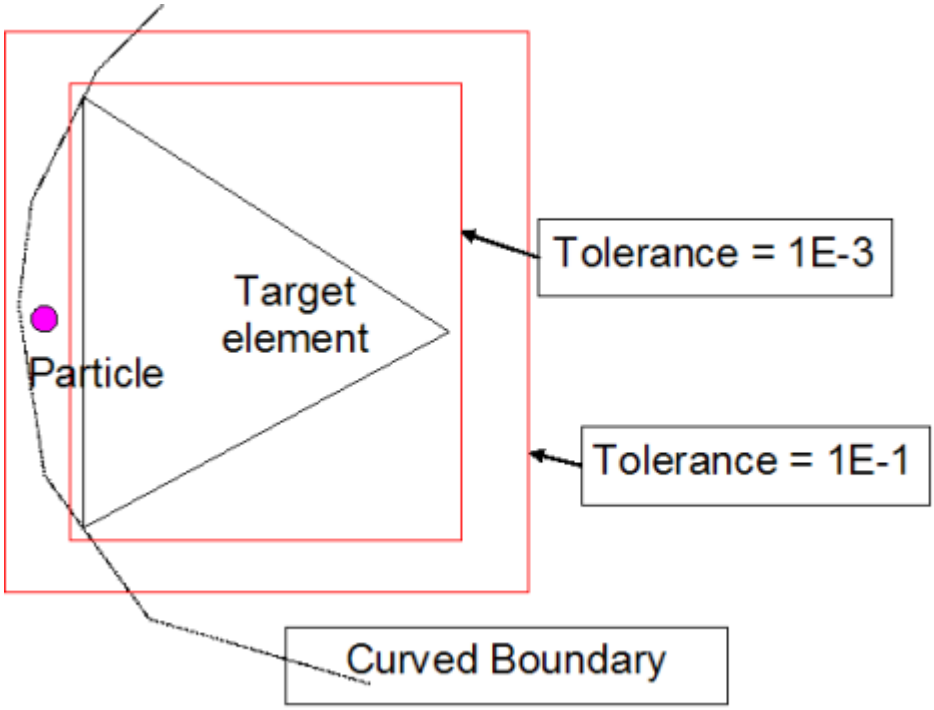
1. If specified, alignment transformations are applied in the order listed.
2. If the **Replication Control** check box is selected and the replication control setting **Transformation** is set to `Rotation` or `Turbo Rotation`, then the **Theta Offset** setting is applied. Note that the **Theta Offset** setting effects a type of alignment transformation, but is located among the **Replication Control** settings for convenience.
3. If specified, replication is applied.

If available, the **Theta Offset** setting in the `Rotation` and the `Turbo Rotation` replication control options is applied after the alignment transformation (if available), and the replication of multiple copies (that is, **Total Num. Instances**) is applied in the last transformation.

3.4.4. Adjusting the Bounding Box Tolerance

If the CFX-Interpolator is being used to relocalize particles from a source file mesh to a target file mesh, then if the mesh is different in the two files, some particles in the particle database of the source file may appear to be outside the mesh of the target file even if the underlying geometry is the same. This could occur if the underlying geometry has curved faces and the particles were very

close to the edge of the mesh in the source file, particular if the mesh was coarse in this region. In this case, the CFX-Interpolator will attempt to remap these initially-unmapped particles to the closest element, but will only succeed if the particles are close enough to that element, to within a tolerance. This tolerance is called the **Particle Relocalization Tolerance**.



The figure above illustrates the **Particle Relocalization Tolerance**. If the **Particle Relocalization Tolerance** is very small (0.001) then the particle shown in the figure will appear to be unmapped (outside the target file's mesh) due to the large curvature of the geometry relative to the local mesh length scale. However, if the **Particle Relocalization Tolerance** is increased to a larger value (for example, 0.1) then the particle will be able to be mapped to that target file mesh element. If you see that particles are (unexpectedly) unmapped in the diagnostic information provided by the CFX-Interpolator, then you should consider increasing the **Particle Relocalization Tolerance**.

Similarly, during interpolation of variables in a domain, a target node may be initially unmapped because it lies outside the source file mesh. Nodes within the **Bounding Box Tolerance** can be mapped onto the nearest element face/edge/node rather than being left as unmapped. This situation can occur when the mesh is coarse and the underlying geometry is highly-curved, as for particle relocalization. It can also occur when nodes in the target mesh are very slightly outside the source mesh simply due to numerical error (which is much less likely to happen with particles). If you see that nodes are unmapped in the diagnostic information provided by the CFX-Interpolator, but the underlying geometry in the source and target files is the same, you should consider increasing the **Bounding Box Tolerance**.

To specify the **Particle Relocalization Tolerance** and **Bounding Box Tolerance** values, visit the **Interpolator** tab of the **Configuration** or **Execution Control** (Interpolator Tab in the *CFX-Pre User's Guide*) details view in CFX-Pre, or visit the **Interpolator** tab in the **Define Run** dialog box of CFX-Solver Manager (first select **Show Advanced Controls** on the **Run Definition** tab in CFX-Solver Manager).

3.4.5. Interpolating Onto a Solver Input File with Results Fields

If the CFX-Interpolator is asked to interpolate from a source file onto a target file that already contains solution variables, then it will proceed, but it will write a warning message into the diagnostic information, and you should check the results carefully as they may not be what you expect. (This should not be confused with using the CFX-Interpolator to calculate difference variables, as described in Using the CFX-Interpolator to Calculate Difference Variables (p. 196)).

This situation will occur if you select a results file as a target file (rather than a CFX-Solver Input file written by CFX-Pre with a name of the form `*.def`), or if you select a target file that has previously been used as a target file by the CFX-Interpolator (not as part of a run definition). Exactly what initial conditions will be used if the resulting target file is used by the CFX-Solver if this situation occurs is dependent upon the details of what solution fields are already present in the target file.

If you only ever use the CFX-Interpolator as part of a run definition, then you will not encounter this situation unless you choose a CFX-Solver Results file as your **Solver Input File** *and* specify one or more Initial Values Files as part of the run definition. This should be avoided.

If you use the CFX-Interpolator outside of a run definition (such as by using the `cfx5interp` command or by using **Tools** > **Interpolate Results** from the CFX-Solver Manager) then once you use a CFX-Solver Input file written by CFX-Pre as your target file, and perform the interpolation, then this CFX-Solver Input file will now contain solution fields. If you repeat the interpolation with a different target file but with the same settings as previously (particularly with respect to whether or not the CFX-Interpolator is run in **Initial Guess** mode or **Run Continuation** mode) then the results of the second interpolation will overwrite any results of the first interpolation. Using a CFX-Solver Results file as the target file, or interpolating onto a CFX-Solver Input file that has previously been used for interpolation but with different interpolator settings should be avoided.

3.4.6. Miscellaneous Limitations of the CFX-Interpolator

- Monitor data is not interpolated if the solution units in the source file and the target file are different.
- Particle track data is not interpolated if the solution units in the source file and the target file are different.
- The CFX-Interpolator cannot interpolate values from a fluid in the one or more source files to a fluid in the target file if the fluid is defined as a Continuous or Dispersed Fluid/Solid in one file and a Particle Transport Fluid/Solid in another file. In this case it will stop with an error about inconsistent morphology.
- When using the CFX-Interpolator in initial guess mode, and the source files are generated from CFX-Solver Release 11.0 or later, then only principal variables will be copied (if the meshes are the same) or interpolated (if the meshes are different).
- When using the CFX-Interpolator in initial guess mode, and the source files are generated from CFX-Solver Release 10.0 or earlier, then both principal and derived variables will be copied (if the meshes are the same) or interpolated (if the meshes are different).
- You cannot use CEL expressions or power syntax to define any settings that are used for interpolation, including transformation settings and replication control settings.

- When using interpolation mapping, there must be exactly one object specified in the **Initial Values** list on the **Initial Values** tab, which is found in these locations:
  - In CFX-Solver Manager in the **Define Run** dialog box
  - In CFX-Pre in the **Configuration** details view
  - In CFX-Pre in the **Execution Control** details view
- In the following cases, only variables containing Fourier coefficient data are interpolated or copied:
  - The initial values file is from a time transformation simulation, but the solver input file is not for a time transformation simulation.
  - The initial values file is from a transient blade row simulation, and its solution data has been replicated onto target domains in the solver input file.
- When running, from the command line or in batch mode, a case that uses interpolation mapping, the initial values file may be specified either within the definition file or within a CCL file that contains an INITIAL VALUES object (but not within both the definition file and a CCL file). The initial values file must not be specified by the `-initial-file` or the `-continue-from-file` arguments.

3.4.7. Using the CFX-Interpolator to Calculate Difference Variables

The main use of the CFX-Interpolator is in providing initial conditions from existing results file for use in a new run. However, you can also use the CFX-Interpolator to compare two results files. When used in this mode, you provide it with a two results files, and the CFX-Interpolator writes a new set of variables into one of the results files of the form `Pressure.Difference`, which is the difference between the pressures in the two results files. These difference variables can then be viewed in CFD-Post. For a description of the general variable syntax, see Quantitative CEL Functions in Ansys CFX in the *CFX Reference Guide*.

For Release 12.0 and later, this use of the CFX-Interpolator is largely superseded by the ability of CFD-Post to calculate difference variables and to compare two results files in a more flexible way. For details, see Case Comparison in the *CFD-Post User's Guide*.

The functionality to calculate difference variables using the CFX-Interpolator is available either through **Tools** > **Interpolate Results** in the CFX-Solver Manager, or through the `cfx5interp` command on the command line. See Using the Command Line to Interpolate Results in the *CFX-Solver Manager User's Guide* for details. See Calculating Difference Variables in the *CFD-Post User's Guide* for details on how the difference of vector variables such as `Velocity` is calculated.

For moving mesh cases, the difference variables are calculated using the final mesh positions.

Chapter 4: Turbulence and Near-Wall Modeling

This chapter describes:

- Turbulence Models (p. 198)
- Modeling Flow Near the Wall (p. 226)

Turbulence models are used to predict the effects of turbulence in fluid flow without resolving all scales of the smallest turbulent fluctuations. For details, see Turbulence Models in the *CFX-Solver Theory Guide*.

A number of models have been developed that can be used to approximate turbulence based on the Reynolds Averaged Navier-Stokes (RANS) equations. Some have very specific applications, while others can be applied to a wider class of flows with a reasonable degree of confidence. The models can be classified as either eddy-viscosity or Reynolds stress models. The following turbulence models based on the RANS equations are available in Ansys CFX and are described on the following pages:

- Eddy-viscosity models:
  - Zero equation model.
  - Standard $k-\varepsilon$ model.
  - RNG $k-\varepsilon$ model.
  - Standard $k-\omega$ model.
  - Baseline (BSL) zonal $k-\omega$ based model. This model can only be selected through **Expert Control Parameters**.
  - SST zonal $k-\omega$ based model.
  - $\left(k-\varepsilon\right)_{1\mathrm{E}}$ model. This model can only be selected through **Expert Control Parameters**.
  - Curvature correction for two-equation models.
- Reynolds stress models (RSM):
  - Launder, Reece and Rodi Isotropization of Production model (LRR Reynolds Stress).
  - Launder, Reece and Rodi Quasi-Isotropic model (QI Reynolds Stress).
  - Speziale, Sarkar and Gatski (SSG Reynolds Stress).
  - SMC-ω model (Omega Reynolds Stress).
  - Baseline (BSL) Reynolds stress model.

– Explicit Algebraic Reynolds stress model (EARSM).

CFX also provides the Large Eddy Simulation (LES) and Detached Eddy Simulation (DES) turbulence models. This class of turbulence models is not based on the RANS equations.

All turbulence models in CFX use advanced wall functions to model near-wall flow. For details, see Modeling Flow Near the Wall (p. 226).

In order to predict transition, different transition models of varying complexity are available in Ansys CFX; see Ansys CFX Laminar-Turbulent Transition Models (p. 205).

Information on turbulence modeling for multiphase flow is available in Turbulence Modeling in Multiphase Flow (p. 319).

4.1. Turbulence Models

The following topics will be discussed:

- The Laminar Model (p. 198)
- The Zero Equation Model (p. 199)
- The k-epsilon Model (p. 199)
- The RNG k-epsilon Model (p. 199)
- The k-omega and SST Models (p. 200)
- Curvature Correction for Two-Equation Models (p. 202)
- Corner Correction (p. 202)
- The Reynolds Stress Model (p. 203)
- Omega-Based Reynolds Stress Models (p. 204)
- Explicit Algebraic Reynolds Stress Model (p. 205)
- Ansys CFX Laminar-Turbulent Transition Models (p. 205)
- The Large Eddy Simulation Model (LES) (p. 213)
- The Detached Eddy Simulation Model (DES) (p. 218)
- The Scale-Adaptive Simulation (SAS) (p. 224)
- Buoyancy Turbulence (p. 226)

4.1.1. The Laminar Model

Laminar flow is governed by the unsteady Navier-Stokes equations. The laminar option does not apply a turbulence model to the simulation and is only appropriate if the flow is laminar. This typically applies at low Reynolds number flows (for example, for pipe flow the laminar flow regime is ~ Re < 1000).

Energy transfer in the fluid is accomplished by molecular interaction (diffusion). In the case of high speed flows, the work of the viscous stresses can also contribute to the energy transfer. You should always check in the CFX-Solver Output file that the Reynolds number is in the laminar flow regime. If you set up a simulation using laminar flow, but the real flow is turbulent, convergence is difficult and the simulation will not reach the correct solution.

4.1.2. The Zero Equation Model

The Zero Equation model implemented in CFX is simple to implement and use, can produce approximate results very quickly, and provides a good initial guess for simulations using more advanced turbulence models. In CFX, a constant turbulent eddy viscosity is calculated for the entire flow domain. If you specify an eddy viscosity, this value is used instead of the solver calculated value. You should not use this model to obtain final results. For details, see The Zero Equation Model in Ansys CFX in the *CFX-Solver Theory Guide*.

4.1.3. The k-epsilon Model

One of the most prominent turbulence models, the $k-\varepsilon$ (*k*-epsilon) model, has been implemented in most general purpose CFD codes and is considered the industry standard model. It has proven to be stable and numerically robust and has a well established regime of predictive capability. For general purpose simulations, the $k-\varepsilon$ model offers a good compromise in terms of accuracy and robustness.

Within CFX, the $k-\varepsilon$ turbulence model uses the scalable wall-function approach to improve robustness and accuracy when the near-wall mesh is very fine. The scalable wall functions enable solutions on arbitrarily fine near-wall grids, which is a significant improvement over standard wall functions.

While standard two-equation models, such as the $k-\varepsilon$ model, provide good predictions for many flows of engineering interest, there are applications for which these models may not be suitable. Among these are:

- Flows with boundary layer separation.
- Flows with sudden changes in the mean strain rate.
- Flows in rotating fluids.
- Flows over curved surfaces.

A Reynolds stress model may be more appropriate for flows with sudden changes in strain rate or rotating flows, while the SST model may be more appropriate for separated flows. For details, see The k-epsilon Model in Ansys CFX in the *CFX-Solver Theory Guide*.

4.1.4. The RNG k-epsilon Model

The RNG $k-\varepsilon$ model is an alternative to the standard $k-\varepsilon$ model. In general it offers little improvement compared to the standard $k-\varepsilon$ model. For details, see The RNG k-epsilon Model in Ansys CFX in the *CFX-Solver Theory Guide*.

4.1.5. The k-omega and SST Models

One of the main problems in turbulence modeling is the accurate prediction of flow separation from a smooth surface. Standard two-equation turbulence models often fail to predict the onset and the amount of flow separation under adverse pressure gradient conditions. This is an important phenomenon in many technical applications, particularly for airplane aerodynamics because the stall characteristics of a plane are controlled by the flow separation from the wing. For this reason, the aerodynamic community has developed a number of advanced turbulence models for this application. In general, turbulence models based on the ε-equation predict the onset of separation too late and under-predict the amount of separation later on. This is problematic, as this behavior gives an overly optimistic performance characteristic for an airfoil. The prediction is therefore not on the conservative side from an engineering stand-point. The models developed to solve this problem have shown a significantly more accurate prediction of separation in a number of test cases and in industrial applications. Separation prediction is important in many technical applications both for internal and external flows.

Currently, the most prominent two-equation models in this area are the $k-\omega$ based models of Menter [9]. The $k-\omega$ based Shear-Stress-Transport (SST) model was designed to give highly accurate predictions of the onset and the amount of flow separation under adverse pressure gradients by the inclusion of transport effects into the formulation of the eddy-viscosity. This results in a major improvement in terms of flow separation predictions. The superior performance of this model has been demonstrated in a large number of validation studies (Bardina et al. [76]).

For theoretical details on the $k-\omega$ based models, see The k-omega Models in Ansys CFX in the *CFX-Solver Theory Guide*.

The SST model is recommended for accurate boundary layer simulations. To benefit from this model, a resolution of the boundary layer of more than 10 points is required. For details, see Modeling Flow Near the Wall (p. 226).

For free shear flows, the SST model is mathematically identical to the $k-\varepsilon$ model.

The SST model was developed to overcome deficiencies in the $k-\omega$ and BSL $k-\omega$ models. Therefore, using the SST model over these models is recommended. Details of the differences between the three $k-\omega$ based models available in CFX is available. For details, see The k-omega Models in Ansys CFX in the *CFX-Solver Theory Guide*.

One of the advantages of the $k-\omega$ formulation is the near wall treatment for low-Reynolds number computations where it is more accurate and more robust. For details, see Automatic Near-Wall Treatment for Omega-Based Models (p. 228).

The convergence behavior of the $k-\omega$ model is often similar to that of the $k-\varepsilon$ model. Because the zonal $k-\omega$ models (BSL and SST) include blending functions in the near wall region that are a function of wall distance, an additional equation is solved to compute the wall distance at the start of simulations (the first few iterations). This is done automatically by the CFX-Solver.

Similar to all RANS models, the SST model exaggerates flow separation from smooth surfaces under the influence of adverse pressure gradients. A modified SST model, called the Reattachment Modification (RM) model, may improve separation and reattachment predictions. For details, see The Reattachment Modification (RM) Model in the *CFX-Solver Theory Guide*.

The Generalized k-Omega (GEKO) model is a two-equation model, based on the k-Omega model formulation, but with the flexibility to tune the model over a wide range of flow scenarios. For details, see GEKO model (p. 201).

4.1.5.1. GEKO model

The GEKO model's main parameters are:

- Separation Coefficient
  - The Separation Coefficient is the main parameter for adjusting separation predictions for boundary layers.
  - This parameter affects all flows. Increasing this parameter reduces eddy viscosity, leading to more sensitivity to adverse pressure gradients for boundary layers and leading to lower spreading rates for free shear flows (compensated by the Mixing Coefficient, which is described below).
- Near Wall Coefficient
  - The Near Wall Coefficient affects mostly the inner part of wall boundary layers (limited to no impact on free shear flows).
  - For most applications, you can use a value of `0.5` (the default).
  - Increasing the Near Wall Coefficient leads to higher wall shear stress and wall heat transfer rates in non-equilibrium flows. In particular, the Near Wall Coefficient has a strong effect on heat transfer predictions in reattachment and stagnation regions.
  - The effect on non-generic flows (such as vortices) seems to be moderate (although this has not been systematically tested).
- Mixing Coefficient
  - The Mixing Coefficient parameter affects only free shear flows. It has no impact on boundary layers due to the blending function.
  - Increasing the Mixing Coefficient leads to larger eddy viscosity in free shear flows, leading to higher spreading rates of the velocity profile and higher levels of turbulence kinetic energy.
  - For each value of the Separation Coefficient, there is an optimal value of the Mixing Coefficient that maintains optimal free shear flows. This value can be approximated as:

    ```
    0.35 * sign(Separation Coefficient - 1) * ( | Separation Coefficient
    -1 | ) ^ 0.5
    ```
- Jet Coefficient
  - Affects mostly jet flows. The Jet Coefficient allows you to adjust the spreading rate of jet flows while maintaining the spreading rate of mixing layers. Increasing the Jet Coefficient while the Mixing Coefficient is active decreases the spreading rate for jets.
  - You can usually use a value of `0.9` (the default). For round jets, set the Separation Coefficient to a value between 1.75 and 2.00 and leave the Jet Coefficient at `0.9`.

– The Jet Coefficient has no effect in the case of a Mixing Coefficient of `0`.

  With reduction in the Separation Coefficient and the corresponding reduction in the Mixing Coefficient, the effect of the Jet Coefficient vanishes.

– The Jet Coefficient is active in a sub-model of the Mixing Coefficient (but has no impact for a Mixing Coefficient of `0`).

The GEKO model is affected by the following advanced parameters:

- Curvature Correction

  An existing model for curvature correction, which can be combined with the GEKO model.

  For details, see Curvature Correction for Two-Equation Models (p. 202).

- Corner Correction

  A non-linear stress-strain term to account for secondary flows in corners (for example, at wing-body junctions).

  For details, see Corner Correction (p. 202).

- GEKO Coefficients

  Advanced parameters that normally should not be changed.

  For details, see The GEKO Model in the *CFX-Solver Theory Guide*.

The GEKO model is also affected by a blending function, which deactivates the effects of both the Mixing Coefficient and the Jet Coefficient inside boundary layers. The default blending function is sufficient in most cases. If required, you can specify a user-defined blending function. The blending function should be formulated such that a value of 1 is used inside boundary layers and a value of 0 is used for free flows.

4.1.6. Curvature Correction for Two-Equation Models

One weakness of the eddy-viscosity models is that these models are insensitive to streamline curvature and system rotation, which play a significant role in many turbulent flows of practical interest. A modification of the turbulence production term is available to sensitize the standard eddy-viscosity models to these effects. For details, see Curvature Correction for Two-Equation Models in the *CFX-Solver Theory Guide*.

4.1.7. Corner Correction

This parameter is currently used for the GEKO turbulence model.

The only option is `Production Correction`. The value you provide is part of a non-linear stress-strain term that helps to model secondary flows in corners (for example wing-body junctions).

4.1.8. The Reynolds Stress Model

Two-equation turbulence models ($k-\varepsilon$ and $k-\omega$ based models) offer good predictions of the characteristics and physics of most flows of industrial relevance. In flows where the turbulent transport or non-equilibrium effects are important, the eddy-viscosity assumption is no longer valid and results of eddy-viscosity models might be inaccurate. Reynolds Stress (or Second Moment Closure (SMC)) models naturally include the effects of streamline curvature, sudden changes in the strain rate, secondary flows or buoyancy compared to turbulence models using the eddy-viscosity approximation. You may consider using a Reynolds stress model in the following types of flow:

- Free shear flows with strong anisotropy, like a strong swirl component. This includes flows in rotating fluids.
- Flows with sudden changes in the mean strain rate.
- Flows where the strain fields are complex, and reproduce the anisotropic nature of turbulence itself.
- Flows with strong streamline curvature.
- Secondary flow.
- Buoyant flow.

Reynolds stress models have shown superior predictive performance compared to eddy-viscosity models in these cases. This is the major justification for Reynolds stress models, which are based on transport equations for the individual components of the Reynolds stress tensor and the dissipation rate. These models are characterized by a higher degree of universality. The penalty for this flexibility is a high degree of complexity in the resulting mathematical system. The increased number of transport equations leads to reduced numerical robustness, requires increased computational effort and often prevents their usage in complex flows.

Theoretically, Reynolds stress models are more suited to complex flows, however, practice shows that they are often not superior to two-equation models. An example of this is for wall-bounded shear layers, where despite their (theoretically) higher degree of universality, Reynolds stress models often prove inferior to two-equation models. For wall-bounded flows try to use the SMC-BSL model. It is based on the ω-equation and automatic wall treatment. For details, see Omega-Based Reynolds Stress Models (p. 204).

Three varieties of the Reynolds stress model are available that use different model constants:

- Reynolds stress model (LRR-IP)
- QI Reynolds stress model (LRR-IQ)
- SSG Reynolds stress model (SSG)

In general, the SSG model is more accurate than the LRR versions for most flows. This is particularly true for swirling flows. The SSG model is therefore recommended over the other models, which are there for historic reasons and because they are standard models.

Compared to the $k-\varepsilon$ model, the Reynolds Stresses model has six additional transport equations that are solved for each timestep or outer coefficient loop in the flow solver. The source terms in the Reynolds Stress equations are also more complex than those of the $k-\varepsilon$ model. As a result of these

factors, outer loop convergence may be slower for the Reynolds stress model than for the $k-\varepsilon$ model.

In principle, the same timestep can be used for all turbulence model variants, but pragmatically the timestep should be reduced for the Reynolds stress model due to the increased complexity of its equations and due to numerical approximations made at general grid interfaces (GGI) and rotational periodic boundary conditions. If convergence is difficult, it is recommended that a $k-\varepsilon$ or $k-\omega$ based model solution be obtained first and then a Reynolds stress model solution can be attempted from the converged two-equation solution. It is frequently observed that Reynolds stress models produce unsteady results, where two-equation models give steady-state solutions. This can be correct from a physical standpoint, but requires the solution of the equations in transient mode.

The Reynolds stress models may be used with isotropic or anisotropic turbulent diffusion terms in the Reynolds Stress and Epsilon transport equations. The difference is usually second order as there is often domination by the source terms and the effects of diffusion are small. An exception might be buoyant flows, which can be diffusion dominated. However, the model that uses isotropic turbulent diffusion terms is potentially more robust than the model that uses anisotropic turbulent diffusion terms.

Selection of the appropriate model and settings are carried out on the **Fluid Models** tab of the **Domains** form in CFX-Pre.

Further theoretical model information is available in Reynolds Stress Turbulence Models in the *CFX-Solver Theory Guide*.

4.1.9. Omega-Based Reynolds Stress Models

Some of the main deficiencies of the Reynolds stress models for the simulation of boundary layers are an inheritance from the underlying ε-equation. Particularly the accurate prediction of flow separation is problematic when the ε-equation is used. Furthermore, low-Reynolds number formulations for the ε-equation are usually complex and difficult to integrate, a deficiency that is exaggerated in combination with a Reynolds stress model formulation. In order to avoid these issues, a Reynolds stress model has been implemented that uses the ω-equation instead of the ε-equation as the scale-determining equation.

There are two types of SMC-ω model in CFX:

- Omega Reynolds Stress
- Baseline (BSL) Reynolds Stress

As the freestream sensitivity of the standard $k-\omega$ model does carry over to the Reynolds stress model, the BSL Reynolds stress model was developed, which is based on the ω-equation used in the BSL model, and is preferred over the Omega Reynolds stress model. This is the formulation recommended when using the ω-Reynolds stress model.

All variants of the ω-Reynolds stress model are combined with automatic wall treatment, which allows a flexible grid resolution near the wall. For details, see:

- Modeling Flow Near the Wall (p. 226)
- Omega-Based Reynolds Stress Models in the *CFX-Solver Theory Guide*

4.1.10. Explicit Algebraic Reynolds Stress Model

Explicit algebraic Reynolds stress models (EARSM) represent an extension of the standard two-equation models. They are derived from the Reynolds stress transport equations and give a nonlinear relation between the Reynolds stresses and the mean strain-rate and vorticity tensors. Due to the higher order terms many flow phenomena are included in the model without the need to solve transport equations. The EARSM is an extension of the current $k-\varepsilon$ and BSL model to capture effects of secondary flows and of flows with streamline curvature and system rotation. For details, see Explicit Algebraic Reynolds Stress Model in the *CFX-Solver Theory Guide*.

4.1.11. Ansys CFX Laminar-Turbulent Transition Models

Engineering turbulence transition predictions are based mainly on two modeling concepts. The first is the use of low-Reynolds number turbulence models, where the wall damping functions of the underlying turbulence model trigger turbulent transition onset. This concept is attractive because it is based on transport equations and can therefore be implemented without much effort. However, experience has shown that this approach is not capable of reliably capturing the influence of the many different factors that affect transition, such as free-stream turbulence, pressure gradients and separation.

The second approach is the use of experimental correlations. The correlations usually relate the turbulence intensity, Tu, in the free-stream to the momentum-thickness Reynolds number, $Re_{\theta t}$, at transition onset. Ansys has developed a locally formulated transport equation for intermittency, which can be used to trigger transition. The full laminar-turbulent transition model is based on two transport equations, one for the intermittency and one for the transition onset criteria in terms of momentum thickness Reynolds number. It is called the Gamma Theta model and is the recommended transition model for general-purpose applications. It uses an empirical correlation (Langtry and Menter) that has been developed to cover standard bypass transition as well as flows in low free-stream turbulence environments. This built-in correlation has been extensively validated together with the SST turbulence model for a wide range of transitional flows. The transition model can also be used with the BSL or SAS-SST turbulence models. A description of the full Gamma Theta model can be found in Two Equation Gamma Theta Transition Model in the *CFX-Solver Theory Guide*.

In addition, a very powerful option has been included to enable you to enter your own user defined empirical correlation together with the Gamma Theta model, which can then be used to control the transition onset momentum thickness Reynolds number equation. The method is driven by CCL and as a result any valid CCL expression can be used to define the correlation. A simple example is shown below where the transition Reynolds number was specified as a function of the x-coordinate:

```
FLUID MODELS:
   TURBULENCE MODEL :
      Option = SST
      TRANSITIONAL TURBULENCE :
         Option = Gamma Theta Model
         TRANSITION ONSET CORRELATION:
            Option = User Defined
            Transition Onset Reynolds Number = 260.0*(1.0 + x/(1.0 [m]))
         END
       END
   END
END
```

Besides the two-equation Gamma Theta transition model, other reduced models are available:

- Specified Intermittency

A zero-equation model, where you can prescribe the intermittency directly as a CEL expression.

The best way to specify the intermittency is with a user defined subroutine that is based on the x, y and z coordinates. This way, conditional statements can be used to define geometric bounds where the intermittency can be specified as zero (laminar flow) or one (turbulent flow). This method can be used to prescribe laminar regions at the leading edges of the wings, for example.

- Gamma Model

  A one-equation model that solves only the intermittency equation (of the two-equation Gamma Theta transition model) using a user specified value of the transition onset momentum thickness-based Reynolds number.

- Intermittency Model

  A one-equation model. For details, see One Equation Intermittency Transition Model in the *CFX-Solver Theory Guide*.

Note:

- The Gamma Theta transition model is not Galilean invariant and should therefore not be applied (unless with caution) to walls that move relative to the coordinate system for which the velocity field is computed. For this purpose the Intermittency Model should be used.
- Both Gamma Theta and Intermittency models are currently not linked to the buoyancy production terms. Therefore the models should not be used in conjunction with non-zero buoyancy forces.

More details can be found in Ansys CFX Laminar-Turbulent Transition Models in the *CFX-Solver Theory Guide*.

The following topics are discussed:

4.1.11.1. Estimating when the Transition Model Should be Used
4.1.11.2. Grid Requirements
4.1.11.3. Specifying Inlet Turbulence Levels
4.1.11.4. Summary

4.1.11.1. Estimating when the Transition Model Should be Used

Because the transition model requires the solution of two extra transport equations, there are additional CPU costs associated with using it. A rough estimate is that for the same grid the transition model solution requires approximately 18 percent additional CPU time compared to a fully turbulent solution. As well, the transition model requires somewhat finer grids than are typically used for routine design purposes. This is because the max grid y^+ must be approximately equal to one (that is, wall-function grids cannot be used because they cannot properly resolve the laminar boundary layer) and sufficient grid points in the streamwise direction are needed to resolve the transitional region. For this reason, it is important to be able to estimate when the additional cost of using the transition model in terms of CPU and grid generation time is justified. The relative percentage of

laminar flow on a device can be estimated using the following formula, which is based on the Mayle [104] empirical correlation for transition onset.

$$\frac{Re_{xt}}{Re_x} = \frac{380000 \cdot \left(100 \cdot Tu\right)^{-\frac{5}{4}}}{\left(\rho / \mu\right) \cdot V \cdot L_{\text{Device}}} \tag{4.1}$$

Where Re_{xt} is the transition Reynolds number, Re_x is the device Reynolds number, L_{Device} is the length of the device, V is a representative velocity, and Tu is the freestream turbulence intensity, which can be calculated, as follows:

$$Tu = \frac{\left(2k / 3\right)^{0.5}}{V} \tag{4.2}$$

where k is the turbulent kinetic energy. The fraction of laminar flow for some representative devices is shown in Table 4.1: Fraction of laminar flow for a variety of different devices (p. 207). Clearly, there are many cases where the assumption of fully turbulent flow is not correct and a significant amount of laminar flow could be present.

Table 4.1: Fraction of laminar flow for a variety of different devices

Case	Re_x	Tu (%)	Fraction Laminar Flow
LP Turbine blade	150 000	2.5	0.80
HP Turbine blade	500 000	6.0	0.10
Compressor blade	1 000 000	1.0	0.38
Small Aircraft wing	5 000 000	0.2	0.57
F1 Race Car Spoiler	2 000 000	0.3	0.86

4.1.11.2. Grid Requirements

The effect of increasing and decreasing y^+ for a flat plate test case (T3A) is shown in Figure 4.1: Effect of increasing y+ for the flat plate T3A test case (p. 208) and Figure 4.2: Effect of decreasing y+ for the flat plate T3A test case (p. 208). For y^+ values between 0.001 and 1, there is almost no effect on the solution. Once the maximum y^+ increases above 8, the transition onset location begins to move upstream. At a maximum y^+ of 25, the boundary layer is almost completely turbulent. For y^+ values below 0.001, the transition location appears to move downstream. This is presumably caused by the large surface value of the specific turbulence frequency ω, which scales with the first grid point height. Additional simulations on a compressor test case have indicated that at very small y^+ values the SST blending functions switch to k-ε in the boundary layer. For these reasons, very small (below 0.001) y^+ values should be avoided at all costs.

Figure 4.1: Effect of increasing y+ for the flat plate T3A test case

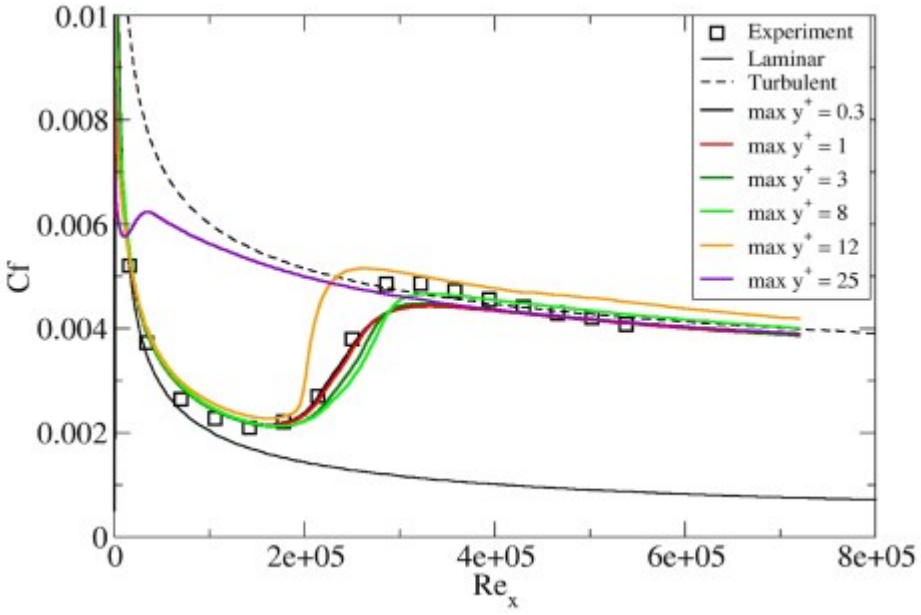
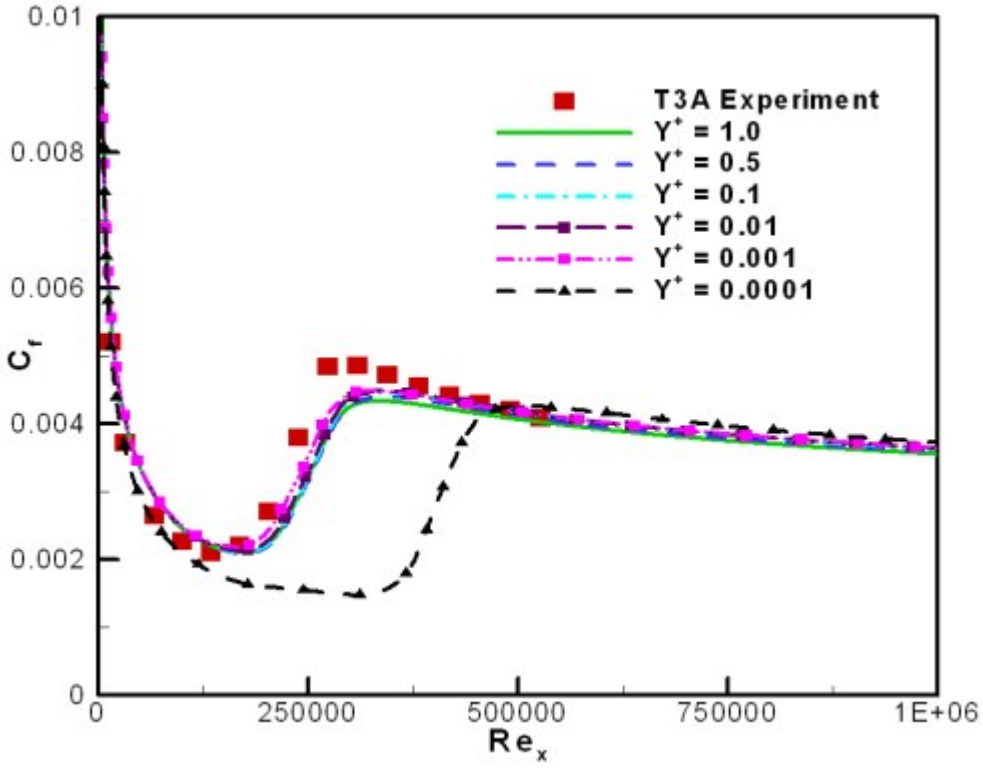


Figure 4.2: Effect of decreasing y+ for the flat plate T3A test case

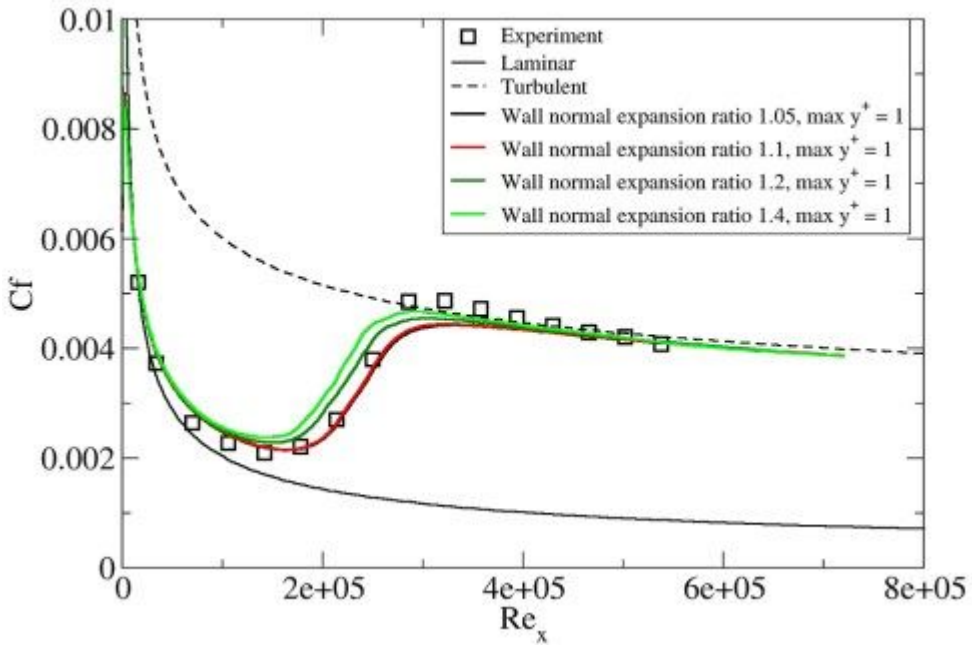


(Figure 4.2: Effect of decreasing y+ for the flat plate T3A test case (p. 208) provided by Likki, Suzen and Huang [102])

The effect of wall normal expansion ratio from a y^+ value of 1 is shown in Figure 4.3: Effect of wall normal expansion ratio for the flat plate T3A test case (p. 209). For expansion factors of 1.05 and 1.1, there is no effect on the solution. For larger expansion factors of 1.2 and 1.4, there is a small

but noticeable upstream shift in the transition location. Because the sensitivity of the solution to wall-normal grid resolution can increase for flows with pressure gradients, it is recommended that you apply grids with $y^+<1$ and expansion factors smaller than 1.1.

Figure 4.3: Effect of wall normal expansion ratio for the flat plate T3A test case



The effect of streamwise grid refinement is shown in Figure 4.5: Effect of streamwise grid density for the flat plate T3A test case (p. 211). Surprisingly, the model was not very sensitive to the number of streamwise nodes. The solution differed significantly from the grid independent one only for the case of 25 streamwise nodes where there was only one cell in the transitional region. Nevertheless, the grid independent solution appears to occur when there are approximately 75 - 100 streamwise grid points. As well, better streamwise resolution is most likely necessary if separation induced transition takes place. In this case, the number of nodes has to be sufficient to resolve the separation bubble. A general rule is that at least 20 nodes should cover the bubble length. For low Reynolds numbers (<105) the bubble size is comparable to the blade chord length and this requirement is easy to satisfy. Even the recommended 75-100 grid points from the leading to the trailing edge are typically sufficient. However, the bubble decreases as the Reynolds number increases; therefore the grid has to be refined in the regions of the expected separation. An example of how the mesh density influences the resolution of a separation bubble near the leading edge of a wind turbine airfoil is illustrated in Figure 4.4: Effect of streamwise grid density for resolving the separation-induced transition due to a leading edge separation bubble for a wind turbine airfoil (p. 210). The two meshes shown differ by the number of nodes in streamwise direction in the region covered by the bubble. On Mesh 1, the bubble is resolved by just a few nodes and appears to be much smaller compared to that for the finer mesh, Mesh 2. Clearly, if there are not enough streamwise nodes, the model cannot resolve the rapid separation-induced transition and the boundary layer on the suction side stays laminar, which might have a large influence on the lift coefficient and stall angle predictions.

Note that the high resolution advection scheme has been used for all equations including the turbulent and transition equations and this is the recommended default setting for transitional computations. In CFX, when the transition model is active and the high resolution scheme is selected for the hydrodynamic equations, the default advection scheme for the turbulence and transition equations is automatically set to high resolution.

Figure 4.4: Effect of streamwise grid density for resolving the separation-induced transition due to a leading edge separation bubble for a wind turbine airfoil

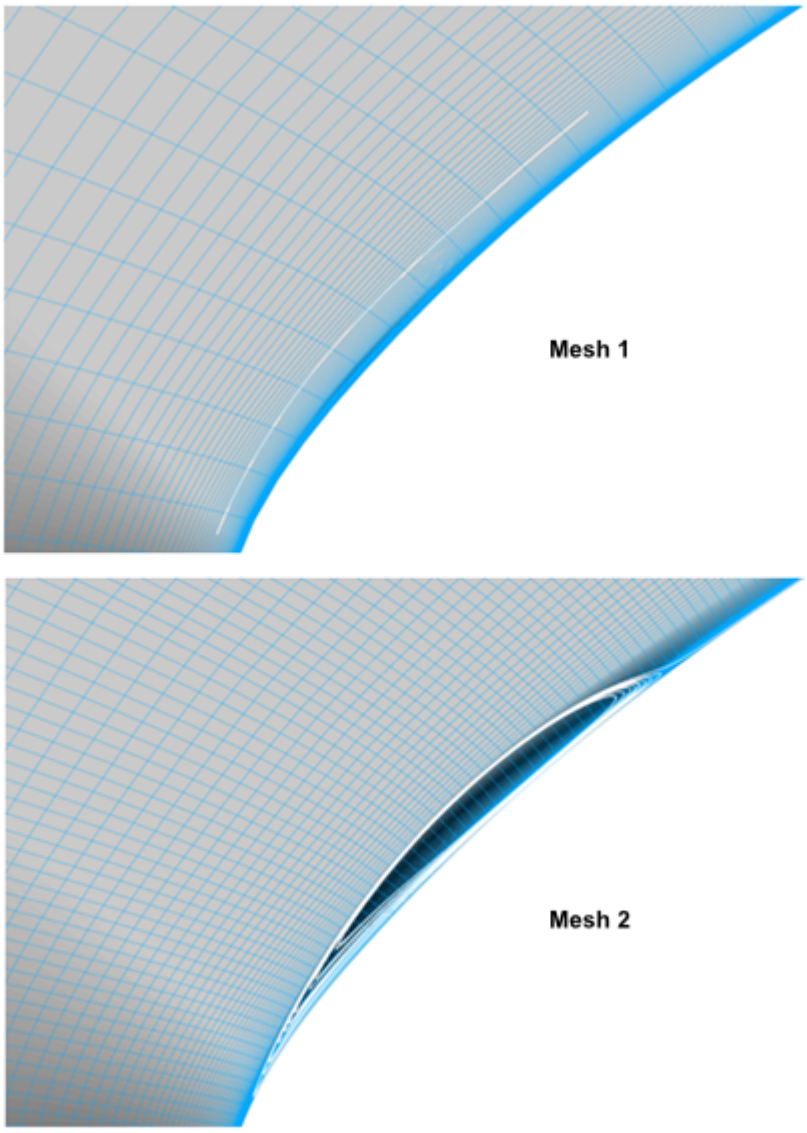
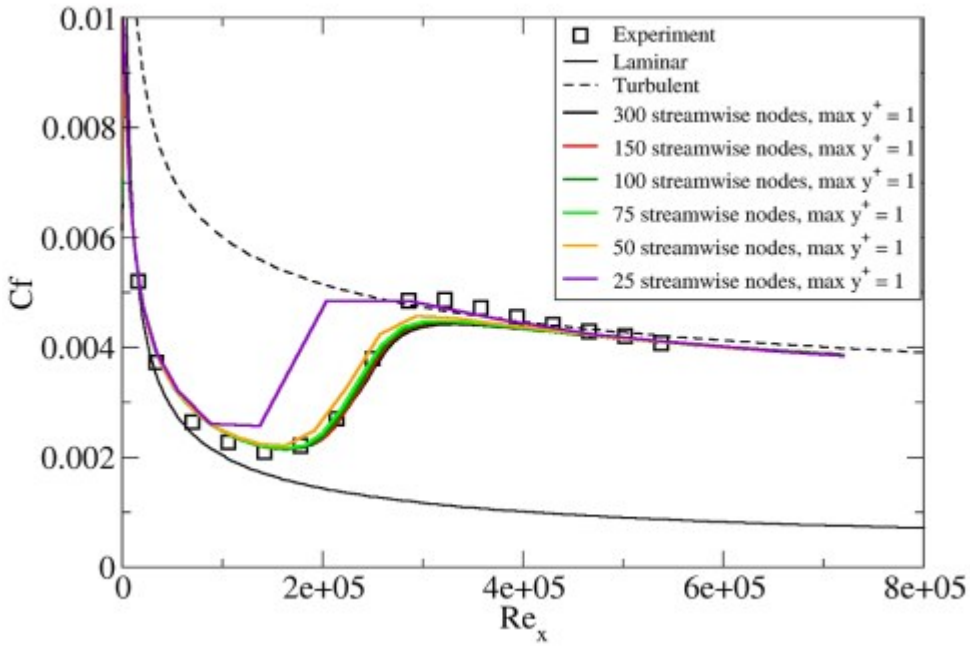


Figure 4.5: Effect of streamwise grid density for the flat plate T3A test case



One point to note is that for sharp leading edges, often transition can occur due to a small leading edge separation bubble. If the grid is too coarse, the rapid transition caused by the separation bubble is not captured.

Based on the grid sensitivity study the recommended best practice mesh guidelines are a max y^+ of 1, a wall normal expansion ratio of 1.1 and about 75 - 100 grid nodes in the streamwise direction. Note that if separation induced transition is present, additional grid points in the streamwise direction are most likely needed. For a typical 2D blade assuming an H-O-H type grid, the above guide lines result in an inlet H-grid of 15x30, an O-grid around the blade of 200x80 and an outlet H-grid of 100x140 for a total of approximately 30 000 nodes, which has been found to be grid independent for most turbomachinery cases. It should also be noted that for the surfaces in the out of plane z-direction, symmetry planes should always be used, not slip walls. The use of slip walls has been found to result in an incorrect calculation of the wall distance, which is critical for calculating the transition onset location accurately. Another point to note is that all the validation cases for the transition model have been performed on hexahedral meshes. At this point, the accuracy of the transition model on tetrahedral meshes has not been investigated.

4.1.11.3. Specifying Inlet Turbulence Levels

It has been observed that the turbulence intensity specified at an inlet can decay quite rapidly depending on the inlet viscosity ratio $\left(\mu_t / \mu\right)$ (and hence turbulence eddy frequency). As a result, the local turbulence intensity downstream of the inlet can be much smaller than the inlet value (see Figure 4.6: Decay of turbulence intensity (Tu) as a function of streamwise distance (x) (p. 212)). Typically, the larger the inlet viscosity ratio, the smaller the turbulent decay rate. However, if too large a viscosity ratio is specified (that is, >100), the skin friction can deviate significantly from the laminar value. There is experimental evidence that suggests that this effect occurs physically; however, at this point it is not clear how accurately the transition model reproduces this behavior. For this reason, if possible, it is desirable to have a relatively low (that is ≈1 - 10) inlet viscosity ratio

and to estimate the inlet value of turbulence intensity such that at the leading edge of the blade/airfoil, the turbulence intensity has decayed to the desired value.

The decay of turbulent kinetic energy can be calculated with the following analytical solution:

$$k=k_{\text{inlet}}\cdot\left(1+\omega_{\text{inlet}}\cdot\beta\cdot t\right)^{-\frac{\beta^*}{\beta}} \tag{4.3}$$

For the SST turbulence model in the freestream the constants are:

$$\beta=0.09 \quad \beta^*=0.0828 \tag{4.4}$$

The time scale can be determined as follows:

$$t=\frac{x}{V} \tag{4.5}$$

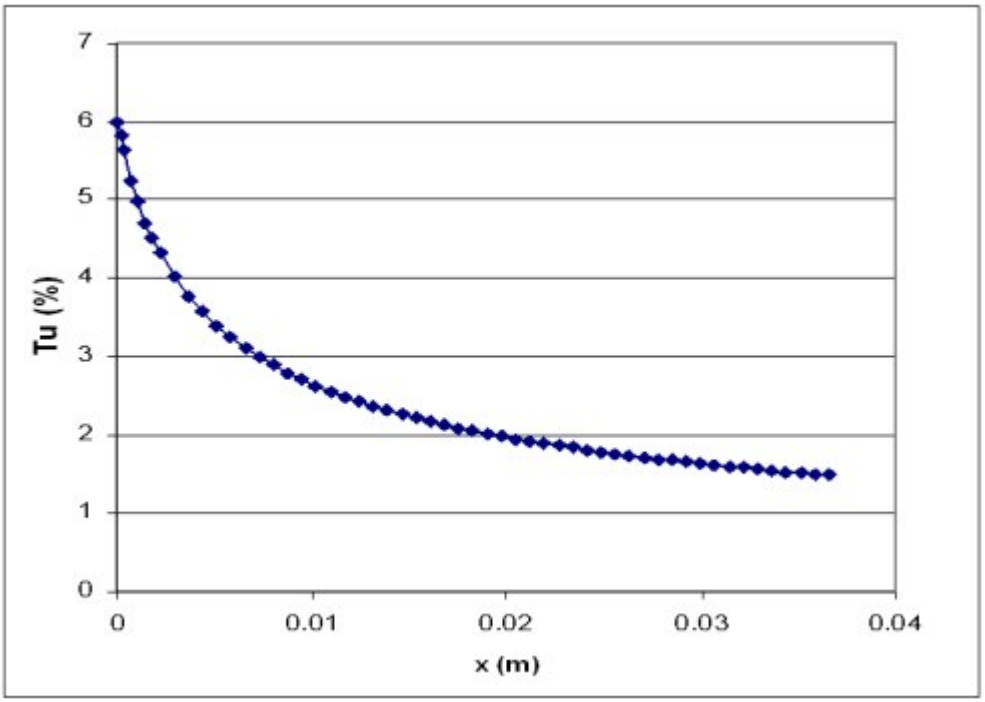
where x is the streamwise distance downstream of the inlet and V is the mean convective velocity. The eddy viscosity is defined as:

$$\mu_t=\frac{\rho\, k}{\omega} \tag{4.6}$$

The decay of turbulent kinetic energy equation can be rewritten in terms of inlet turbulence intensity (Tu_{inlet}) and eddy viscosity ratio ($\mu_t/\ \mu$) as follows:

$$Tu=\left(Tu_{\text{inlet}}^2\left[1+\frac{3\cdot\rho\cdot V\cdot x\cdot\beta\cdot Tu_{\text{inlet}}^2}{2\cdot\mu\left(\mu_t/\ \mu\right)}\right]^{-\frac{\beta^*}{\beta}}\right)^{0.5} \tag{4.7}$$

Figure 4.6: Decay of turbulence intensity (Tu) as a function of streamwise distance (x)



You should ensure that the Tu values around the body of interest roughly satisfy $\text{Tu}>0.1\%$. For smaller values of Tu, the reaction of the SST model production terms to the transition onset becomes too slow and transition can be delayed past the physically correct location.

Simulations of external aerodynamic flows typically require a computational domain with remote outer boundaries. In such cases, it might be necessary to specify excessively high values of turbulence intensity and eddy viscosity ratio at the inlet in order to get the desired value at the airfoil. In addition, numerical grids around aerodynamic bodies typically feature very coarse grids at the far-field boundary. For such coarse grids, Equation 4.7 (p. 212) is no longer accurately resolved and the numerical decay can differ substantially from the analytical one. Alternatively, sensible inlet values can be obtained if the onset of the Tu decay is shifted closer to the airfoil. This can be implemented via adding source terms into the k and ω equations, which keep the inlet values of the transported variables unchanged until the specified coordinate. These source terms read as:

$$S_k = \beta \rho k_{\text{inlet}} \omega F_{\text{step}}$$

$$S_\omega = \beta^* \rho \omega_{\text{inlet}} \omega F_{\text{step}}$$

$$F_{\text{step}} = \begin{cases} 1, x < x_0 \\ 0, x \geq x_0 \end{cases}$$

$$\beta = 0.09, \beta^* = 0.0828$$

where x_0 is an arbitrary coordinate between the computational domain inlet and the airfoil. The two source terms are actually the inverted sign sink terms for the SST model when it is working in the $k-\varepsilon$ regime. Adding these terms prevents k and ω values from undergoing any decay in the freestream until the x_0 location is reached. The streamwise distance in Figure 4.6: Decay of turbulence intensity (Tu) as a function of streamwise distance (x) (p. 212) counts then from the x_0 coordinate. More general F_{step} functions can be used, but it needs to be ensured that F_{step} is zero in the vicinity of the wall boundary layers around the aerodynamic body.

4.1.11.4. Summary

Proper grid refinement and specification of inlet turbulence levels is crucial for accurate transition prediction. In general, there is some additional effort required during the grid generation phase because a low-Re grid with sufficient streamwise resolution is needed to accurately resolve the transition region. As well, in regions where laminar separation occurs, additional grid refinement is necessary in order to properly capture the rapid transition due to the separation bubble. Finally, the decay of turbulence from the inlet to the leading edge of the device should always be estimated before running a solution as this can have a large effect on the predicted transition location.

4.1.12. The Large Eddy Simulation Model (LES)

4.1.12.1. Using the LES model in CFX

Turbulent flows contain a wide range of length and time scales, with large scale motions being generally more energetic than the small scale ones. The prediction of industrially important fluctuating flow problems can be performed using the technique of LES. LES is an approach that solves for large-scale fluctuating motions and uses "sub-grid" scale turbulence models for the small-scale motion.

4.1.12.2. Introduction to LES

The usual approach to predicting turbulent flows is to use the Reynolds Averaged Navier-Stokes equations, (RANS), which solve for time averaged quantities. However, there are some situations where the approach is not adequate, and the alternative approaches of Large Eddy Simulation (LES) or Direct Numerical Simulation (DNS) can be adopted. With these methods, time dependent equations are solved for the turbulent motion with either no approximations and all relevant scales resolved (DNS) or the equations are filtered in some way to remove very fine time and length scales (LES). These approaches require fine grids and small timesteps, particularly for wall bounded flows, as well as a large number of timesteps to generate statistically meaningful correlations for the fluctuation velocity components. However, they can give details on the structure of turbulent flows, such as pressure fluctuations and Lighthill stresses, which cannot be obtained from a RANS formulation.

Further theoretical model information is available in Large Eddy Simulation Theory in the *CFX-Solver Theory Guide*.

4.1.12.3. When to use LES

Before starting an LES simulation, you should consider if it is the most appropriate solution approach. For low Reynolds numbers ($Re < 5000$) or when it is important to be able to resolve all scales (for example, for transition to turbulent flow), consider DNS if you have a large computer available. For Higher Reynolds numbers, LES might be a suitable option for cases where:

- The flow is likely to be unstable, with large scale flapping of a shear layer or vortex shedding.
- The flow is likely to be unsteady with coherent structures (cyclone, flasher).
- The flow is buoyant, with large unstable regions created by heating from below, or by lighter fluid below heavier fluid (multiphase flows in inclined pipelines).
- Conventional RANS approach are known to fail (for example due to highly anisotropic turbulence).
- A good representation of the turbulent structure is required for small-scale processes such as micro-mixing or chemical reaction.
- The noise from the flow is to be calculated, and especially when the broadband contribution is significant.
- Other fluctuating information is required (such as fluctuating forces, gusts of winds).
- You can afford to wait for up to a week for results, using an 8 to 16 processor system.

It should be noted that for wall bounded flows, so called 'streaky structures' develop in the near wall region. These 'streaky structures' must be resolved and this leads to high resolution requirements and computing times for LES of wall-bounded flows. This should be kept in mind and you should consider the SAS (Scale-Adaptive Simulation) or DES (Detached Eddy Simulation) approach first before performing a LES for wall bounded flows.

4.1.12.4. Setting up an LES Simulation

4.1.12.4.1. Geometry for LES

Even though there might be symmetries in the geometry and flow, the geometrical model should include the full region because, while the time averaged flows may be symmetric, the instantaneous flows are not, and restricting the domain could constrain the turbulence. Two-dimensional approximations are particularly poor.

4.1.12.4.2. Meshing

The mesh and timesteps are an inherent part of the model. LES models make use of the grid scale for filtering out the turbulence. If the mesh is anisotropic to resolve a jet, for example, the longer length scale in the flow direction may also have an undue effect on the cross-stream turbulence. For this reason, consider the use of isotropic grids, perhaps using tetrahedral rather than hexahedral elements.

4.1.12.4.3. Boundary Layers

If the boundary layer structure is important, you should resolve it with at least 15 points across the boundary layer and with the first grid point at a position of approximately $y^+ = 1$. Note however, that large aspect rations in the mesh are not suitable for LES. This often results in excessive resolution requirements for boundary layer flows.

4.1.12.4.4. Analysis Type

In CFX-Pre set the **Steady-State/Transient** setting to `Transient`.

4.1.12.4.5. Domains

Three LES models are available: it is recommended that you use the wall-adapted local eddy-viscosity model by Nicoud and Ducros [200](LES WALE model) as a first choice. This is an algebraic model like the Smagorinsky model, but overcomes some known deficiencies of the Smagorinsky model: the WALE model produces almost no eddy-viscosity in wall-bounded laminar flows and is therefore capable to reproduce the laminar to turbulent transition. Furthermore, the WALE model has been designed to return the correct wall-asymptotic y^{+3}-variation of the subgrid-scale viscosity and needs no damping functions.

In addition to the WALE model, the Smagorinsky model [34] and the Dynamic Smagorinsky-Lilly model (Germano et al. [198], Lilly [199]) are available. The Dynamic Smagorinsky-Lilly model is based on the Germano-identity and uses information contained in the resolved turbulent velocity field to evaluate the model coefficient, in order to overcome the deficiencies of the Smagorinsky model. The model coefficient is no longer a constant value and adjusts automatically to the flow type. However this method needs explicit (secondary) filtering and is therefore more time consuming than an algebraic model. Details on the LES models can be found in Large Eddy Simulation Theory in the *CFX-Solver Theory Guide*.

4.1.12.4.6. LES Boundary Conditions: Inlet

The representation of the turbulent structures at inlets can be a difficult part of the setup. The detailed properties of the incoming flow can have a strong effect on the development of the downstream solution. This has been observed in turbulent jets, as well as developing boundary

layer cases. For other cases, however, the turbulent conditions at the inlet can have relatively little impact on the flow in the device. For example, in a cyclone body, the flow can be essentially determined by very strong anisotropy effects.

If the inlet turbulence is felt to play an important role, you should consider:

1. Moving the inlet far upstream of the geometry of interest. This allows the correct turbulent structures to establish themselves before the flow reaches that geometry. Additional obstructions can be placed upstream of geometry to affect the turbulent structures. This procedure, however, will require the simulation of the transition process.

2. Using unsteady values computed from a separate LES simulation (pipe flows and channel flows).

4.1.12.4.7. LES Boundary Conditions: Outlets and Openings

With the transport of turbulent eddies outside of the computational domain, some recirculation can occur at outlets. Experience shows that if the code is allowed to bring some fluid back into the domain at outlets, it can destabilize the solution. Therefore, you should use outlet boundary conditions rather than openings when using LES. Opening types of boundary conditions enable the flow to come back in, whereas with outlet boundary conditions, the code builds artificial walls at the boundary, when the flow tries to come back in. These artificial walls are later taken away when the flow goes out again.

The use of artificial walls at outlets might introduce some un-physical behavior locally, but it increases the robustness of the calculation.

4.1.12.4.8. LES Initialization

For simulations initialized with values (rather than from an initial guess field), it is possible to perturb the initial guess by setting an RMS `Velocity Fluctuation`. This has the effect of kickstarting the solution process. For a velocity distribution V_i, the fluctuation is the quantity:

$$V_{\mathrm{rms}}=\sqrt{\sum_{i=1}^{N}\frac{\left(V_i-\overline{V}\right)^2}{N}} \tag{4.8}$$

The behavior of the velocity fluctuation is applied according to the following rules when you select `Automatic` or `Automatic with Value` as the **Initialization** option, as well as specify a velocity fluctuation:

- If an initial values file is not found, the fluctuation is applied to the initial velocity field.
- When an initial values file is used, it is assumed that you are restarting from a previous set of results, and the fluctuation is not applied.

If you want to apply a velocity fluctuation on a restart (for example, if the initial guess is a RANS solution), you can override this behavior by setting the expert parameter `apply ic fluctuations for les` to `t`.

4.1.12.4.9. LES Solver Control

On the **Solver Control** form, it is recommend that you use the **Central Difference** advection scheme rather than the **High Resolution** scheme because it is less dissipative and it has provided good answers in CFX. Select the transient scheme as `2nd Order Backward Euler`.

4.1.12.4.10. LES Timestep Considerations

- First order fully implicit methods in time are usually too diffusive, and the turbulence is damped out. For highly unstable problems, such as cyclones, lower order methods may work, but the results will be very damped, unless very small timesteps are used.
- For accuracy, the average Courant (or CFL) number should be in the range of 0.5-1. Larger values can give stable results, but the turbulence may be damped. For compressible flows where the acoustic behavior is being modeled (eg, for noise calculations), this conclusion still holds, but for the CFL number based on the acoustic velocity as well as the convective velocity.
- 1,000 - 10,000 timesteps are typically required for getting converged statistics. More steps are required for second order quantities (for example, variances) than for means. Check the convergence of the statistics. For a vortex-shedding problem, several cycles of the vortex shedding are required.
- The implicit coupled solver used in CFX requires the equations to be converged within each timestep to guarantee conservation. The number of coefficient loops required to achieve this is a function of the timestep size. With CFL numbers of order 0.5-1, convergence within each timestep should be achieved quickly. It is advisable to test the sensitivity of the solution to the number of coefficient loops, to avoid using more coefficient loops (and hence longer run times) than necessary. LES tests involving incompressible flow past circular cylinders indicates that one coefficient loop per timestep is sufficient if the average CFL number is about 0.75. If the physics or geometry is more complicated, additional coefficient loops (3-5) may be required. Timesteps sizes that require more coefficient loops than this will tend to degrade solution accuracy.

4.1.12.5. Solver Memory

You may find, especially if running averages are to be calculated, that for some cases you will need to increase the amount of memory needed for the calculation by using a memory factor above 1 (typically 1.2).

4.1.12.6. Useful Values For LES Runs

The following values can be written to results files as the solution progresses using the **Solver Output** tab in CFX-Pre. For details, see Output Control in the *CFX-Pre User's Guide*.

Although the statistics outlined in the Statistical Reynolds Stresses are useful for any simulation, they are particularly important in the context of LES. Of particular interest, are the Reynolds Stress (RS) components:

$$\rho \overline{u_i' u_j'} \tag{4.9}$$

where u_i' is the fluctuation of the i^{th} velocity component. For details, see Statistical Reynolds Stresses (p. 218).

4.1.12.6.1. Statistical Reynolds Stresses

In LES runs, Reynolds Stress components are automatically generated using running statistics of the instantaneous, transient velocity field. As outlined above for the calculation of the standard deviation, a Reynolds Stress component can be evaluated using the difference between the running arithmetic average of the instantaneous velocity correlation and the running arithmetic average of the instantaneous velocities as:

$$\overline{u_i' u_j'} = \overline{u_i u_j} - \overline{u_i}\,\overline{u_j} \tag{4.10}$$

The noted running averages are also automatically generated for LES runs.

4.1.12.6.2. Delaying the Start of Reynolds Stress Calculations

Data sampling for the Statistical Reynolds Stresses begins on the first timestep by default. This is because the transient statistics used in the stress evaluation (that is, the arithmetic averages of velocity and its correlation) start accumulating on the first timestep by default.

Sampling for the Statistical Reynolds Stresses is deferred, indirectly, by explicitly creating one of the required transient statistics (that is, the arithmetic averages of velocity or its correlation) and setting the start iteration (the starting timestep index) to a value larger than unity. If different start iterations are set for each of the velocity or velocity correlation averages, then the maximum start iteration set is used for both averages to ensure that the Statistical Reynolds Stresses are correctly evaluated.

The Statistical Reynolds Stress variable is evaluated using Equation 4.10 (p. 218) during every timestep, regardless of the start iteration specified for the required velocity-based statistics. This means that if the current timestep is less than the start iteration specified for the velocity-based statistics, then those statistics will be initialized as outlined in Working with Transient Statistics in the *CFX-Pre User's Guide*.

4.1.13. The Detached Eddy Simulation Model (DES)

Note:

The classical DES model family has been superseded by the newer model formulations of the SBES family (The Stress-Blended Eddy Simulation (SBES) Model (p. 223)).

In an attempt to improve the predictive capabilities of turbulence models in highly separated regions, Spalart [57] proposed a hybrid approach, which combines features of classical RANS formulations with elements of Large Eddy Simulations (LES) methods. The concept has been termed Detached Eddy Simulation (DES) and is based on the idea of covering the boundary layer by a RANS model and switching the model to a LES mode in detached regions. Ideally, DES would predict the separation line from the underlying RANS model, but capture the unsteady dynamics of the separated shear layer by resolution of the developing turbulent structures. Compared to classical LES methods, DES saves orders of magnitude of computing power for high Reynolds number flows. Though this is due to the moderate costs of the RANS model in the boundary layer region, DES still offers some of the advantages of an LES method in separated regions.

See Detached Eddy Simulation Theory in the *CFX-Solver Theory Guide* and the detailed CFX report [55] for theoretical model information.

Additional modeling advice can be found in Best Practices: Scale-Resolving Simulations in Ansys CFD in the *CFX Reference Guide*.

4.1.13.1. Using the Detached Eddy Simulation Model in CFX

4.1.13.1.1. When to use DES

It is well known that RANS models does not accurately predict all flow details in massively separated flow regions. In addition, the RANS formulation does not provide any information on turbulent flow structures and spectral distribution, which might be of importance to predict flow-induced noise or vibrations. In these cases, DES can provide valuable details far exceeding RANS simulations. Typical examples of DES simulations are:

- Flow around non-aerodynamic obstacles (buildings, bridges, and so on)
- Flow around ground transport vehicles with massively separated regions (cars, trains, trucks)
- Flow around noise generating obstacles (such as car side mirrors)
- Massively separated flow around stalled wings

On the other hand, DES is a computer intensive method because large (detached) turbulent structures need to be resolved in space and time. While the grid resolution requirements are not significantly higher than RANS for simulations, the time resolution imposes high CPU demands. In CFX, steady-state solutions can typically be obtained within 100-200 iterations. A typical DES simulation requires several thousands timesteps using three coefficient loops each. DES is therefore at least one order of magnitude more computer intensive than RANS models. DES does not enable the use of symmetry conditions, because turbulent structures are usually not symmetric. Two-dimensional and axisymmetric simulations are not feasible, as turbulent structures are always three-dimensional. Hence, you should carefully weigh the need for additional information against the required resources.

4.1.13.2. Setting up a DES Simulation

4.1.13.2.1. Geometry for DES

The main issue with geometry is that 2D or axisymmetric simulations are not visible. In addition, in most cases, symmetry or periodicity conditions are not allowed. As an example, RANS simulations enable the use of a half model for a car simulation. As the turbulent structures are asymmetric, this is not possible in DES.

4.1.13.2.2. Meshing Requirements for DES

There are many requirements in terms of grid generation, which depend on the selection of the specific formulation of the DES model (see report [55] or Turbulence and Wall Function Theory in the *CFX-Solver Theory Guide*). The grid resolution requirement in the attached boundary layers is the same as that of a regular RANS solution (10-15 nodes in the boundary layer). In the DES region, the grid has to be designed in all three space dimensions to enable a minimum resolution equivalent to the resolution of the largest turbulent structures. Because the largest grid spacing is used in the switch for the DES limiter, it is essential to have sufficient resolution in the spanwise direction (if it exists).

One of the main issues of DES models is the danger to produce grid-induced separation due to an activation of the DES limiter in the RANS region. For a detailed explanation, refer to [55]. The zonal formulation used in CFX helps to avoid the severity of this potential limitation.

Examples of DES grids are provided in [55]. For a cylinder in cross-flow, 50 nodes were used in the spanwise direction for an extension of 2 diameters. For the Ahmed car body, 70 nodes were used in the cross-flow direction.

4.1.13.2.3. DES Timestep Considerations

Timestep selection is a sensitive issue, as it determines the quality of the resolution of the turbulent structures. In most cases, a typical shedding frequency can be estimated for the largest turbulent structures. An example is a cylinder in cross-flow. For a cylinder, the Strouhal number is of the order of:

$$S_t = \frac{f\,D}{U} \sim 0.2 \tag{4.11}$$

In the CFX simulation [55], the freestream velocity is U=46 m/s and the diameter is D=0.37 m. This results in a frequency f=24.8 1/s. A timestep of $D_t = 10^{-4}\,s$ is sufficient to resolve the turbulent structures on the given grid.

4.1.13.2.4. Boundary Conditions

In almost all cases, the same boundary conditions as in RANS simulation can be applied. Exceptions are symmetry or periodicity conditions, which may not be satisfied by the turbulent structures.

For a detailed discussion of inlet conditions, see [55]. In most cases, RANS inlet conditions can be applied, particularly for flows without profile distributions at the inlet, as mostly used in industrial flows.

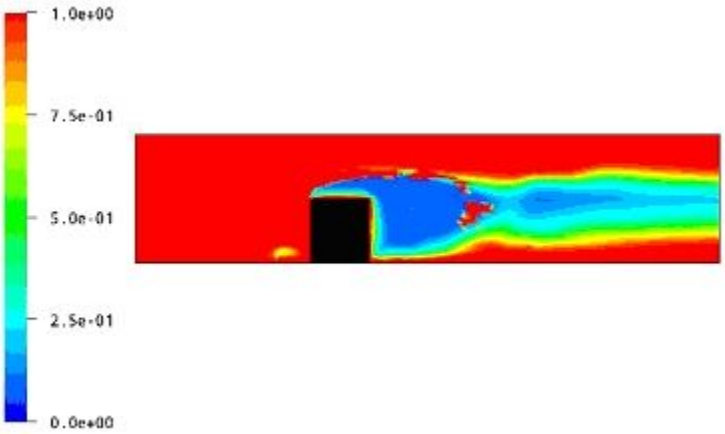
4.1.13.2.5. DES Initialization

It is recommended that you start the DES simulation from a converged RANS solution. For comparison purposes, it is useful to use the same RANS model as in the DES formulation (in CFX this means the SST model). The RANS solution supplies to the DES formulation important elliptic information on the pressure field, which might take a long time to build up in an unsteady simulation.

4.1.13.2.6. Monitoring a DES Simulation

Due to high computing costs, it is essential to monitor a DES run to ensure that it approaches an appropriate solution. It is recommended that you check the blending function for DES model during the simulation (every 50-100 timesteps). In regions where the function is zero, the LES model is used, and the region where its value is one, the RANS model is activated. Figure 4.7: Blending Function for DES Model for Flow Around Cube. (p. 221) shows a typical example for the flow around a cube. The blue region behind the cube is governed by the LES part, and the red region by the RANS model. This is desirable for this case as only the structures behind the obstacle are of interest.

Figure 4.7: Blending Function for DES Model for Flow Around Cube.



A second variable to monitor is the invariant:

$$I=\Omega^2-S^2 \tag{4.12}$$

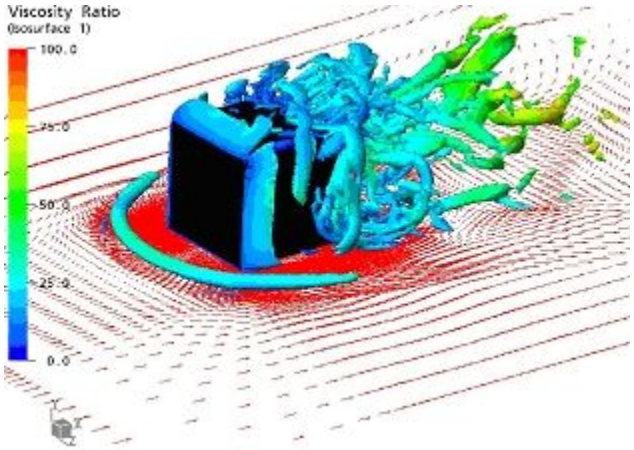
where S is the absolute value of the strain rate (S = Shear Strain Rate in CFD-Post) and Ω is the absolute value of the vorticity. Note that the vorticity has to be activated in order to be available in CFD-Post. The following lines are to be added to the solver CCL:

```
LIBRARY:
   VARIABLE: vorticity
      Output to Postprocessor = Yes
   END
END
```

In CFD-Post, you have to define a user variable based on the following expression:

```
LIBRARY:
   CEL:
      EXPRESSIONS:
         Invariant = Vorticity^2 - Shear Strain Rate^2
      END
   END
END
USER SCALAR VARIABLE:VelGradInvariant
   Boundary Values = Conservative
   Expression = Invariant
   Recipe = Expression
END
```

Figure 4.8: Strain Rate Invariant for Flow Around Cube



The above figure shows a typical picture of a successful DES simulation. In case that only single scale (or no) vortex shedding is observed, it might be necessary to consider the following options:

- Reduce timestep
- Refine grid
- Use F_{SST}=F1 or even F_{SST}=0 (see report [55] or Turbulence and Wall Function Theory in the *CFX-Solver Theory Guide*).

After the turbulent structures are visually established, the averaging process within CFX can be started. Variables of interest are vorticity, eddy viscosity, pressure, velocity and k and ω. This is achieved by the following steps (for example, start averaging at timestep 2087):

```
FLOW:
   OUTPUT CONTROL:
      TRANSIENT STATISTICS: mean1
         Output Variables List = Vorticity, Eddy Viscosity, \
         Pressure, Velocity, \
         Turbulence Eddy Frequency, \
         Turbulence Kinetic Energy
         Option = Arithmetic Average
         Start Iteration List = 2087,2087,2087,2087,2087,2087
      END
   END
END
```

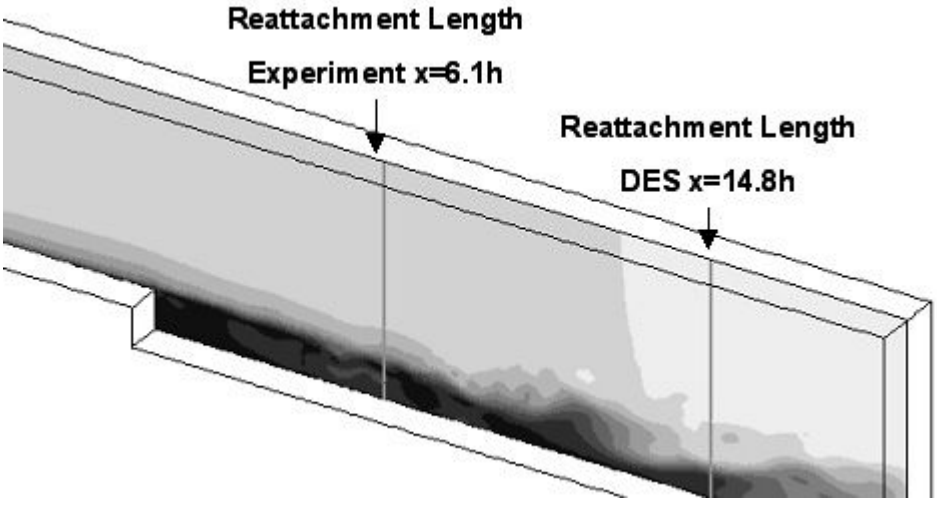
4.1.13.3. Limitations/Concerns of Using the DES Model

One of the requirements for DES is that the flow must develop a strong instability once the LES mode is activated. Otherwise the RANS model is reduced, but the unsteady structures do not develop quickly enough to keep the overall turbulent kinetic energy (resolved + modeled) consistent. Such a situation occurs inside boundary layers or channel/pipe flows, if the grid is refined below a critical level. Once this level is reached, the RANS model is reduced, but in most cases no instability develops quickly enough to compensate for the lowered stresses. In severe cases, this can result in grid-induced separation, as shown by Menter et al. [133]. For industrial applications this is a practical concern, as standard DES models have a RANS grid limit of $\Delta_{max}>\delta$, where Δ_{max} is the maximal local cell length and δ is the boundary layer thickness. This limit can easily be reached in industrial CFD

simulations. In order to reduce this problem, Menter [133] proposed shielding functions that protect the RANS boundary layer from grid impact.

A second requirement for DES is that the grid spacing in the LES region must immediately be of LES-quality, as otherwise the turbulence model will produce a mix of RANS and LES components ([135], [137]). A good example is the flow over a backward-facing step. Assume that the grid is of LES-quality behind the step in the main flow plane (x,y) and that the "spanwise" resolution defines the DES limiter, $\Delta_{\max}=\Delta_z$. In the case that Δ_z is larger than an upper critical limit $\Delta_{\max}>\Delta_{RANS}$, the model will operate in RANS mode and produce a steady-state solution. In the case that Δ_z is lower than a lower critical limit $\Delta_{\max}<\Delta_{LES}$, the model will operate in LES mode and develop a flow instability past the step with the correct balance of the stresses. However, if $\Delta_{RANS}>\Delta_{\max}>\Delta_{LES}$, the model is undefined, and the solution will be neither RANS nor LES. The most likely scenario is that starting from $\Delta_{\max}=\Delta_{RANS}$, refinement will impact the underlying RANS model, without enabling an unsteady solution to develop. As a result, the separation zone will grow much larger than in the experiments. From a certain point of refinement, unsteady structures will develop, but instead of starting at the step, they will start further downstream, where the DES limiter has a stronger influence. Again, this results in an overly large separation bubble. With further grid refinement, the unsteadiness will move further upstream and eventually the entire separation zone will be computed in LES mode, when $\Delta_{\max}<\Delta_{LES}$ is reached. The situation for $\Delta_{RANS}>\Delta_{\max}>\Delta_{LES}$ is shown in Figure 4.9: Unsteady flow from DES over backward facing step using insufficient spanwise grid resolution (p. 223) where the backward facing step of Jovic and Driver [138] was computed with the SST-DES model with a grid with only 21 cells in spanwise (2xH - H-step height) direction. The turbulent structures appear much too late and time-averaged flow reattachment takes place at $L_{reattach}$~15H.

Figure 4.9: Unsteady flow from DES over backward facing step using insufficient spanwise grid resolution



4.1.14. The Stress-Blended Eddy Simulation (SBES) Model

One of the weaknesses of the original DES model family is that it shields attached boundary layers insufficiently from the impact of the grid-dependent term. For this reason, the Stress Blended Eddy Simulation (SBES) model has been developed. This model features a much improved shielding function to protect RANS boundary layers.

The SBES model uses that shielding function, which is also utilized to switch to an existing algebraic LES model in the LES zone. The advantage is that you can clearly distinguish where the RANS and

LES zones are (by visualizing the shielding function) and which model is used in which zone. Because SBES blends between two existing models, it is not a true turbulence model, but rather an automated way of switching between existing RANS and LES models. The switching function is unpublished.

The SBES model "transitions" much quicker from RANS to Scale Resolving Simulation (SRS) mode in separating shear layers. This leads to more realistic solutions with higher internal consistency. In addition, this model allows for a RANS-LES "transition" on much coarser grids than classical DES.

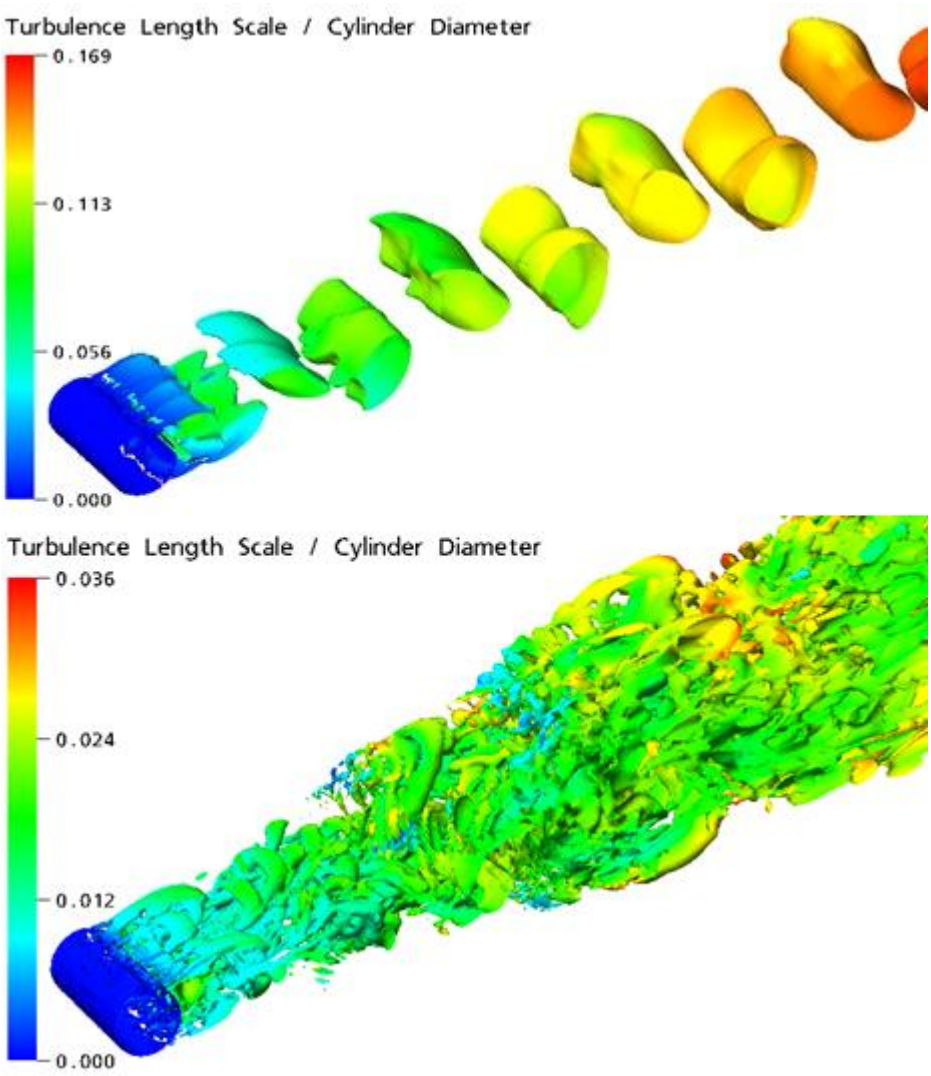
For SBES implementation details, see Stress-Blended Eddy Simulation (SBES) in the *CFX-Solver Theory Guide*.

4.1.15. The Scale-Adaptive Simulation (SAS)

The Detached Eddy Simulation is a combination of RANS and LES methods. In an alternative approach, a new class of URANS models has been developed that can provide a LES-like behavior in detached flow regions (Menter et al. [132], Menter and Egorov [130], Menter and Egorov [131], Egorov and Menter [197]). The Scale-Adaptive Simulation (SAS) concept is based on the introduction of the von Karman length-scale into the turbulence scale equation. The information provided by the von Karman length-scale allows SAS models to dynamically adjust to resolved structures in a URANS simulation, which results in a LES-like behavior in unsteady regions of the flowfield. At the same time, the model provides standard RANS capabilities in stable flow regions.

Figure 4.10: Resolved structures for cylinder in cross flow (top: URANS; bottom: SAS-SST) (p. 225) shows isosurfaces where $S^2-\Omega^2=10^5$ [s^-2] for a cylinder in cross flow (Re=3.6x10^6), calculated using the SST model (URANS) and the SAS-SST model. The URANS simulation produces only the large-scale unsteadyness, whereas the SAS-SST model adjusts to the already resolved scales in a dynamic way and allows the development of a turbulent spectrum in the detached regions.

Figure 4.10: Resolved structures for cylinder in cross flow (top: URANS; bottom: SAS-SST)



The original version of the SST-SAS model (Menter and Egorov [131]) has undergone continued evolution. This model is still available in Ansys CFX, but the latest version of the SAS model (Egorov and Menter [197]) is now the default model. More details can be found in Scale-Adaptive Simulation Theory in the *CFX-Solver Theory Guide*.

The SAS method is an improved URANS formulation, with the ability to adapt the length scale to resolved turbulent structures. In order for the SAS method to produce a resolved turbulent spectrum, the underlying turbulence model has to go unsteady. This is typically the case for flows with a global instability, as observed for flows with large separation zones, or vortex interactions. There are many technical flows, where steady-state solutions cannot be obtained or where the use of smaller timesteps results in unsteady solutions. One could argue that the occurrence of unsteady regions in a RANS simulation is the result of a non-local interaction, which cannot be covered by the single-point RANS closure. All these flows will benefit from the SAS methodology.

On the other hand, there are classes of flows where the SAS model will return a steady solution. The two main classes are attached or mildly separated boundary layers and undisturbed channel/pipe flows. These flows (or areas in a complex flow domains) will be covered by the RANS formulation.

This behavior of SAS is ideal for many technical applications, as flows for which the model produces steady solutions are typically those, where a well-calibrated RANS model can produce accurate results.

Contrary to DES, SAS cannot be forced to go unsteady by grid refinement. However, it has been shown in the backstep simulation above that enforcing unsteadiness by grid refinement does require an in-depth understanding of the flow, the turbulence model and its interaction with the grid to produce reliable results. Nevertheless, there will be cases where the grid sensitivity of DES will be advantageous, as it allows unsteady simulations where SAS might produce a steady-state flow-field.

This leads to the following hierarchy of models of increased complexity, which can be used as a guideline in order to estimate the needed effort:

1. URANS
2. SAS
3. DES
4. LES

4.1.15.1. Using the Scale Adaptive Simulation model in CFX

Due to the similar functionality of the SAS-SST and DES model, most of the guidelines that have been given for the DES model regarding timestepping, boundary conditions and initialization are also valid for the SAS-SST model.

4.1.16. Buoyancy Turbulence

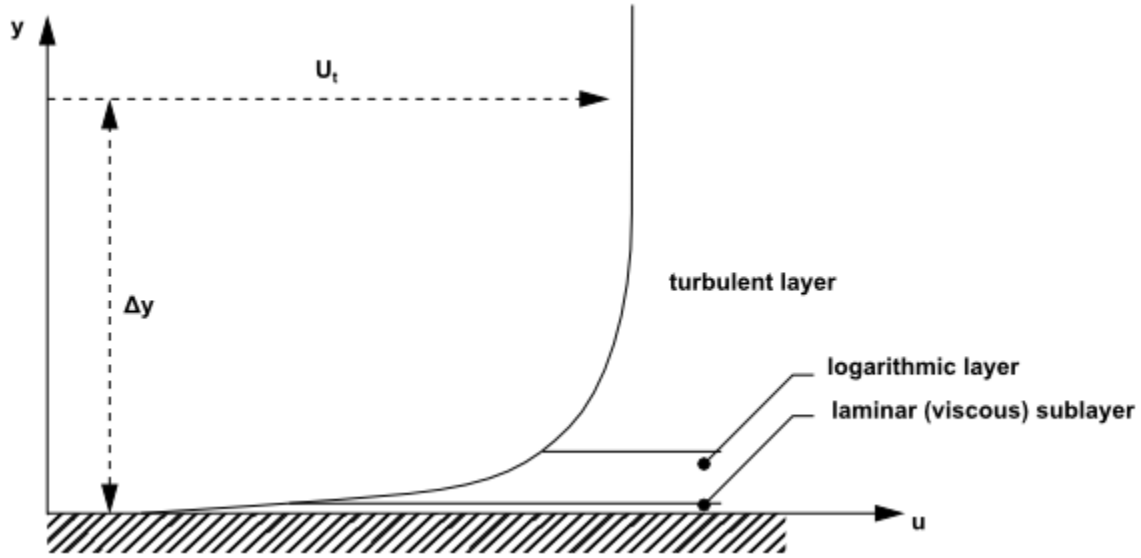
`Buoyancy Turbulence` is an option in CFX-Pre that is available for some turbulence models when buoyancy is being modeled. When the option is set to `Production`, a buoyancy production term is included in the equation for k. When the option is set to `Production and Dissipation`, a buoyancy production term is included in the equations for k, ε, ω (as applicable). See Buoyancy Turbulence and the accompanying text, which describe the treatment for the k-ε and k-ω models.

4.2. Modeling Flow Near the Wall

Near a no-slip wall, there are strong gradients in the dependent variables. In addition, viscous effects on the transport processes are large. The representation of these processes within a numerical simulation raises the following problems:

- How to account for viscous effects at the wall.
- How to resolve the rapid variation of flow variables that occurs within the boundary layer region.

Experiments and mathematical analysis have shown that the near-wall region can be subdivided into two layers. In the innermost layer, the so-called viscous sublayer, the flow is almost laminar-like, and the (molecular) viscosity plays a dominant role in momentum and heat transfer. Further away from the wall, in the logarithmic layer, turbulence dominates the mixing process. Finally, there is a region between the viscous sublayer and the logarithmic layer called the buffer layer, where the effects of molecular viscosity and turbulence are of equal importance. The figure below illustrates these subdivisions of the near-wall region.



Assuming that the logarithmic profile reasonably approximates the velocity distribution near the wall, it provides a means to numerically compute the fluid shear stress as a function of the velocity at a given distance from the wall. This is known as a 'wall function' and the logarithmic nature gives rise to the well known 'log law of the wall.'

Two approaches are commonly used to model the flow in the near-wall region:

- The wall function method uses empirical formulas that impose suitable conditions near the wall without resolving the boundary layer, thus saving computational resources. All turbulence models in CFX are suitable for a wall function method.

  The major advantages of the wall function approach is that the high gradient shear layers near walls can be modeled with relatively coarse meshes, yielding substantial savings in CPU time and storage. It also avoids the need to account for viscous effects in the turbulence model.

- The Low-Reynolds-Number method resolves the details of the boundary layer profile by using very small mesh length scales in the direction normal to the wall (very thin inflation layers). Turbulence models based on the ω-equation, such as the SST or SMC-ω models, are suitable for a low-Re method. Note that the low-*Re* method does *not* refer to the device Reynolds number, but to the turbulent Reynolds number, which is low in the viscous sublayer. This method can therefore be used even in simulations with very high device Reynolds numbers, as long as the viscous sublayer has been resolved.

  The computations are extended through the viscosity-affected sublayer close to the wall. The low-*Re* approach requires a very fine mesh in the near-wall zone and correspondingly large number of nodes. Computer-storage and run-time requirements are higher than those of the wall-function approach and care must be taken to ensure good numerical resolution in the near-wall region to capture the rapid variation in variables. To reduce the resolution requirements, an automatic wall treatment was developed by CFX, which allows a gradual switch between wall functions and low-Reynolds number grids, without a loss in accuracy.

Wall functions are the most popular way to account for wall effects. In CFX, **Scalable Wall Functions** are used for all turbulence models based on the ε-equation. For $k-\omega$ based models (including the SST model), an `Automatic` near-wall treatment method is applied.

4.2.1. Standard Wall Functions

In CFX-5.4.1 and earlier, standard wall functions were used. These have been replaced by scalable wall functions as the default but standard wall functions are still available for backward compatibility. You should not use standard wall functions, as they offer no advantage over the scalable wall functions.

4.2.2. Scalable Wall Functions

Scalable wall functions overcome one of the major drawbacks of the standard wall function approach in that they can be applied on arbitrarily fine meshes. If the boundary layer is not fully resolved, you will be relying on the logarithmic wall function approximation to model the boundary layer without affecting the validity of the scalable wall function approach. If you are not interested in the details of the boundary layer, then it may not be worth fully resolving it. However, if you have produced a very fine near-wall mesh to examine details of the boundary layer, then you should use the SST model with Automatic near-wall treatment to take advantage of the additional effect in the viscous sublayer.

4.2.3. Automatic Near-Wall Treatment for Omega-Based Models

Automatic near-wall treatment automatically switches from wall-functions to a low-*Re* near wall formulation as the mesh is refined.

One of the well known deficiencies of the $k-\varepsilon$ model is its inability to handle low turbulent Reynolds number computations. Complex damping functions can be added to the $k-\varepsilon$ model, as well as the requirement of highly refined near-wall grid resolution ($y^+ < 0.2$) in an attempt to model low turbulent Reynolds number flows. This approach often leads to numerical instability. Some of these difficulties may be avoided by using the $k-\omega$ model, making it more appropriate than the $k-\varepsilon$ model for flows requiring high near-wall resolution (for example, high wall heat transfer, transition). However, a strict low-Reynolds number implementation of the model would also require a near wall grid resolution of at least $y^+ < 2$. This condition cannot be guaranteed in most applications at all walls. For this reason, a new near wall treatment was developed by CFX for the $k-\omega$ based models that allows for a smooth shift from a low-Reynolds number form to a wall function formulation. This near wall boundary condition, named automatic near wall treatment in CFX, is used as the default in all models based on the ω-equation (standard $k-\omega$, Baseline $k-\omega$, SST, ω-Reynolds Stress).

To take advantage of the reduction in errors offered by the automatic switch to a low-*Re* near wall formulation, you should attempt to resolve the boundary layer using at least 10 nodes when using these models.

4.2.4. Treatment of Rough Walls

The near wall treatment that is discussed above (Scalable Wall Functions, Automatic Wall Treatment) is appropriate when walls can be considered as hydraulically smooth. For rough walls, the logarithmic profile exists, but moves closer to the wall. The near wall treatment becomes more complex, because it depends now on two variables: the dimensionless wall distance y^+ and the roughness height.

The roughness height that must be specified in the wall boundary section (see also Wall Roughness (p. 143)), is the equivalent sand grain roughness height. Note that the sand-grain roughness is not equal to the geometric roughness height of the surface under consideration. Wall friction depends not only on roughness height but also on the type of roughness (shape, distribution, and so on).

Therefore, one must determine the appropriate equivalent sand-grain roughness height. Guidance can be obtained from White [14], Schlichting [77] and Coleman et al. [193]. Details on the formulation of the rough wall treatment for turbulence models based on the ε and ω-equation can be found in Treatment of Rough Walls in the *CFX-Solver Theory Guide*.

If the Ansys-CFX two-equation transition model is used together with rough walls, then the roughness correlation must be turned on in the user interface. This correlation requires the geometric roughness height as input parameter. More details can be found in Ansys CFX Laminar-Turbulent Transition Models in the *CFX-Solver Theory Guide*.

4.2.5. Solver Yplus and Yplus

Yplus (y^+) is the dimensionless distance from the wall. It is used to check the location of the first node away from a wall. This was more important when standard wall functions were used, as you had to avoid y^+ values smaller than approximately 20. With the scalable wall functions and the automatic wall treatment, these values are only provided for information on the near wall resolution. Two variables related to y^+ are output to the results file, Yplus and Solver Yplus.

- The variable `Yplus` is based on the distance from the wall to the first node and the wall shear stress. This is the usual definition in the CFD community.
- The variable `Solver Yplus` is used internally by the CFX-Solver. It uses different definitions for the wall distance of the first wall point. This variable is available for backwards consistency, but is of little use to you.

There is usually no need for you to work with the y^+ used internally (`Solver Yplus`). The information required to judge the near wall mesh spacing should be based on the standard y^+ (`Yplus`) definition.

4.2.6. Guidelines for Mesh Generation

One of the most essential issues for the optimal performance of turbulence models is the proper resolution of the boundary layer. In this section, two criteria are suggested for judging the quality of a mesh:

- Minimum spacing between nodes in the boundary layer
- Minimum number of nodes in the boundary layer

These are simple guidelines for the generation of meshes that satisfy the minimal requirements for accurate boundary layer computations.

4.2.6.1. Minimum Node Spacing

The goal is to determine the required near wall mesh spacing, Δy, in terms of Reynolds number, running length, and a Δy^+ target value. This target value depends on the flow type and the turbulence model in use (that is, on the near-wall treatment in use). See Scalable Wall Functions and Automatic Near-Wall Treatment for Omega-Based Models in the *CFX-Solver Theory Guide* for more information.

4.2.6.1.1. Determination of the Near Wall Spacing

The estimates will be based on correlations for a flat plate with a Reynolds number of:

$$Re_L = \frac{\rho U_\infty L}{\mu} \tag{4.13}$$

with characteristic velocity U_∞ and length of the plate L.

The correlation for the wall shear stress coefficient, c_f, is given by White [14]:

$$c_f = 0.027\, Re_x^{-1/7} \tag{4.14}$$

where x is the distance along the plate from the leading edge.

The definition of Δy^+ for this estimate is:

$$\Delta y^+ = \frac{\Delta y\, u_\tau}{\nu} \tag{4.15}$$

with Δy being the mesh spacing between the wall and the first node away from the wall.

Using the definition

$$c_f = 2\frac{\rho u_\tau^2}{\rho U_\infty^2} = 2\left(\frac{u_\tau}{U_\infty}\right)^2 \tag{4.16}$$

u_τ can be eliminated in Equation 4.15 (p. 230) to yield:

$$\Delta y = \Delta y^+ \sqrt{\frac{2}{c_f}}\ \frac{\nu}{U_\infty} \tag{4.17}$$

c_f can be eliminated using Equation 4.14 (p. 230) to yield:

$$\Delta y = L\, \Delta y^+ \sqrt{74}\ Re_x^{1/14} \frac{1}{Re_L} \tag{4.18}$$

Further simplification can be made by assuming that:

$Re_x = C\, Re_L$

where C is some fraction.

Assuming that $C^{1/14} \approx 1$, then, except for very small Re_x, the result is:

$$\Delta y = L\, \Delta y^+ \sqrt{74}\ Re_L^{-13/14} \tag{4.19}$$

This equation allows us to set the target Δy^+ value at a given x location and obtain the mesh spacing, Δy for nodes in the boundary layer.

4.2.6.2. Minimum Number of Nodes

4.2.6.2.1. Goal

A good mesh should have a minimum number of mesh points inside the boundary layer in order for the turbulence model to work properly. As a general guideline, a boundary layer should be resolved with at least:

$$N_{\text{normal,min}} = \begin{cases} 10 \text{ for wall function} \\ 15 \text{ for low-Remodel} \end{cases} \tag{4.20}$$

where $N_{\text{normal, min}}$ is the minimum number of nodes that should be placed in the boundary layer in the direction normal to the wall.

4.2.6.2.2. Formulation

The boundary layer thickness δ can then be computed from the correlation:

$$Re_{\delta} = 0.14\, Re_x^{6/7} \tag{4.21}$$

to be:

$$\delta = 0.14\, L\, Re_x^{6/7} \frac{1}{Re_L} \tag{4.22}$$

The boundary layer for a blunt body does not start with zero thickness at the stagnation point for Re_x. It is, therefore, safe to assume that Re_{δ} is some fraction of Re_L, say 25%. With this assumption, you get:

$$\delta = 0.035\, L\, Re_L^{-1/7} \tag{4.23}$$

You would, therefore, select a point, say the fifteenth off the surface (for a low-Re model, or 10th for a wall function model) and check to make sure that:

$$n(15) - n(1) \leq \delta \tag{4.24}$$

Chapter 5: Domain Interface Modeling

This chapter describes:

5.1. Overview of Domain Interfaces

5.2. Interface Type

5.3. Interface Models

5.4. Mesh Connection Options

5.5. Defining Domain Interfaces as Thin Surfaces

5.6. Recommendations For Using Domain Interfaces

5.7. Automatic Creation and Treatment of Domain Interfaces

5.1. Overview of Domain Interfaces

Before reading this section, you should be familiar with the concepts of primitive regions, composite regions, and assemblies in CFX-Pre. For details, see Mesh Topology in CFX-Pre in the *CFX-Pre User's Guide*.

Domain Interfaces provide a way of connecting meshes or domains together. There are many uses of them. For example, they can be used in the following situations:

- If the flow field is repeated in multiple identical regions, then only one region must be solved, but the boundaries are specified as periodic (via a rotation or translation). A pressure change or mass flow rate may also be imposed for a translationally periodic interface.
- Domains can be connected together with a domain interface. This is often useful for connecting static domains together with non-matching grids. For example, hex meshes can be connected to tetrahedral or hybrid meshes. This is a useful strategy for reducing meshing effort. Pitch change can also be included.
- You can use domain interfaces to reduce mesh generation effort. The "divide and conquer" principle is useful, where the daunting task of generating a single mesh for the entire domain is divided into the much simpler task of generating a series of meshes that are later connected together.
- Domains that have a change in the reference frame (such as rotor-stator) can be connected together. Pitch change can also be included.
- Domains with different physical types can be connected together. For example, energy may flow through a fluid-solid interface. These interface types are often generated automatically. Other examples are fluid-porous and solid-porous interfaces.
- A domain may contain multiple mesh blocks. Non-matching interfaces can be connected together using a domain interface.

- A domain interface can be defined as a thin surface; a thin surface that has its **Mass and Momentum** set to `Conservative Interface Flux` can have physics defined across it.

Two methods of connecting meshes exist in Ansys CFX; these are one-to-one (direct) and general grid interface (GGI). Additional information on these connections and when they are used is available in Mesh Connection Options (p. 245).

5.2. Interface Type

The interface type is used to define the type of domain model connection across the interface. It may be set to one of the following options:

- `Fluid Fluid`
- `Fluid Porous`
- `Fluid Solid`
- `Porous Porous`
- `Solid Porous`
- `Solid Solid`

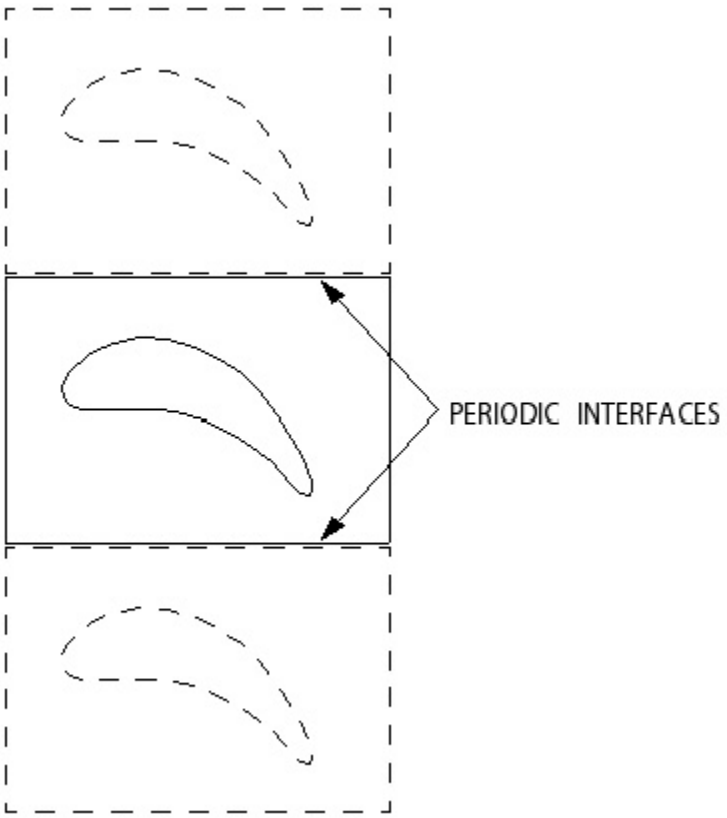
5.3. Interface Models

The interface model defines the way the solver models flow physics across the interface. The available interface model options are:

- `Translational Periodicity`
- `Rotational Periodicity`
- `General Connection`

The periodic options enable a simplified geometry to be used if the full problem features multiple identical regions, as in the following pictures.

Figure 5.1: Periodic Interfaces



The periodic condition ensures that the flow out of one side of the interface automatically appears on the other side. Periodicity is most common for fluid-fluid interfaces, but is also available for all other interface types.

5.3.1. Translational Periodicity

In this case, the two sides of the interface must be parallel to each other such that a single translation transformation can be used to map `Region List 1` to `Region List 2`. The trivial case is where the two sides already coincide so that a translation of `<0, 0, 0>` is used; this is equivalent to a general connection with no frame change or pitch change (discussed below).

A translational periodic interface could be used, for example, to represent an infinite straight row (a linear cascade) of turbine blades. For details, see Translational Periodicity (p. 243).

5.3.2. Rotational Periodicity

In this case, the two sides of the periodic interface can be mapped by a single rotational transformation about an axis. This is the most common case of periodicity and is used, for example, in the analysis of a single blade passage in a rotating machine.

If a domain interface involves rotational periodicity, the axis for the rotational transformation must also be specified.

5.3.3. General Connection

The `General Connection` interface model is a powerful way to connect regions together. A general connection can be used to:

- Apply a frame change at the interface between a rotor and stator
- Connect non-matching grids
- Apply fully transient sliding interfaces between domains

Pitch change can also be applied. For example, a domain interface can connect a stator domain with a rotor domain where the number of stator blades is not equal to the number of rotor blades, even if the mesh contains only one blade from each side.

Examples of the general connection type are shown below:

Figure 5.2: General Connection: No Frame Change

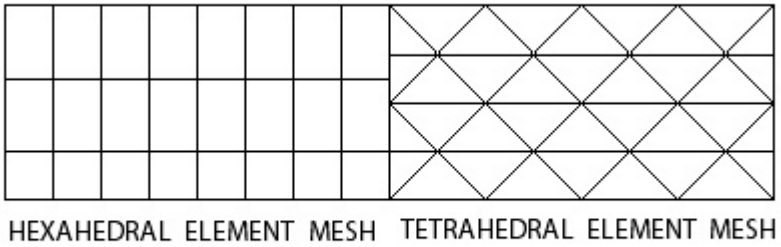
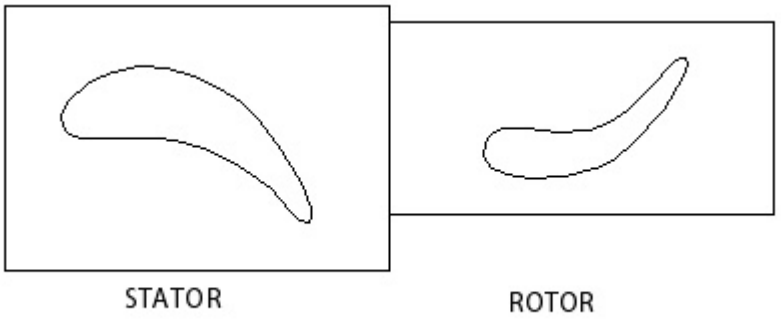


Figure 5.3: General Connection: Frame Change and Pitch Change



The `General Connection` option is necessary when the frame of reference or pitch changes across the interface. For example, the general connection option must be used in the following cases:

- One side is in a stationary frame of reference and the other side is in a rotating frame of reference
- Both sides are in a rotating frame of reference but each with a different rate of rotation

- Both sides are in the same frame of reference but have unequal pitches. In this situation, the flows through the interface must account for pitch change.

When a general connection is selected, the following details need to be specified:

- Frame Change/Mixing Model (p. 237)
- Pitch Change (p. 240)

5.3.3.1. Frame Change/Mixing Model

There are three types of frame change/mixing models available in Ansys CFX:

- `Frozen Rotor`
- `Stage (Mixing-Plane)`
- `Transient Rotor-Stator`

Each side of the interface must be a surface of revolution and both sides must sweep out the same surface of revolution.

5.3.3.1.1. None

Select `None` when there is no frame change or pitch change.

5.3.3.1.2. Frozen Rotor

The frame of reference and/or pitch is changed but the relative orientation of the components across the interface is fixed. The two frames of reference connect in such a way that they each have a fixed relative position throughout the calculation. If the frame changes the appropriate equation transformations are made. If the pitch changes, the fluxes are scaled by the pitch change.

This model produces a steady-state solution to the multiple frame of reference problem, with some account of the interaction between the two frames. The quasi-steady approximation involved becomes small when the through flow speed is large relative to the machine speed at the interface. `Frozen Rotor` analysis is most useful when the circumferential variation of the flow is large relative to the component pitch. This model requires the least amount of computational effort of the three frame change/mixing models.

The disadvantages of this model are that the transient effects at the frame change interface are not modeled. Modeling errors are incurred when the quasi-steady assumption does not apply. Also, the losses incurred in the real (transient) situation as the flow is mixed between stationary and rotating components is not modeled.

5.3.3.1.2.1. Rotational Offset

The **Rotational Offset** settings control the angle by which the side 1 or side 2 domains are rotated before the interface calculation is performed. The domain axis of rotation is used to perform the rotation. The rotational offset that is applied to the domains on side 1 or 2 depends on whether domains are rotating on side 1, 2, both, or none. The ways in which rotational offsets are stored in results file and are interpreted by CFD-Post are:

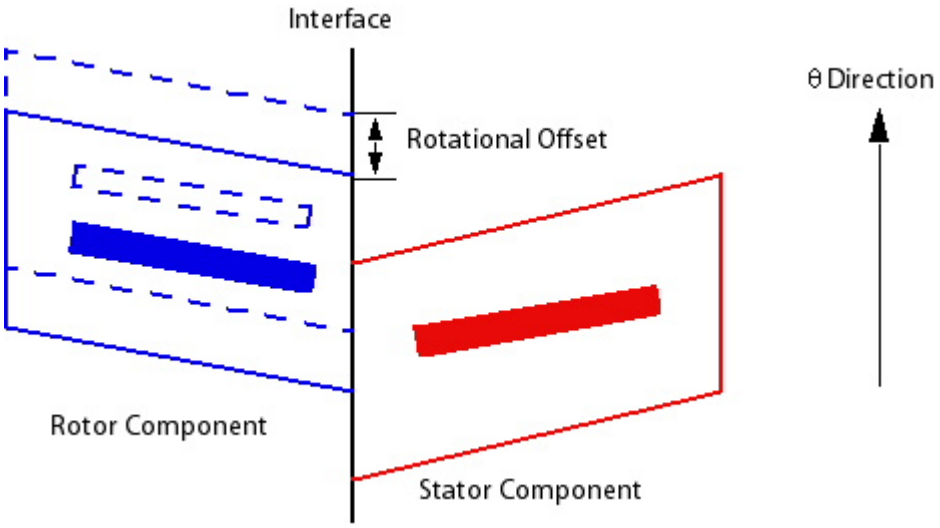
- If the side 1 domains are rotating and the side 2 domains are stationary then a rotational offset will be written for the side 1 domains, and interpreted by CFD-Post.
- If the sides 1 and 2 domains are rotating then a rotational offset will be written for the side 1 domains, and interpreted by CFD-Post.
- If the side 1 domains are stationary and the side 2 domains are rotating then a rotational offset will be written for the side 2 domains, and interpreted as a counter-rotation by CFD-Post.
- If the sides 1 and 2 domains are stationary then a rotational offset will be written for the side 1 domains, but not interpreted by CFD-Post; the rotational offset will not be applied.

You can use a rotational offset if you want to change the relative position of the components on each side of the interface without altering the position of the meshes. The positive direction for an offset is determined by the right-hand rule. Figure 5.4: Rotational Offset (p. 238) shows how a rotational offset will alter the relative position of a rotor and stator component.

Note:

As an alternative to the **Rotational Offset** parameter, you can also apply a domain rotation in CFX-Pre and create a new CFX-Solver input file.

Figure 5.4: Rotational Offset



By default, the domain positions are stored in their original (non-offset) location. To have the domains appear in their offset positions in CFD-Post, include the expert parameter `rotational offset for post=t` in the problem definition and, in CFD-Post, open **Edit** > **Options**, then set the **CFD-Post** > **General** option **Angular Shift for Rotating Locations** to **Always rotate**. Click **OK** to save the change.

5.3.3.1.3. Stage (Mixing Plane)

The Stage model (also known as the Mixing-Plane model) is an alternative to the Frozen Rotor model for modeling frame and/or pitch change. Instead of assuming a fixed relative position of the components, the Stage model performs a circumferential averaging of the fluxes through

bands on the interface. Steady-state solutions are then obtained in each reference frame. This model enables steady-state predictions to be obtained for multi-stage machines. The stage averaging at the frame change interface incurs a one-time mixing loss. This loss is equivalent to assuming that the physical mixing supplied by the relative motion between components is sufficiently large to cause any upstream velocity profile to mix out prior to entering the downstream machine component. `Stage` analysis is most appropriate when the circumferential variation of the flow is of the order of the component pitch.

`Stage` averaging between blade passages accounts for time average interaction effects, but neglects transient interaction effects. `Stage` analysis is not appropriate when the circumferential variation of the flow is significant relative to the component pitch (for example, a pump and volute combination at off-design conditions).

The Stage model usually requires more computational effort than the Frozen Rotor model to converge, but not as much as the transient rotor-stator model. You should obtain an approximate solution using a Frozen Rotor interface and then restart with a Stage interface to obtain the best results.

5.3.3.1.3.1. Pressure Profile Decay

The pressure profile at a Stage (Mixing-Plane) interface is determined by extrapolation from the pressure profile just inside the interface location (just upstream or just downstream of the Stage interface, for the upstream and downstream sides respectively). Sometimes, this procedure is unstable and a small amount of stiffness must be added to the pressure profile rather than just letting it float. By default, the pressure profile is decayed by 5% towards a constant value. The amount of decay can be controlled by this setting. Using a 5% pressure profile decay is recommended.

5.3.3.1.3.2. Downstream Velocity Constraint

At a Stage (Mixing-Plane) interface, the average static pressure within each band on both the upstream and downstream side of the interface is set to the average band pressure. Here are possible treatments for velocity on the downstream side:

- It may be calculated as the average band velocity. (`Stage Average Velocity`)
- It may be calculated from the average band total pressure and direction in the relative frame (`Constant Total Pressure`). Note that, in this case, the frame type will be rotating.

The constant total pressure options permit the downstream velocity profile to naturally adjust to downstream influences. For tightly-coupled components, the `Constant Total Pressure` option provides a better approximation than the `Stage Average Velocity` option. To obtain results comparable to CFX-TASCflow, use `Constant Total Pressure`, `Frame Type=Rotating`.

5.3.3.1.3.3. Implicit Stage Averaging

This option helps to improve the convergence of the Stage (Mixing-Plane) model and minimizes reflection on the upstream side of the interface due to locally supersonic flow that is subsonic in the direction normal to the interface.

5.3.3.1.4. Transient Rotor-Stator

This model should be used any time it is important to account for transient interaction effects at a sliding (frame change) interface. It predicts the true transient interaction of the flow between a stator and rotor passage. In this approach the transient relative motion between the components on each side of the GGI connection is simulated. It ultimately accounts for all interaction effects between components that are in relative motion to each other. The interface position is updated each timestep, as the relative position of the grids on each side of the interface changes. It is possible to use a transient sliding interface anywhere a `Stage` or `Frozen Rotor` sliding interface could be used.

The principle disadvantage of this method is that the computer resources required may be large, in terms of simulation time, disk space and quantitative postprocessing of the data. The resource requirement problem is exacerbated if unequal pitch between components occurs. In these situations, spatial periodicity cannot formally be used to limit the analysis to a single blade passage per component. Often the problem of unequal pitch is addressed by modifying the geometry to the nearest integer pitch ratio, which may affect the validity of the analysis. In practice, components of unequal pitch can be treated by solving *N* passages on one side and *M* passages on the other side, with *N* and *M* determined such that the net pitch change across the interface is close to unity. As with a Frozen Rotor interface, pitch change is automatically accounted for by scaling of flows by the pitch ratio.

It is possible to start a `Transient Rotor-Stator` computation from a simple initial guess, or from an existing prediction. If you are interested in the start-up transient of the machine, then start from the appropriate physical initial conditions. If you are interested in simulating a periodic-in-time quasi-steady-state, then it may be helpful to first obtain a steady-state solution using `Frozen Rotor` interfaces between components. This solution will contain most of the overall flow features, and should converge to the desired transient simulation in the fewest transient cycles.

Translational relative motion is not supported at a transient sliding interface (only rotational motion is supported).

5.3.3.2. Pitch Change

To connect dissimilar meshes, an intersection algorithm is used to find the overlapping parts of each mesh face at the interface; for details, see GGI and MFR Theory in the *CFX-Solver Theory Guide*. The CFX-Solver must decide the most appropriate way to perform this intersection. The choices are:

- Intersect in physical space. The connection will be made as is with no correction for pitch change or rotational offset. This method is used when the pitch change model is set to `None`.
- Intersection in the radial direction. This method is used when the pitch change model is not `None` and the radial variation at the interface is larger than the axial (z-) variation. For example, this method is used for the surface of constant z in Figure 5.5: Radial vs. Z Direction (p. 241).
- Intersect in the axial direction. This method is used when the pitch change model is not `None` and the axial variation at the interface is larger than the radial variation. For example, this method is used for the surface of constant radius in Figure 5.5: Radial vs. Z Direction (p. 241).

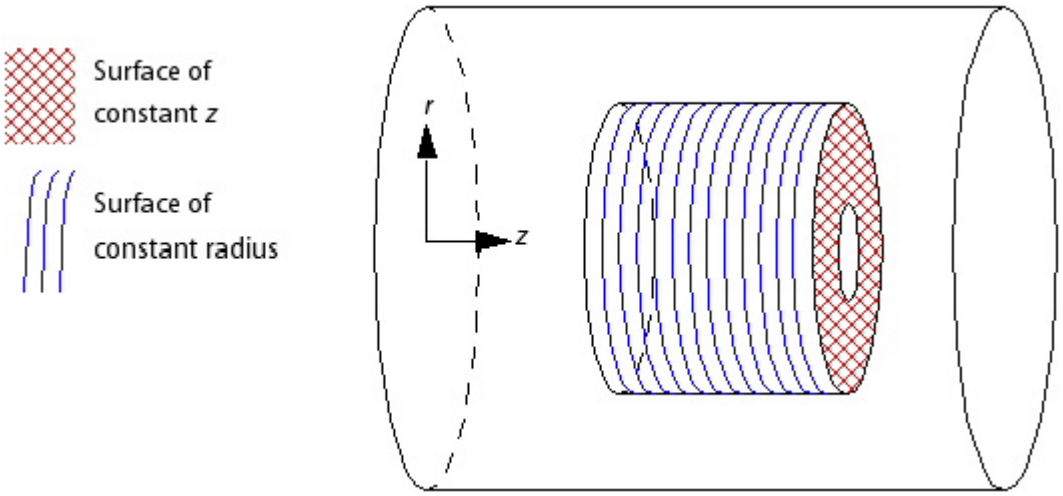
If the pitch change model is not None, then the intersection algorithm will fail if the interface has any of the following:

- surfaces of zero radius
- surfaces of constant radius with surfaces of constant z

For example, an interface cannot contain both highlighted regions in Figure 5.5: Radial vs. Z Direction (p. 241). Instead, two separate interfaces should be used.

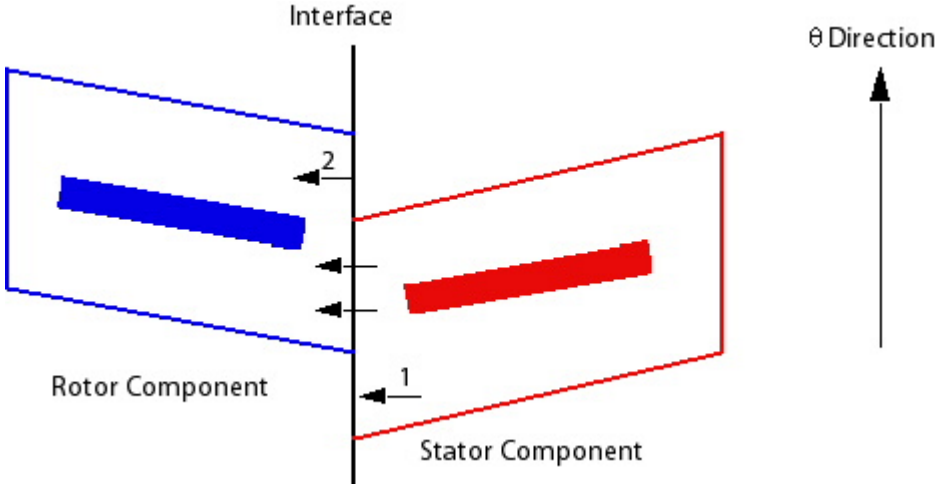
In addition, for best accuracy, the interface should not be highly curved. For example, if a small change in radius produces a large change in z and a small change in z produces a large change in radius in the same interface, then separate interfaces should be used instead.

Figure 5.5: Radial vs. Z Direction



When there is pitch change, a transformation is also performed between components to account for the fact that they may not be rotationally aligned. This does not affect the relative position of the components. In Figure 5.6: Misaligned Components (p. 241) the components are not aligned; flow passes through the overlapping area as expected, but the flow that reaches the interface at 1 is transformed such that it emerges at 2 having crossed the interface.

Figure 5.6: Misaligned Components



5.3.3.2.1. None

Using None as the pitch change option will cause the connection to be made "as is." If you have non-aligned components, as shown in Figure 5.6: Misaligned Components (p. 241), flow will only pass through the overlapping region. Therefore, this option should only be used when the overall extent and shape of each side of the interface perfectly match and the surfaces are aligned with each other. It may therefore be suitable when full 360 degree or exactly equal pitch components are being analyzed. A pitch change option of None cannot be used for a stage interface.

5.3.3.2.2. Automatic

The Automatic pitch change option is the standard option for accounting for pitch change between components. The pitch ratio is taken to be the ratio of the areas of the two components. The two sides must each have the same radial or axial extent and can therefore only differ in extent in the direction of rotation of the frame of reference. This means that the surfaces on each side of a frame change interface must sweep on the same surface of revolution.

For the Frozen Rotor model, the flow variable profiles in the pitch-wise direction are stretched or compressed to the extent that there is pitch change across the interface. All flows (mass, momentum, energy, and so on) are scaled accordingly, based on the pitch change.

For the Stage (Mixing-Plane) model, the pitch change adjustment is made by applying average values calculated from the upstream side of the interface to the downstream side. The averaging is performed in a conservative manner. Meridional variation of the flow is maintained across the interface. Circumferential variation of all flow variables except pressure are removed across the interface. The average pressure is maintained at each meridional location across the interface, with independent local variation in pressure permitted.

The computational accuracy degrades rapidly with increasing pitch change. It is recommended that sufficient blade components be analyzed on each side of the interface to minimize pitch change. Any pitch change will result in a non-physical transient interaction: the entire 360 degree components must be solved for truly "accurate" transient interactions if the pitch ratio is non-integer. Small pitch changes will introduce relatively smaller errors, large pitch changes will introduce larger errors.

There are a number of scenarios where the algorithm used by the Automatic pitch change option is not valid, such as:

- When the vertices on the intersection of the hub and interface are not all at the same radial and axial position
- When one or both sides of the interface has a 360° or greater pitch angle
- When one or both sides of the interface has a zero radius
- When one or both sides of the interface has mesh faces normal and parallel to the rotation axis
- When one or both sides of the interface has mesh faces at the hub (that is, low radial or axial position) that are thin in the radial or axial direction

When using the Automatic pitch change option, you should examine the CFX-Solver Output file under the **Mesh Information** heading to confirm that the calculated pitch angles are sensible.

Using the `Automatic` pitch change option for determining the pitch change may result in failure of the solver or a solution that is completely wrong. An example of the flow solution being wrong is flow exiting a non-overlapping region of the interface or flow not exiting an overlapping area of the interface. In general if the flow solution through the interface does not make sense, or if the calculated pitch angles are not sensible in the CFX-Solver Output file, then you should use the `Value` or `Specified Pitch Angles` option.

5.3.3.2.3. Value

This option enables you to explicitly provide the pitch ratio. This is the ratio of the pitch angle from the side with the larger pitch angle to the side with the smaller pitch angle across the domain interface. You can use this option when the precise pitch ratio is critical to the analysis; for example, in a closed system containing multiple pitched components.

5.3.3.2.4. Specified Pitch Angles

This option enables you to specify the pitch angle on side 1 and side 2 of the domain interface. You can use this option when the precise pitch angle is needed or when the area ratio of the two sides is not equal to the pitch ratio.

5.3.4. Mass and Momentum Models

CFX offers two kinds of mass/momentum modeling options at domain interfaces: a user-specified pressure change and a user-specified mass flow rate. These have different applications depending on the type of domain interface.

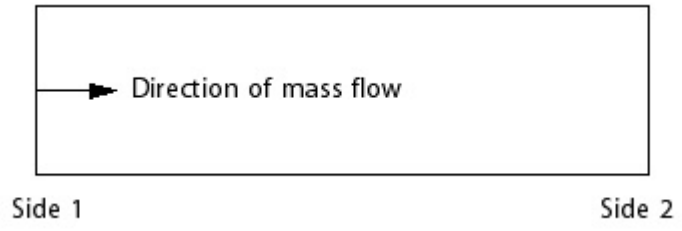
5.3.4.1. Translational Periodicity

It is often useful to model periodic flow but enable a pressure drop between the two sides of the interface. Examples of this include:

- flow through tube banks in heat exchangers
- periodic duct flow

If the pressure drop is specified, the solver will calculate the flow field including the mass flow rate. Alternatively, if the mass flow rate through the interface is known, the solver will calculate the flow field including the pressure drop. If a pressure change condition is specified, the specified pressure change is assumed to be from side 1 to side 2 of the interface (see the figure below). For pressure drop situations like flow through a duct or tube bank, this pressure change should be negative. If a mass flow rate condition is specified, the specified mass flow rate is assumed to be into the domain through side 1 and out of the domain through side 2.

Figure 5.7: Applying a Mass Flow at a Periodic Domain Interface



5.3.4.2. General Connection

The following sections describe applying a specified pressure change or a mass flow rate to a general domain interface.

5.3.4.2.1. Specified Pressure Change

It is sometimes useful to apply a pressure change condition to a general domain interface. For example, a screen model might prescribe a pressure drop at the domain interface which is proportional to the local dynamic head:

$$\Delta p = -k\rho\left(u^2+v^2+w^2\right) \tag{5.1}$$

where k is the screen loss coefficient and Δp is the pressure change measured from side 1 to side 2. This screen model can be implemented using the following CEL expression:

Pressure Change = `-k * density * (u^2 + v^2 + w^2)`

Note that in the case that the density in the above expression is itself dependent on the pressure, such as for an ideal gas, the pressure from side 2 is used to evaluate the density.

Another application of the pressure change condition is modeling a fan for which the pressure rise is a function of the mass flow rate:

$$\Delta p = f\left(\dot{m}\right) \tag{5.2}$$

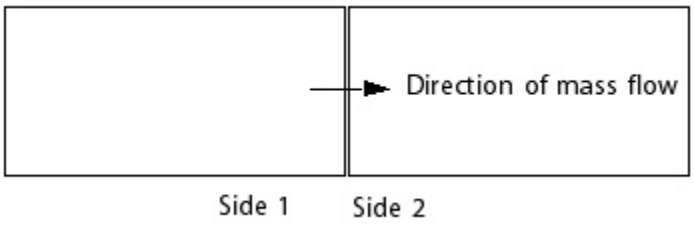
This fan model can be implemented by creating an interpolation function or CEL expression for Δp; for example, if one side of the domain interface has the name `Fan Side 1`, one could create a CEL expression or interpolation function for Δp as follows:

Pressure Change=f(`massFlow()@Fan Side 1`)

5.3.4.2.2. Specified Mass Flow Rate

The mass flow rate can also be specified through a general connection. The specified mass flow rate is assumed to be from side 1 to side 2 of the interface. See the figure below.

Figure 5.8: Applying a Mass Flow at a General Connection



The solver implements the specified mass flow condition by adjusting the pressure change until the specified mass flow rate is satisfied. Prior to convergence, the actual mass flow rate may not satisfy the specified mass flow rate, although it typically becomes close within a few dozen iterations. To enhance robustness of this condition, the pressure change is under-relaxed with a default under-relaxation factor of 0.25. This value can be increased if convergence to the specified mass

flow rate is slow but monotonic and decreased if convergence to the specified mass flow rate is oscillatory or unstable.

5.3.4.3. Further Comments

When using a pressure change, the solver checks an additional criterion when deciding whether the problem has converged. For the pressure change condition, the solver checks whether

$$\frac{\dot{m}-\dot{m}_{old}}{\max\left(|\dot{m}|,|\dot{m}_{old}|\right)}<x \tag{5.3}$$

where x is the domain interface convergence criterion. By default, x is 0.01, but can be modified by changing **Domain Interface Target** on the **Solver Control** tab in CFX-Pre.

When using a mass flow rate condition, the solver checks the following additional criterion when deciding whether the problem has converged:

$$\frac{\Delta p-\Delta p_{old}}{\max\left(|\Delta p|,|\Delta p_{old}|\right)}<x \tag{5.4}$$

The mass flow rate through each side of the domain interface can be monitored by creating a new plot in the CFX-Solver Manager. For the plot line, select `FLOW`, then `DOMAIN INTERFACE`, then the domain interface side you want to monitor, and then choose `P-Mass`.

5.4. Mesh Connection Options

Domain interfaces connect meshes using two region lists. There are three mesh connection options:

- Automatic Connections (p. 245)
- Direct (One-to-One) Connections (p. 246)
- GGI (General Grid Interface) Connections (p. 247)

In certain situations, General Grid Interface (GGI) connections are used even when there is a direct (one-to-one) correspondence in nodes on either side of the interface. The rules for when this occurs are given below. If the physics permits a direct connection, the latter is usually preferable.

All domain interfaces automatically create boundary conditions of type "Interface" for the region lists used. For details, see Interface Boundary Conditions in the *CFX-Pre User's Guide*.

5.4.1. Automatic Connections

CFX-Pre automatically chooses whether the connection is direct (one-to-one) or not, based on whether the nodes are coincident and the selection of regions on `side 1` and `side 2`. The tolerance used to determine if nodes match across the interface, **Mesh Match Tolerance**, can be set in CFX-Pre under **Edit** > **Options** > **CFX-Pre** > **Mesh**. The actual connection type chosen is reported by CFX-Pre when the CFX-Solver input file is written.

An automatic connection cannot be made when conditional connection control is used for the interface. In such a case, the only available mesh connection option is `GGI`. For details on conditional connection control, see Conditional Connection Control in the *CFX-Pre User's Guide* and Conditional Connections (p. 248).

5.4.2. Direct (One-to-One) Connections

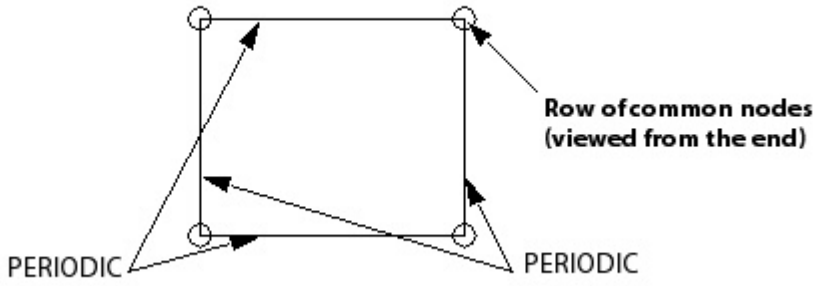
The direct (one-to-one) connection is available if, after any applicable rotational or translational transformation (such as for a periodic interface), all of the nodes on one side of the interface correspond in location with all of the nodes on the other side of the interface, to within a spatial tolerance governed by the **Mesh Match Tolerance** parameter. The latter can be set in CFX-Pre by changing **Edit** > **Options** > **CFX-Pre** > **Mesh** > **Mesh Match Tolerance** setting. For details, see Mesh in the *CFX-Pre User's Guide*. You can verify the type of interface that CFX-Pre will apply by right-clicking `Simulation` in the **Outline** tree view, and selecting **Report Interface Summary** from the shortcut menu.

In addition to requiring a precise node correspondence from one side to the other, a direct (one-to-one) connection requires that:

- The number of regions in the list for one side is equal to the number of regions in the list for the other side
- For each region in the region list for one side, there is exactly one region in the list for the other side that corresponds entirely in terms of the number of nodes and their positions
- Both sides of the interface are in the same domain for `Fluid Fluid` and `Solid Solid` interfaces
- The interface type is not `Fluid Porous` or `Porous Porous`
- A pressure change or mass flow condition is not set at the interface.

In addition, for periodic interfaces, CFX-Solver will change direct connections to GGI connections as necessary to ensure that every node that participates in multiple direct connections has only one unique required transformation (translation or rotation). For example, in Figure 5.9: Nodes Shared by Interfaces with Different Transformations (p. 246), the corner nodes are involved in two interfaces that have different transformations. In this case, at least one of the interfaces will be changed to use a GGI connection.

Figure 5.9: Nodes Shared by Interfaces with Different Transformations



A direct (one-to-one) connection cannot be made when conditional connection control is used for the interface. For details on conditional connection control, see Conditional Connection Control in the *CFX-Pre User's Guide* and Conditional Connections (p. 248).

5.4.3. GGI (General Grid Interface) Connections

General Grid Interface (GGI) connections refer to the class of grid connections where the grid on either side of the two connected surfaces does not match. In general, GGI connections permit non-matching of node location, element type, surface extent, surface shape and even non-matching of the flow physics across the connection.

All model options within CFX (for example, turbulence models, multiphase models, particle tracking, mixture models, reaction, and so on) support the use of GGI connections. Any number of GGI connection conditions are possible within a computational domain.

The GGI connection will be made in a conservative and implicit fashion, even if the nodes on the two sides of the connection are not aligned. Likewise, the element types on each side do not have to match. In addition, if the size of the connection region on one side is different than that on the other side, the connection will be made automatically between the mutually-overlapping surfaces. Finally, it is possible to perform a connection where there is a slight gap or interference between the two sides of the GGI connection. Here, "slight" means an interference or gap on the order of ½ the average depth of the elements touching the connection surface. This permits connections where the surfaces do not fit together perfectly (for example, curved connection with different grid densities on each side), or where the components are not perfectly aligned with each other. Even in these situations, conservation is guaranteed and strictly enforced; for details, see GGI and MFR Theory in the *CFX-Solver Theory Guide*.

The regions that are on each side of a GGI connection are allowed to be of different size and are allowed to have non-overlap regions (see Non-overlap Boundary Conditions (p. 247)). The extent of the non-overlap region is reported in the CFX-Solver Output file (see CFX-Solver Output File (GGI Runs) in the *CFX-Solver Manager User's Guide*). If the extent of the non-overlap area is larger than expected, confirm that the correct regions have been selected for each side of the interface. There are Intersection Control settings that can be used to increase the accuracy of the overlap calculation if necessary (see Intersection Control: Option in the *CFX-Pre User's Guide*).

5.4.3.1. Non-overlap Boundary Conditions

The portions of the connecting surfaces in a GGI connection that do not overlap each other behave by default in the same way as the default wall boundary condition that is made by CFX-Pre in a fluid domain. This default condition is a no-slip adiabatic wall with a zero-flux condition applied to all other transport equations. Particles will bounce off these walls with restitution coefficients of 1, and radiation will see an adiabatic wall.

The default behavior can be changed after the domain interface has been created by visiting the boundary condition object for the side of the interface where you want the boundary condition changed. For example, you may choose to set a wall roughness, or set a wall velocity, or change to a free slip wall. You can also change the zero-flux condition for any transport equation. For example, you can enable heat transfer or the transport of Additional Variables.

CFX-Solver will output a variable, `Nonoverlap Fraction`, for each side of the interface to indicate which nodes are in the overlap portion (`Nonoverlap Fraction=0`) and which are in the non-overlap portion (`Nonoverlap Fraction=1`) of the interface. This variable is available to CFD-Post. For example, you could multiply a quantity in an expression by `Nonoverlap Fraction` or (`1-Nonoverlap Fraction`) to ignore contributions from nodes on a GGI interface that are in the overlap portion, or non-overlap portion, respectively.

5.4.3.2. Conditional Connections

Conditional Connection Control enables you to use an expression to control whether an interface is open (connected) or closed (not connected; a wall boundary is applied). This feature can be used to avoid having to use a moving mesh to suddenly snap interface surfaces apart at a given time (in order to force a non-overlap condition to exist), or to use multiple configurations to load different meshes during a simulation.

To set up a conditional connection control, configure the **Conditional Connection Control** settings on the **Additional Interface Models** tab for the domain interface. These settings are described in Conditional Connection Control in the *CFX-Pre User's Guide*.

Note:

When conditional connection control is used, the **Mesh Connection** option (on the **Mesh Connection** tab) is forced to be `GGI`.

The CFX-Solver will write diagnostics information to the CFX-Solver Output file whenever the state of an interface with a specified Conditional Connection Control is changing.

5.4.4. Mesh Connection Recommendations

Domain interfaces that use GGI connections are a very powerful and flexible mesh connection method, but they do require some additional computational effort and memory, and may introduce numerical inaccuracy compared to an equivalent computation that does not use GGI connections. For these reasons, you should use GGI connections wisely and sparingly.

For periodic interfaces, GGI connections always introduce some error into your simulation compared with direct connections. This is because direct connections solve the equations as if the mesh were actually replicated, so there is no additional discretization error due to the presence of the interface.

For simulations involving conjugate heat transfer, on the other hand, direct connections are less desirable than GGI connections because in direct connections the discretization of heat flow through fluid-solid, porous-solid, and solid-solid interfaces is *nonsymmetric* (that is, direct connections use more information from the more conductive side of the interface, while GGI connections sample both regions equally). The direct connection's nonsymmetric treatment may lead to accuracy and/or convergence difficulties in the following situations:

- The domain interface is fluid-solid or porous-solid.
- The domain interface is solid-solid and there is a large conductivity ratio across the interface.

Note:

- In laminar domains, fluid-solid interfaces should be GGI to avoid the possibility of a Class 3 error.
- To avoid spurious overshoots and undershoots of the solution in the vicinity of the interface, the **Automatic** setting employs GGI for fluid-solid, porous-solid, and solid-solid interfaces.

In these situations, you should consider using a GGI connection rather than a direct connection, even if a direct connection is possible.

If you believe that an interface should be a one-to-one connection, but a GGI connection is made instead, then you should first check that you have set the correct rotation axis for a rotational periodic interface. Next, you should load the CFX-Solver input file into CFD-Post to check that the mesh appears to match. You can plot the mesh on each side of the interface using a different color to check that the two sides match. In the case of periodic interfaces, you can use an instancing transform to manually perform the transformation and then check that the meshes match. Before increasing the **Mesh Match Tolerance**, you should try to reproduce the mesh such that it matches within the default tolerance. If the meshes appear to match but a GGI connection is still created, you can increase the **Mesh Match Tolerance**. Increasing the tolerance above 0.02 (2%) is not recommended as this will result in a loss of accuracy.

5.5. Defining Domain Interfaces as Thin Surfaces

In Release 12.0, thin surfaces have been implemented to enable the modeling of physics (such as heat transfer, mass transfer, and momentum losses) across thin-surfaces. In previous releases, the **Create Thin Partner** option created two independent walls, which did not enable physics to be implemented. Starting with Release 12.0, you can model a thin surface either as:

- A boundary condition (for details see Default Boundary Condition in the *CFX-Pre User's Guide*)
- A domain interface with the physics modeled across the interface.

The types of domain interfaces you can define as thin surfaces depends on the domains involved:

Domains	Allowed Domain Interface
Fluid-Fluid	Solid
Fluid-Porous	
Porous-Porous	
Solid-Fluid	Solid or Constant-Property Substance[a]
Solid-Porous	
Solid-Solid	

[a] A Constant-Property Substance could be a fluid that has no motion and that is not pressure-dependent.

5.5.1. Modeling Thin Surfaces: Overview

Conceptually, to create a thin surface:

1. Create your domains.
2. Define a domain interface. Physics definitions are set on the **Additional Interface Models** tab; the provided models are determined automatically from the geometry.

   Choose **Conservative Interface Flux** to define a thin surface that has physics operating between the two sides; if you choose that option, set the **Interface Model** as well.

3. When you click **OK** to create the domain interface, a pair of boundaries are created (`Domain Interface 1 Side 1` and `Domain Interface 1 Side 2`) under **Simulation** > **Flow Analysis** > **<domain_name>**.

5.6. Recommendations For Using Domain Interfaces

This section describes:

- Using Domain Interfaces (p. 250)
- Using Multiple Domain Interfaces (p. 250)
- Using Domain Interfaces in Turbomachinery Applications (p. 251)

5.6.1. Using Domain Interfaces

The following points should be considered when using domain interfaces:

- Domain interfaces are always required when multiple assemblies are used within a domain or across multiple domains. If you do not create the required domain interfaces, the regions will remain part of the default wall boundary condition.
- Domain interfaces are also always required when multiple domains are created within a single assembly. If you do not create the required domain interfaces, default domain interfaces will be created which may or may not be appropriate.
- Domain interfaces are not required when multiple 3D regions within the same assembly are selected for a single domain.
- Any given domain must contain at least one node that is not used in a domain interface. Domains defined only by interface nodes will not be accepted by the CFX-Solver.
- You cannot use **Mesh Adaption** when a simulation contains domain interfaces.

Information on the implementation of GGI connections is available in GGI and MFR Theory in the *CFX-Solver Theory Guide*.

5.6.2. Using Multiple Domain Interfaces

A given simulation can make use of multiple domain interfaces. Each `Fluid Fluid` interface can use a different frame change model. For example, two domains can be connected by a `Fluid Fluid` interface without a frame change; one of these domains might also contain a rotationally periodic interface to define a blade passage. Finally, there might be another `Fluid Fluid` interface between the first two domains and a third domain that uses a frame change model. A four stage axial compressor, consisting of four stators and four rotors, might contain seven `Fluid Fluid` interfaces with a `Frozen Rotor` frame change model (one at each frame change interface between components).

It is possible to have a combination of frame change models within a single problem definition. For example, in a stator/rotor/stator configuration, the first `Fluid Fluid` interface could be a `Stage` or `Frozen Rotor` interface, and the second interface could be a `Transient Rotor-Stator`

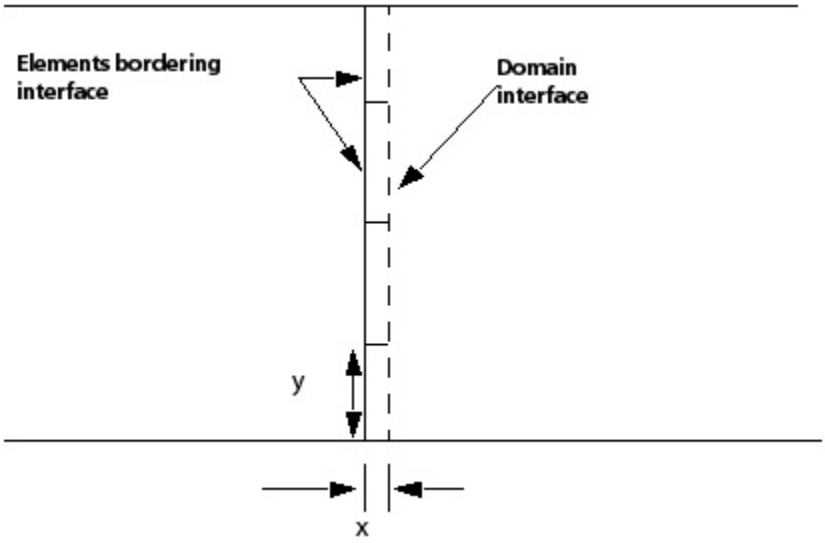
interface. This might make sense if you are only interested in accounting for the transient conditions between the rotor and the downstream stator.

5.6.3. Using Domain Interfaces in Turbomachinery Applications

The setup of domain interfaces is an important consideration when defining a turbomachinery problem. The following section outlines some approved practices for use in turbomachinery applications.

- Domain Interfaces should typically be placed midway between the rotor and stator for turbomachinery cases.
- Where circular domain interfaces exist, they must be axisymmetric in shape.
- The aspect ratio of elements on the domain interface should not be greater than 10:1. Figure 5.10: Element Aspect Ratio (p. 251) gives a sketch of the requirement for an element aspect ratio of less than 10 on the domain interface. The element aspect ratio (x/y) should lie in the range 0.1 to 10 to avoid numerical errors.

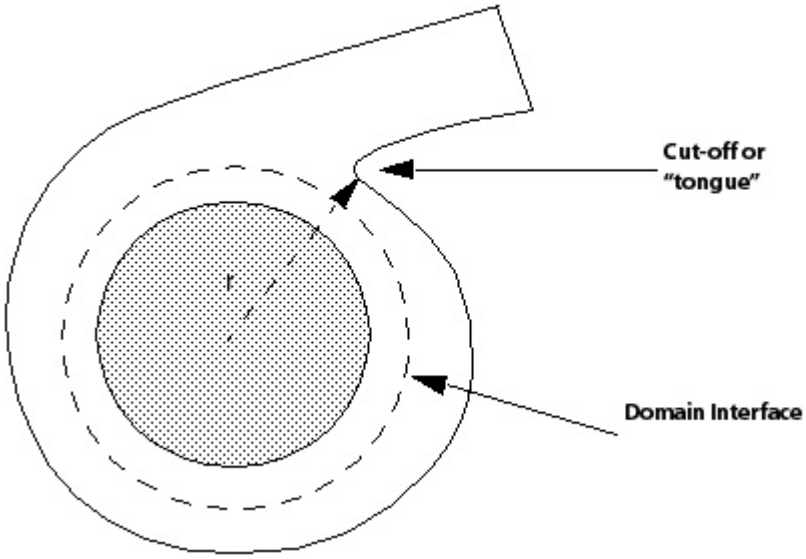
Figure 5.10: Element Aspect Ratio



5.6.3.1. Case 1: Impeller/Volute

A basic impeller/volute sketch is shown in Figure 5.11: Basic Impeller/Volute (p. 252). The edge of the inner circle shows the maximum extent of the impeller blades. A good practice here is to create the domain interface halfway across the most narrow gap between the blade and volute wall. This usually occurs around the cut-off or tongue illustrated in the diagram.

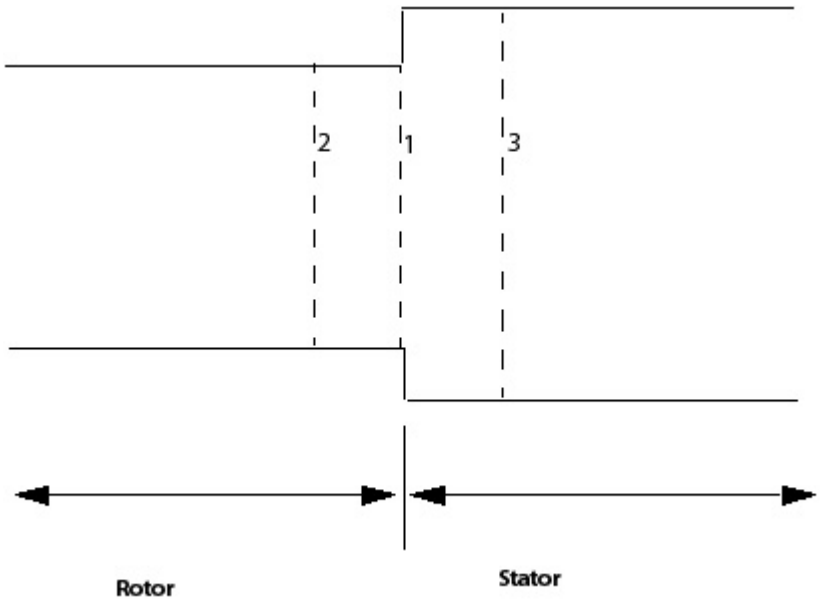
Figure 5.11: Basic Impeller/Volute



5.6.3.2. Case 2: Step change between rotor and stator

For the case shown in Figure 5.12: Step change (p. 252), there is a step change between the rotor and stator. A common choice for placement of the interface would be choice 1. However, take care with this set-up because the non-overlap regions above and below the interface should be specified as walls. A better alternative may be to use a domain interface upstream or downstream of the step change, at position 2 or position 3.

Figure 5.12: Step change

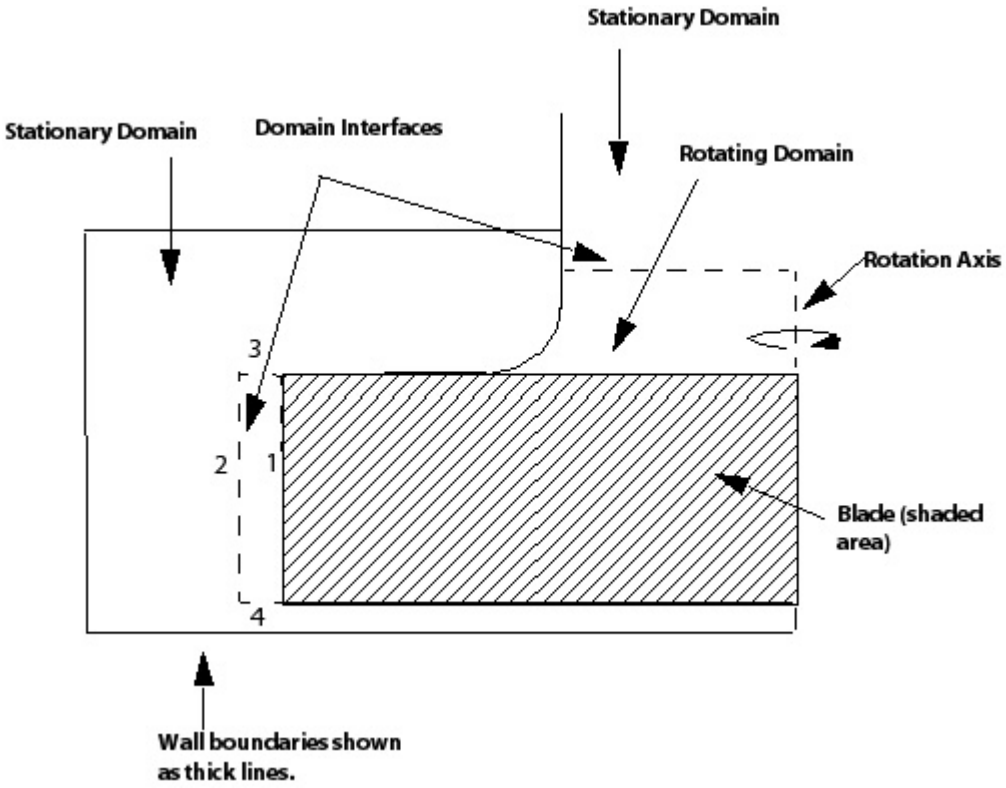


5.6.3.3. Case 3: Blade Passage at or close to the edge of a domain

Figure 5.13: Blade extending to the edge of the rotating domain. (p. 253) shows a blade that extends to the edge of the rotating domain. Although it is convenient to place a domain interface at the

blade edge (1), this can result in convergence difficulties. A better arrangement is to extend the rotating domain away from the blade edge. Domain Interfaces can then be created at (2), (3), and (4).

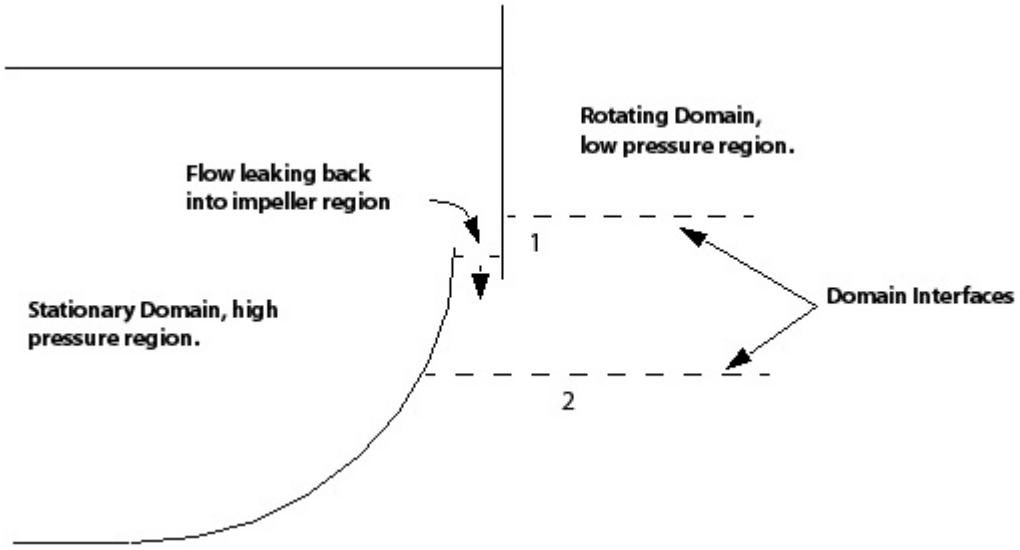
Figure 5.13: Blade extending to the edge of the rotating domain.



5.6.3.4. Case 4: Blade Passage at or close to the edge of a domain

Figure 5.14: Blade passage (p. 254) shows a close-up view of part of Figure 5.13: Blade extending to the edge of the rotating domain. (p. 253), which models flow leaking from a volute back into the impeller region. To model the feature, you can use two domain interfaces (at position 1), or a single domain interface downstream of the leak (position 2).

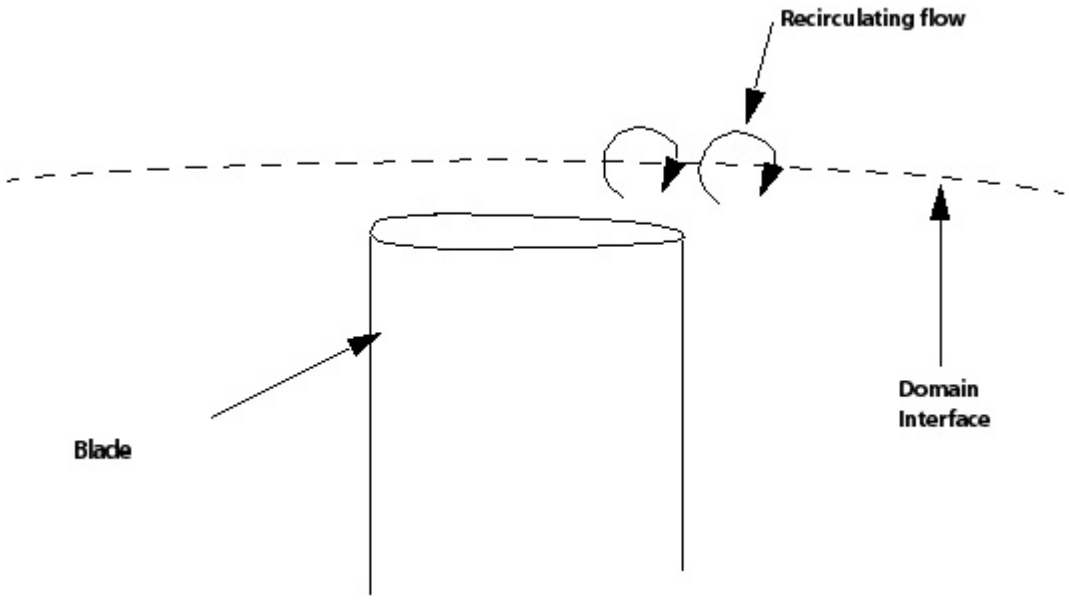
Figure 5.14: Blade passage



5.6.3.5. Case 5: Blade with thick trailing edge

As Figure 5.15: Thick trailing edge (p. 254) shows, when a blade with a thick trailing edge passes close to the domain interface, recirculating flow that crosses the interface can occur. Running a case like this with a Stage interface can produce unphysical results. It is recommended that you either move the domain interface away from possible recirculation zones, or run with the Frozen Rotor interface.

Figure 5.15: Thick trailing edge



5.7. Automatic Creation and Treatment of Domain Interfaces

You should generally create all domain interfaces that are needed in your simulation explicitly in CFX-Pre. However, CFX-Pre will create default domain interfaces in certain situations.

Domain interfaces will be automatically created, at the Write Solver File stage, when there are multiple domains within a single assembly and you have not explicitly created all the required domain interfaces. These could be `Fluid Fluid`, `Fluid Solid` or `Solid Solid` domain interfaces depending on the type of domain on either side of the interface.

When a default `Fluid Fluid` interface is created and there is no change in reference frame across the interface, the **Frame Change** option is set to `None`. When a frame change occurs, the Frozen Rotor model is used. If these models are not appropriate, you should explicitly create all domain interfaces to prevent the automatic creation of interfaces.

When a one-to-one node mapping does not exist, you must manually create the domain interfaces. If multiple domains are created using separate assemblies, domain interfaces are always required.

Fluid-solid 2D primitives are initially contained in the default domain boundary conditions. Any that remain there when a CFX-Solver input file is written are assigned to default `Fluid Solid` domain interfaces.

If you create a MFR case using 3D primitives from a single assembly, then a one-to-one node pairing will exist because an assembly contains a continuous mesh. However, a GGI connection is required for MFR cases. A default domain interface using the `Frozen Rotor` frame change model will be automatically created if no other domain interfaces are created.

Chapter 6: Turbomachinery Blade Row Modeling

This chapter discusses turbomachinery related models in CFX. Some of the models described here are applicable outside of turbomachinery applications.

The following topics are discussed.

6.1. Transient Blade Row Modeling
6.2. Blade Film Cooling

6.1. Transient Blade Row Modeling

It can be computationally expensive to predict the unsteady phenomena that occur as a result of the interaction between adjacent blade rows, blade flutter, or boundary disturbances. It can be even more expensive if you need to include more than a few passages per blade row to capture the phenomena more accurately.

In typical turbomachinery applications, it is very common for one or both blade rows to have a prime number of blades per wheel. Formerly in such cases, it was necessary to model the whole 360° wheel in order to attain the required level of accuracy. The Transient Blade Row models available in Ansys CFX make it possible to reduce the size of the computational problem (memory and computational time) by solving the blade row solution for one or two passages per row, while still obtaining reasonably accurate solutions, therefore providing a solution to the unequal pitch problem between the blade passages of neighboring rows.

The Transient Blade Row models available in Ansys CFX are:

- Profile Transformation (PT)

  The PT method overcomes the unequal pitch problem by scaling the flow profile across the blade row interfaces.

  For modeling information, see Profile Transformation in the *CFX-Solver Modeling Guide* (p. 262).

- Time Transformation (TT)

  The TT method overcomes the unequal pitch problem by applying a time transformation to the flow equations. This transformation enables the use of simple periodic boundary conditions on the pitch-wise boundaries.

  For modeling information, see Time Transformation in the *CFX-Solver Modeling Guide* (p. 263).

- Fourier Transformation (FT)

  The FT method uses a phase-shifted boundary condition with Fourier data compression to account for the unequal pitch between the blade rows.

For modeling information, see Fourier Transformation in the *CFX-Solver Modeling Guide* (p. 265).

For a description of the settings used in Transient Blade Row modeling, see Transient Blade Row Models in the *CFX-Pre User's Guide*.

The Fourier Transformation and Time Transformation methods can be used to apply periodic conditions that are phase-shifted in time in order to account for unequal pitches between adjacent blade rows (for transient rotor stator cases) or to account for inequality between signal and domain pitches (for boundary disturbance cases) or to account for the inter-blade phase angle (for blade flutter cases).

Turbomachinery flow is, in general, transient and periodic. You can obtain a faster solution to the final steady-periodic state by using a frequency-based solution method rather than by marching in time. The Harmonic Analysis (HA) solution method in Ansys CFX is based on the Harmonic Balance / Time-Spectral methods and provides fast solutions for transient periodic turbomachinery flows. A use case involving HA is described in Blade Flutter using Harmonic Analysis in the *CFX-Solver Modeling Guide* (p. 283). For details on how HA relates to a transient run with time-marching, see Transient versus Harmonic Solution Method in the *CFX Reference Guide*.

For more on this topic, and for theoretical information about Profile Transformation, the Time Transformation model, the Fourier Transformation model, and Harmonic Analysis, see Transient Blade Row Modeling Theory in the *CFX-Solver Theory Guide*.

This chapter describes:

6.1.1. Transient Blade Row Modeling Terminology
6.1.2. Setting up a Transient Blade Row Model
6.1.3. Running and Postprocessing a Simulation that uses a Transient Blade Row Model
6.1.4. Profile Transformation
6.1.5. Time Transformation
6.1.6. Fourier Transformation
6.1.7. Guidelines for Using Transient Blade Row Features
6.1.8. Use Cases

6.1.1. Transient Blade Row Modeling Terminology

The following terminology applies in the context of Transient Blade Row modeling:

- *Turbo Component* - is the sector of blades in a given blade row that are being modeled in the analysis. Transient Rotor-stator analyses involve at least two components, one rotating and one stationary component.
- *Blade Flutter* - used here to describe the one-way coupling between a prescribed blade motion and the effect it has on its surrounding fluid (more properly referred to as *forced response*).
- *Conventional Periodicity* - are periodic boundaries with no special time lag for phase-shift treatment. In such cases, the rotating and stationary components are not necessarily periodic to each other at different instances of time.
- *Reference Solution* (or *Full-domain modeling*) - refers to solving a transient blade row problem using an ensemble of blade passages where the pitch ratio is one.

- *Transient Blade Row methods* - are the CFX Transient Blade Row methods:
  - Fourier Transformation method
  - Time Transformation method
  - Profile Transformation method

6.1.1.1. Abbreviations Used in this Document

Abbreviation	Expanded Form
CEL	CFX Expression Language
CHT	Conjugate Heat Transfer
EO	Engine Order (inverse of the time required for an engine component to complete a 360° rotation)
FT	Fourier Transformation
HA	Harmonic Analysis
HB	Harmonic Balance
IGV	Inlet Guide Vane
PT	Profile Transformation
STT	Single-sided Time Transformation
TBR	Transient Blade Row
TRS	Transient Rotor Stator
TSPP	Time Steps Per Period
TT	Time Transformation

6.1.2. Setting up a Transient Blade Row Model

You can use the Turbomachinery wizard in CFX-Pre to set up a Transient Blade Row model.

1. Under the Basic Setting Panel set the **Analysis Type** to `Transient Blade Row`; and select either the `Time Transformation` or `Fourier Transformation` option under **Method**.
2. Define the rotor and/or stator blade passages under **Components Definition**.

When using the Fourier Transformation method, you must set up two blade passages per component and adjust the periodic interfaces accordingly. You must set the **Passages and Alignment** > **Passages/Mesh** > **Passages to Model** to 2. This will also create the Sampling Interface required for the Fourier Transformation model.

3. Set up the physics as usual (including domains, domain interfaces, boundary conditions, and so on, as required) under the **Physics Definition** panel.

4. Under the **Interface Definition** panel you can observe and modify the various sampling and periodic interfaces that are automatically generated by the TurboMachinery wizard.

5. Under the **Disturbance Definition** panel, you will specify the periodicity of the disturbance being imposed.

6. Continue clicking **Next** until the **Final Operations** panel is reached. Set **Operation** to `Enter General Mode` because you will continue to define the simulation through settings not available in the TurboMachinery wizard.

7. Ensure that there is a `Transient Blade Row Models` object under the applicable **Flow Analysis** object. If there is not, you can add one according to the instructions provided at Inserting a New Transient Blade Row Models Object in the *CFX-Pre User's Guide*.

8. Configure the settings in the details view for the Transient Blade Models object. For details, see Transient Blade Row Models Details View in the *CFX-Pre User's Guide*.

9. Configure the output control settings. These settings are described at Transient Blade Row Results in the *CFX-Pre User's Guide*. It is a good practice to set up monitor points to check the results. For details, see Setting up Monitors to Check Results (p. 261).

For details, see Starting a New Case in Turbo Mode in the *CFX-Pre User's Guide*.

The procedure for setting up a Transient Blade Row model in General mode is as follows:

1. Set the analysis type to `Transient Blade Row`; also set the initial time.

   For details, see Analysis Type Settings in the *CFX-Pre User's Guide*.

2. Set up the physics as usual (including domains, domain interfaces, boundary conditions, and so on, as required).

   In particular, you must set the **External Passage Definition** settings (**Passages in 360** and **Pass. in Component**) on the **Basic Settings** tab for each domain.

   When using the Fourier Transformation method, you must set up two blade passages per component and adjust the periodic interfaces accordingly. The interface(s) between passages must use a GGI connection.

3. Ensure that there is a `Transient Blade Row Models` object under the applicable **Flow Analysis** object. If there is not, you can add one according to the instructions provided at Inserting a New Transient Blade Row Models Object in the *CFX-Pre User's Guide*.

4. Configure the settings in the details view for the Transient Blade Models object. For details, see Transient Blade Row Models Details View in the *CFX-Pre User's Guide*.

5. Configure the output control settings. These settings are described at Transient Blade Row Results in the *CFX-Pre User's Guide*. It is a good practice to set up monitor points to check the results. For details, see Setting up Monitors to Check Results (p. 261).

You can change an existing transient case to use Transient Blade Row models in General mode as follows:

1. Change the analysis type to `Transient Blade Row`.

   For details, see Analysis Type Settings in the *CFX-Pre User's Guide*.

2. Set the **External Passage Definition** settings (**Passages in 360** and **Pass. in Component**) on the **Basic Settings** tab for each domain.

3. When using the Fourier Transformation method, you must set up two blade passages per component and adjust the periodic interfaces accordingly. The interface(s) between passages must use a GGI connection.

4. Ensure that there is a `Transient Blade Row Models` object under the applicable **Flow Analysis** object. If there is not, you can add one according to the instructions provided at Inserting a New Transient Blade Row Models Object in the *CFX-Pre User's Guide*.

5. Configure the settings in the details view for the Transient Blade Models object. For details, see Transient Blade Row Models Details View in the *CFX-Pre User's Guide*.

6. Configure the output control settings. These settings are described under Transient Blade Row Results in the *CFX-Pre User's Guide*. It is good practice to set up monitor points to verify that a periodic solution has been achieved. For details, see Setting up Monitors to Check Results (p. 261).

Note that a given simulation can use either the Time Transformation or Fourier Transformation model, but not both. Either of these models can be used in conjunction with the Profile Transformation model. The Profile Transformation model, when used in conjunction with one of the other methods, is not explicitly selected in the user interface. For details on the Profile Transformation model, see Profile Transformation (p. 262).

6.1.2.1. Setting up Monitors to Check Results

While setting up an analysis that involves the Time Transformation or Fourier Transformation model, you should add some monitor points for diagnostic purposes. The monitor points should be placed at strategic locations, and should monitor variables that are expected to have periodic fluctuating values. During or after the run, you can check the monitor point values in CFX-Solver Manager to ensure that:

- Each monitored variable value fluctuates in a repeating pattern (periodicity).
- The period of such fluctuations correspond to the blade passing frequency of the neighboring row.

For the case of a flutter analysis, you can set up aerodynamic damping monitors to check whether vibration is damped or undamped (for the frequency being studied). For details on aerodynamic damping monitors, see Case 3: Blade Flutter (p. 277) and Aerodynamic Damping in the *CFX-Pre User's Guide*.

6.1.3. Running and Postprocessing a Simulation that uses a Transient Blade Row Model

For the case of a flutter analysis, your simulation will involve aerodynamic damping monitors, which should remain approximately constant as the flow simulation around the monitored surface reaches a quasi-periodic state. A consistently positive value of aerodynamic damping is an indication that blade vibration is damped (at the frequency being studied). For a description of the settings involved in setting up aerodynamic damping monitors, see Aerodynamic Damping in the *CFX-Pre User's Guide*.

The CFX-Solver Output file contains, for each domain, the starting and ending time steps for the data compression algorithm, as well as the fundamental period. For details, see CFX-Solver Output File (Transient Blade Row Runs) in the *CFX-Solver Manager User's Guide*.

CFD-Post has built-in data instancing capabilities. For details, see Data Instancing Tab in the *CFD-Post User's Guide*.

CFD-Post can show GPU-accelerated animations for transient blade row cases. For details, see GPU Accelerated Animation in the *CFD-Post User's Guide*.

6.1.3.1. Stopping and then Restarting Simulations with an Increased Number of Time Steps Per Period

In many flow simulations, it may be advantageous to start the simulation with a low number of time steps per period in order to pass through the initial transient phases of the solution, then later increase the number of time steps per period to accurately capture the flow physics. For a large simulation, this strategy can help reduce the overall simulation time required to reach convergence.

You can stop a Time Transformation or Fourier Transformation simulation and then resume it with an increased number of time steps per period.

An important constraint on how you increase the number of time steps for Fourier Transformation simulations is that the number must be doubled. For example you could start with 40 time steps per period, then restart with 80, then 160.

There is no such constraint for Time Transformation simulations. For example, you could start with 40 time steps per period, then restart with 50, then 70.

6.1.4. Profile Transformation

The `Profile Transformation` pitch-change method:

- Is available when there is at least one interface that uses the `Transient Rotor Stator` frame change/mixing model.
- Applies to any domain interface using the `Transient Rotor Stator` frame change/mixing model that is not used in the Time Transformation or Fourier Transformation model.
- Uses conventional periodicity at periodic boundaries (that is, has no special time lag for phase shift treatment).

- When using the **Transient Method** > **Option** of `Time Integration`, results in the same solution as a transient simulation that uses the `Transient Rotor Stator` frame change/mixing model with the same time step specifications.

Note:

See Profile Transformation in the *CFX-Solver Theory Guide* for more theoretical information about the Profile Transformation method.

6.1.5. Time Transformation

The main advantage of the `Time Transformation` pitch-change method is that conventional periodicity boundary conditions can be applied directly, resulting in relatively fast convergence.

The Time Transformation method:

- Is valid only for compressible flows
- Is valid only when the pitch ratio between the modeled components does not deviate much from unity
- Requires that the pitch ratio fall within a certain range, as described by the inequality:

$$1-\frac{M_\omega}{1-M_\theta} \quad < \quad \frac{P_S}{P_R} \quad < \quad 1+\frac{M_\omega}{1+M_\theta}$$

  where M_ω is the Mach number associated with the rotor rotational speed (or signal speed in the case of an inlet disturbance problem), M_θ is the Mach number associated with the circumferential velocity, and the ratio of P_S to P_R is the pitch ratio between the stationary component and the rotating component.

  These limits are not strict, but approaching them can cause solution instability. The CFX-Solver calculates these limits at the start of a simulation and reports them in the CFX-Solver Output file at the beginning of a run.

For most compressible turbomachinery applications (for example, gas compressors and turbines), M_ω is in the range of 0.3 - 0.6, enabling pitch ratios in the range of 0.6 - 1.5 (Giles [217]). If necessary, you can make the pitch ratio fall within the permissible range by strategically choosing the number of passages per component for each of the components.

Note:

See Time Transformation in the *CFX-Solver Theory Guide* for more theoretical information about the Time Transformation method.

6.1.5.1. Time Transformation: Workflow Requirements

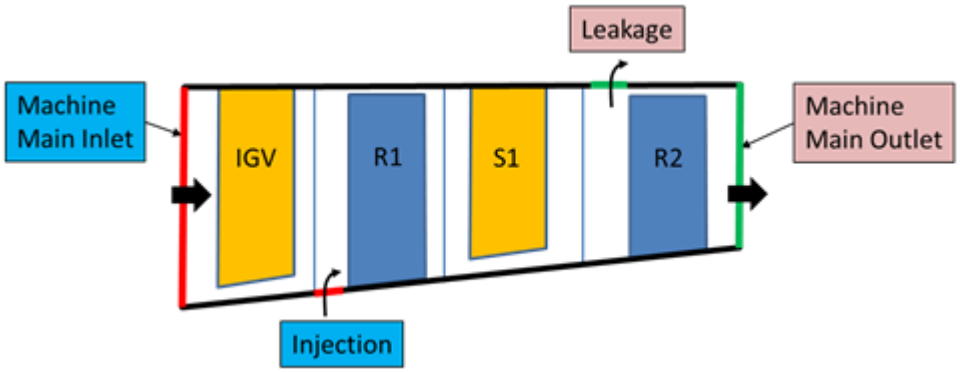
The basic procedure for setting up a Transient Blade Row model is outlined in Setting up a Transient Blade Row Model (p. 259). For simulations involving the Time Transformation model:

- Ensure that the pitch ratio between adjacent components is within the proper range by appropriately adjusting the number of blades in each component. The proper range is described in Time Transformation (p. 263).
- Those transient rotor stator interfaces not referenced by a disturbance will be treated with the Profile Transformation method.
- You can model only a single stage. As a result, there must be exactly one domain interface between two time transformed domains.
- Based on the case you are working on, for a given domain, add one disturbance to the list of disturbances.
- Ensure that the relative speed associated with the disturbance is not zero.

Note:

When multiple Time Transformation instances are used, the solver will attempt to compute the streamwise scale equation(*). In cases where the machine contains more than one inlet or outlet (for example, due to the presence of injection holes or cooling grooves) you must manually specify the main inlet and/or main outlet of the machine; that is done by entering the following lines of CCL:

```
TRANSIENT BLADE ROW MODELS:
   Inflow Boundary = <main inlet name>
   Outflow Boundary = <main outlet name>
   Option = Time Transformation
   .
   .
   .
  END
```



Failure to specify the main inlet and/or outlet will result in the solver issuing the following message:

```
Multiple Inlets and/or Openings were detected for this multi-disturbance Time Transformation
  simulation. Please specify one inlet as streamwise scale specified boundary condition region.
```

(*) The streamwise scale equation is used to compute the streamwise location of the machine components.

6.1.6. Fourier Transformation

Unlike the Time Transformation pitch-change method, the Fourier Transformation pitch-change method does not have any known limitations on the pitch ratio (or nodal diameter for a blade flutter application) and works with both compressible and incompressible flows.

The Fourier transformation method can be used in cases involving:

- Rotational flow boundary disturbances
- Blade flutter (or forced response)
- Transient rotor stator interactions (single stage only)

Note:

See Fourier Transformation in the *CFX-Solver Theory Guide* for more theoretical information about the Fourier Transformation method.

6.1.6.1. Fourier Transformation: Workflow Requirements

The basic procedure for setting up a Transient Blade Row model is outlined in Setting up a Transient Blade Row Model (p. 259). For simulations involving the Fourier Transformation model:

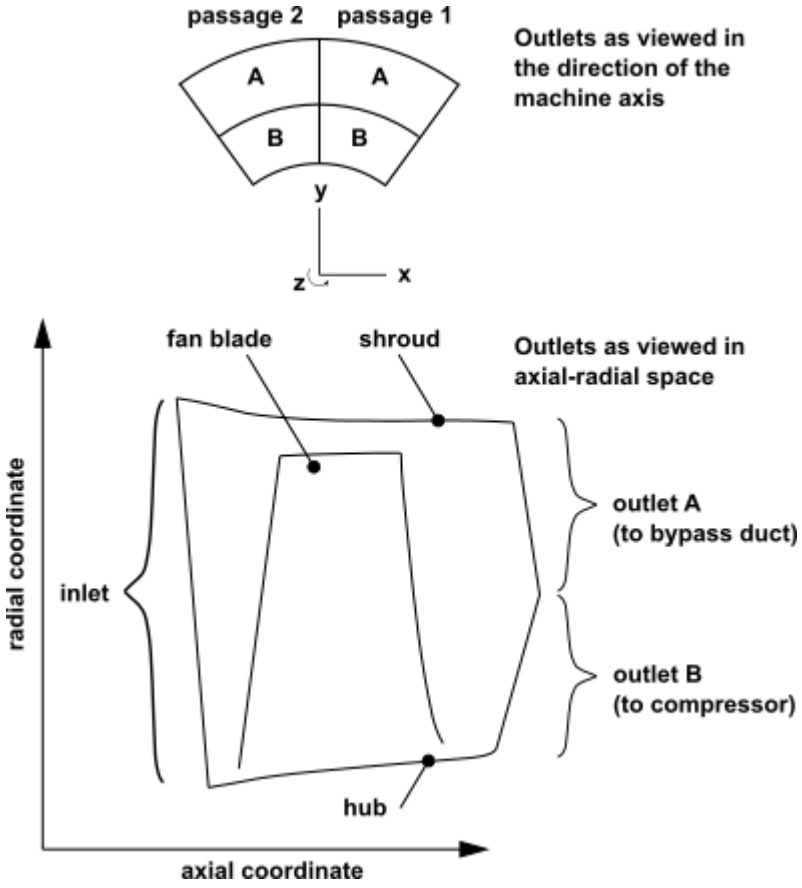
- Each turbo component requires two instances of the passages that constitute a repeatable section. A GGI interface must be created to join the two instances. This domain interface will be selected as the sampling interface. A phase-shifted periodic domain interface connection is required on the outer periodic boundaries. This periodic interface is selected when defining the phase corrected interface. See Fourier Transformation Disturbance Settings in the *CFX-Pre User's Guide* for a description of the sampling domain interface.
- For a given domain, optionally add one disturbance to the list of disturbances.
- Those transient rotor stator interfaces not referenced by a disturbance will be treated with the Profile Transformation method.
- All periodic and sampling interface pairs must use the `Bitmap` intersection method. The option that controls the intersection method is found in CFX-Pre in the details view for the domain interface, on the **Basic Settings** tab, under **Mesh Connection Method** > **Mesh Connection** > **Intersection Control**. For periodic and sampling interface pairs, the `Bitmap` method takes effect when the **Intersection Control** check box is cleared. When the **Intersection Control** check box is selected, the **Option** setting is set to `Bitmap` by default.
- The Transient Rotor Stator interface must use the `Direct` intersection method. The option that controls the intersection method is found in CFX-Pre in the details view for the domain interface, on the **Basic Settings** tab, under **Mesh Connection Method** > **Mesh Connection** > **Intersection Control**. For a Transient Rotor Stator interface, the `Direct` method takes effect when the **Intersection Control** check box is cleared. If you select the **Intersection Control** check box, you must change the **Option** setting (from its default value of `Bitmap`) to `Direct`.
- Ensure that no disturbance signal has a speed of zero relative to the component on which that signal acts.

- When solving a simulation that involves the Fourier Transformation method (including the steady-state simulation for initialization), the use of double precision is strongly recommended for accuracy and robustness.
- Some inlet boundaries might be required to have identical conditions, depending on the geometry. The same applies for outlet boundaries.

  Consider a geometry as viewed in axial-radial space, with the axial coordinate measuring distance along the machine axis and the radial coordinate measuring distance from the machine axis. As viewed in this space, boundaries that overlap must have identical specified conditions. Boundaries that do not overlap, including those that only meet at a point, can have different specified conditions.

  For example, consider the outlets in Figure 6.1: Example Fan Geometry (p. 266) as viewed in axial-radial space. Outlet A of passage 1 and outlet A of passage 2 overlap, so must have identical specified conditions. Similarly, outlet B of passage 1 and outlet B of passage 2 overlap, so must have identical specified conditions. Outlet A (of any passage) and outlet B (of any passage) do not overlap, so can have different specified conditions.

Figure 6.1: Example Fan Geometry



6.1.7. Guidelines for Using Transient Blade Row Features

Guidelines for using Transient Blade Row features are described in the following sections:

6.1.7.1. General Setup Guidelines

6.1.7.2. Guidelines for using the Time Transformation Feature
6.1.7.3. Guidelines for using the Fourier Transformation Feature
6.1.7.4. General Postprocessing Guidelines

6.1.7.1. General Setup Guidelines

- CFX-Pre Turbomachinery Wizard:
  - When setting up your transient blade row problem, it is always recommended that you use the Turbomachinery wizard in CFX-Pre.
- Flow initialization:
  - A steady-state solution can be used as the initial condition for all transient flow simulations. You may use the Stage model (mixing-plane) or Frozen Rotor method, when modeling multiple blade rows.
- Time-integration:
  - For all transient blade row simulations that have **Transient Method** > **Option** set to `Time Integration`, a uniform time step must be used.
  - You should not change the time step (number of time steps per period or timestep multiplier) in the middle of a simulation. However, you can stop a simulation, change the time step, then restart (continue) the simulation. For details, see Stopping and then Restarting Simulations with an Increased Number of Time Steps Per Period (p. 262).
- Inflow and Outflow Boundary Conditions:
  - For proper comparison between the full-domain solution and any of the transient blade row solutions, the inflow and outflow boundary conditions must be equivalent. For example: an average pressure specification on an outlet boundary should not apply on the full extent of the outlet in the full domain model but rather it should be applied to individual passage outlets to replicate the transient blade row solution boundary condition specification. Therefore it is advised to divide the full-domain inlet and outlet boundaries into multiple boundaries accordingly.
  - For an inlet disturbance problem: When working with profiles as boundary conditions, be sure to double-check the periodicity of the profiles on both reference test cases and the transient blade row test cases. You can use monitors at the inlet to do this check.
- Solution Monitoring:
  - You can use solution monitor points to monitor the results of the transient simulation.
  - When comparing the full-domain reference solution with the transient blade row solution, it is important to locate the monitors in the passages of the full-domain geometry with relative locations as in the transient blade row passage configuration.

6.1.7.2. Guidelines for using the Time Transformation Feature

- Pitch ratio:

  - There is a limit on the range of pitch-ratio allowed when using the Time Transformation method. This range is strongly dependent on the Mach number associated with the rotor rotational speed (or signal speed in the case of an inlet disturbance problem). The pitch-ratio limit is defined in Time Transformation (p. 263).

- Solution Monitoring

  - The Time Transformation method does not support monitor points that use CEL expressions. Instead, you can use solution variables (pressure, temperature, and so on) to monitor the solution.

6.1.7.3. Guidelines for using the Fourier Transformation Feature

The Fourier Transformation method works for compressible as well as incompressible flows and for all ranges of disturbance pitch ratios from small to very large. For example, the Fourier Transformation method can handle a once-per-revolution disturbance on a blade passage, which is a situation that cannot be handled with the Time Transformation method.

However, in general for compressible flow and with small pitch ratios (typically between 0.75 and 1.35) it is recommended that you use the Time Transformation method instead because the calculation will be more efficient than when using the Fourier Transformation method.

- When not using the Turbomachinery wizard in CFX-Pre, an additional step is needed for creating turbo topology; you must create two instances of the blade passage.

  There are two options by which you can create a copy of the original passage:

  - Perform a rotation on the copy of the original geometry (mesh) by its own pitch. The rotation angle must be specified with many decimal digits for minimum round-off error.

  - Perform a turbo rotation and update the number of passages to model and the number of passages in 360° in each domain.

- When running a case in serial, it is highly recommended that you run CFX-Solver using double precision. This recommendation also applies for steady-state cases that are used to initialize Fourier Transformation cases.

- When running a case in parallel:

  - You may run CFX-Solver using single precision, but it is highly recommended that the partitioning is done using double precision. This recommendation also applies for steady-state cases that are used to initialize Fourier Transformation cases.

  - Each pair of sampling/phase shift corrected interfaces must belong to a single domain. This is a condition that should be guaranteed if CFX-Pre Turbo Wizard is used to set up the case.

- When setting up a Fourier Transformation case, any domain interface that is part of the Fourier Transformation Transient Blade Row setup should not span more than one domain.

- It is very important that the boundary condition is periodic in time for the passages being modeled. In the case of blade flutter, this can be achieved by selecting a number of timesteps that is a multiple of $4*\frac{Number\ of\ Passages\ in\ 360}{Phase\ Angle\ Multiplier}$.

- In cases where you observe solution instabilities, you should apply frequency filtering. Instabilities are typically seen in elongated domains where the periodic signal becomes very weak at the furthest point away from the source of the disturbance. Frequency filtering removes unwanted frequencies from the solution thereby avoiding the onset of instability. The setting for enabling frequency filtering is described in Frequency Filtering in the *CFX-Pre User's Guide*.
- Instabilities could also exist related to the mass flow Fourier reconstruction for cases with high speed flow where a shock hits the periodic interface. The recommended workaround is to turn off the mass flow Fourier reconstruction with one of the following expert parameters:
  - For inlet disturbance and flutter cases: `ps reconstr mflow from sfc = f`
  - For cases involving a transient rotor stator Fourier interface: `ps reconstr mflow from sfc trs = f`

6.1.7.4. General Postprocessing Guidelines

- Animations:
  - Because the resulting animations of transient blade row cases are usually periodic in time, you must take care not to repeat the start/end point of the period when these are identical. The animation panel has an advanced option to avoid this (the option is named **Don't Encode Last MPEG frame**), and you should select this option when working with a transient blade row animation.
- Note that, for a continued run of a transient blade row case, the `Accumulated Time Step` variable does not reflect the solver run history.
- Note that hybrid values of injection quantities might not be available for FT and HA cases.

6.1.8. Use Cases

The use cases targeted with these multi-blade row models include:

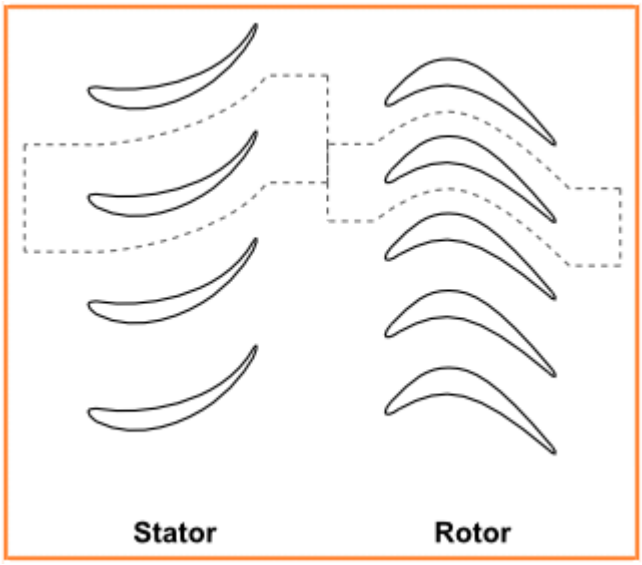
6.1.8.1. Case 1: Transient Rotor Stator Single Stage
6.1.8.2. Case 2: Flow Boundary Disturbance
6.1.8.3. Case 3: Blade Flutter
6.1.8.4. Case 4: Harmonic Forced Response
6.1.8.5. Case 5: Transient Rotor Stator Multi-Stage Cases
6.1.8.6. Case 6: Transient Rotor Stator Cases with Asymmetric Flow

6.1.8.1. Case 1: Transient Rotor Stator Single Stage

The transient rotor stator (TRS) single stage case is supported by the following pitch-change methods: Profile Transformation, Time Transformation, and Fourier Transformation. Each method has different requirements:

- Profile Transformation

Figure 6.2: Typical Transient Rotor Stator Simulation using Profile Transformation

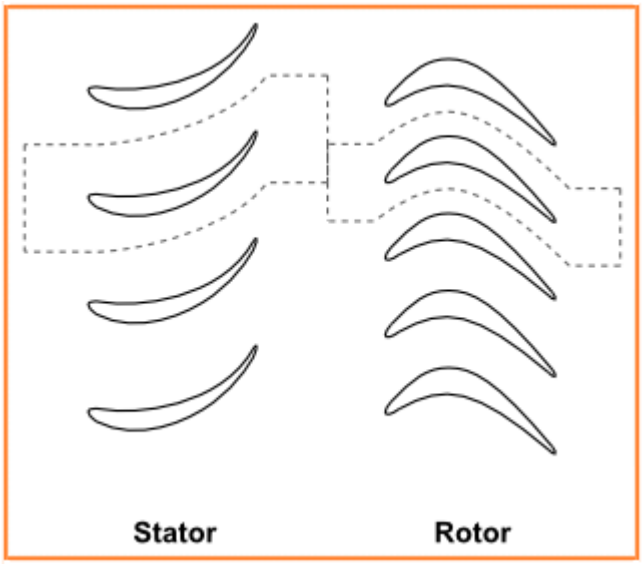


  – For each component, you can model one (or an arbitrary number of) passage(s).

  – Harmonic Analysis is available; for details, see Profile Transformation and Fourier Transformation using Harmonic Analysis (p. 271).

- Time Transformation

Figure 6.3: Typical Transient Rotor Stator Simulation using Time Transformation

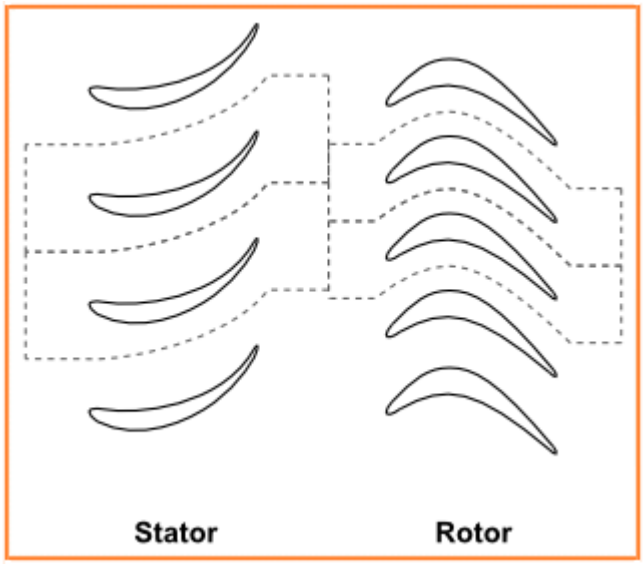


  – You must select the transient rotor stator interface to which the transformation applies.

  – No additional parameters are needed if the 'Automatic' selection of domains is applied for either side of the interface.

- Fourier Transformation

Figure 6.4: Typical Transient Rotor Stator Simulation using Fourier Transformation



– For each component, you must model two instances of the domain, or group of domains (see Fourier Transformation: Workflow Requirements (p. 265)).

– You must select the transient rotor stator interface to which this transformation applies.

– You must specify, for each domain connected to the rotor-stator interface: (a) which interface is the sampling interface and (b) which interface is the phase-shifted periodic interface (to be used for the application of phase-corrections).

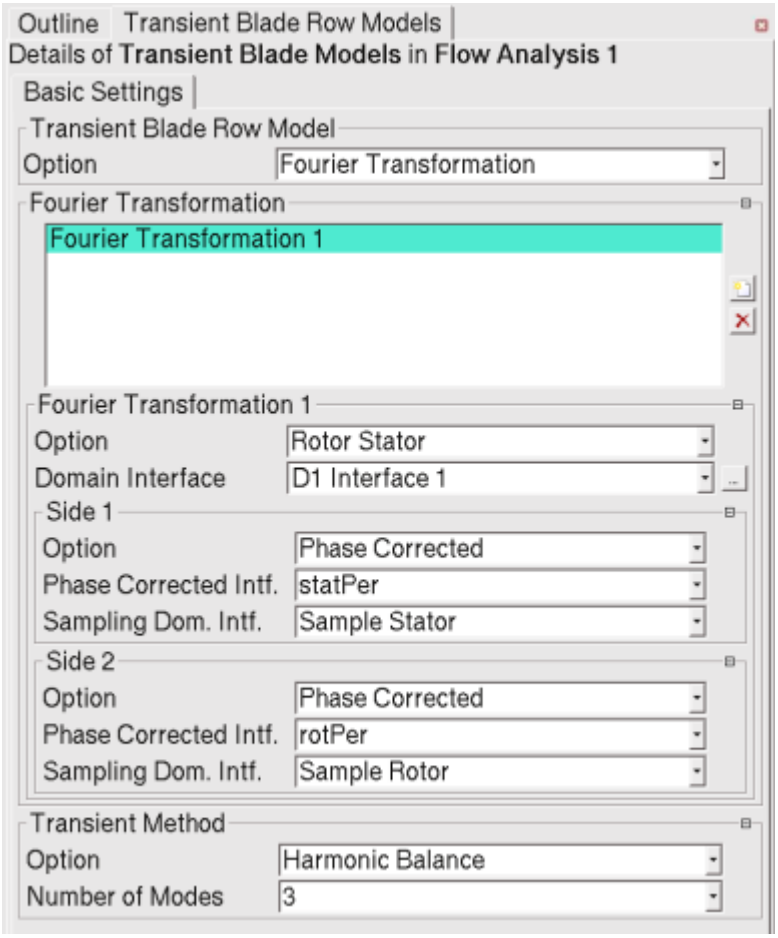
– Harmonic Analysis is available; for details, see Profile Transformation and Fourier Transformation using Harmonic Analysis (p. 271).

6.1.8.1.1. Profile Transformation and Fourier Transformation using Harmonic Analysis

A single-stage rotor-stator simulation can be performed using the Profile Transformation or Fourier Transformation pitch-change method. Compared to a time-marching simulation, a significant reduction in computational time can be achieved by performing the simulation using the Harmonic Analysis (HA) solution method, which avoids the need for solving in time over many disturbance periods in order to reach a steady-periodic state. For details, see Harmonic Analysis in the *CFX-Solver Theory Guide*.

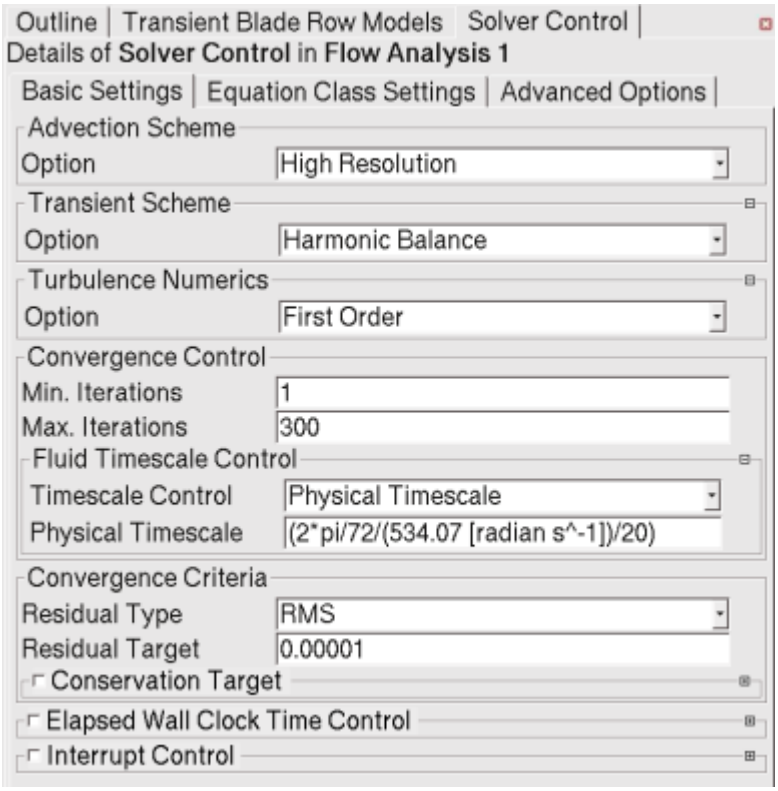
A single-stage rotor-stator simulation that uses the HA methodology is set up like a single-stage rotor-stator simulation that uses the time-marching method, except:

- In the **Transient Blade Row Models** details view, on the **Basic Settings** tab, you must set **Transient Method** > **Option** to `Harmonic Balance` and **Transient Method** > **Number of Modes** to an appropriate value (usually 3 or larger).



For a Profile Transformation with HA, you must also specify the time period, either by providing a value or by specifying the passing period of a rotating domain.

- In the **Solver Control** details view, on the **Basic Settings** tab, ensure that **Transient Scheme** > **Option** is set to `Harmonic Balance`, specify the maximum number of iterations (**Max. Iterations**), and set the pseudo-time step size (**Fluid Timescale Control** settings). Typically, the size of the pseudo-time step (**Physical Timescale**) is equal to the rotor blade passing period divided by the intended number of pseudo-time steps per period.



A typical number of pseudo-time steps per period is between 15 and 30. Recall that the lower the number of pseudo-time steps that can be used while maintaining a stable solution, the more efficient the calculation is with respect to the time marching method.

6.1.8.2. Case 2: Flow Boundary Disturbance

A flow boundary disturbance case involves modeling the effect of an external component (a virtual blade row upstream or downstream of the fluid mesh) via an inlet or outlet boundary condition that varies in time. Such cases are supported by the Time Transformation and Fourier Transformation models.

This type of simulation can be used as a lower-order approximation to a stage simulation. In inlet disturbance simulations the upstream component is modeled by prescribing a profile that is at the inlet and that models the wake of the upstream component.

You can prescribe wake profiles via CEL expressions or via external `*.csv` files. CEL-based profiles need not be formulated to produce rotational motion in time. If a profile does not cover the whole 360° wheel, then it should consist of a periodic portion that can be replicated to produce a 360° profile. You may optionally produce a 360° profile in a `*.csv` file based on a smaller profile file by using the **Edit Profile Data** dialog box in CFX-Pre (see Edit Profile Data in the *CFX-Pre User's Guide*).

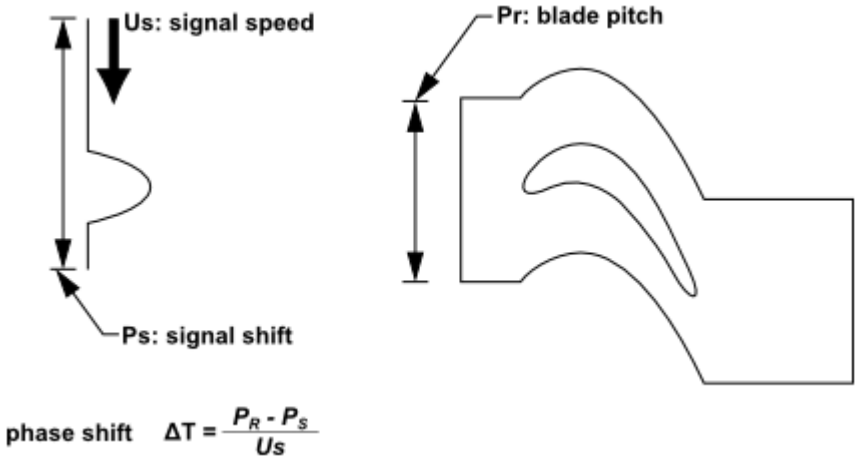
Whether using a CEL expression or a `*.csv` file, you can cause the profile data to move (typically rotationally, about the machine axis) by associating the boundary with a moving coordinate frame. For details, see:

- Local Coordinate Frames (p. 76)
- Frame Motion Settings in the *CFX-Pre User's Guide*

- Coordinate Frame in the *CFX-Pre User's Guide*
- Time Transformation Disturbance Settings in the *CFX-Pre User's Guide*
- Fourier Transformation Disturbance Settings in the *CFX-Pre User's Guide*

For a rotational-flow disturbance case, the required **External Passage Definition** input parameters are aimed at specifying the pitch of the profile (or signal) at the inlet/outlet. These are needed to help compute the phase shift (ΔT of Figure 6.5: Inlet Disturbance Case (p. 274)). These parameters are important to guarantee that the resultant signal has periodic behavior.

Figure 6.5: Inlet Disturbance Case

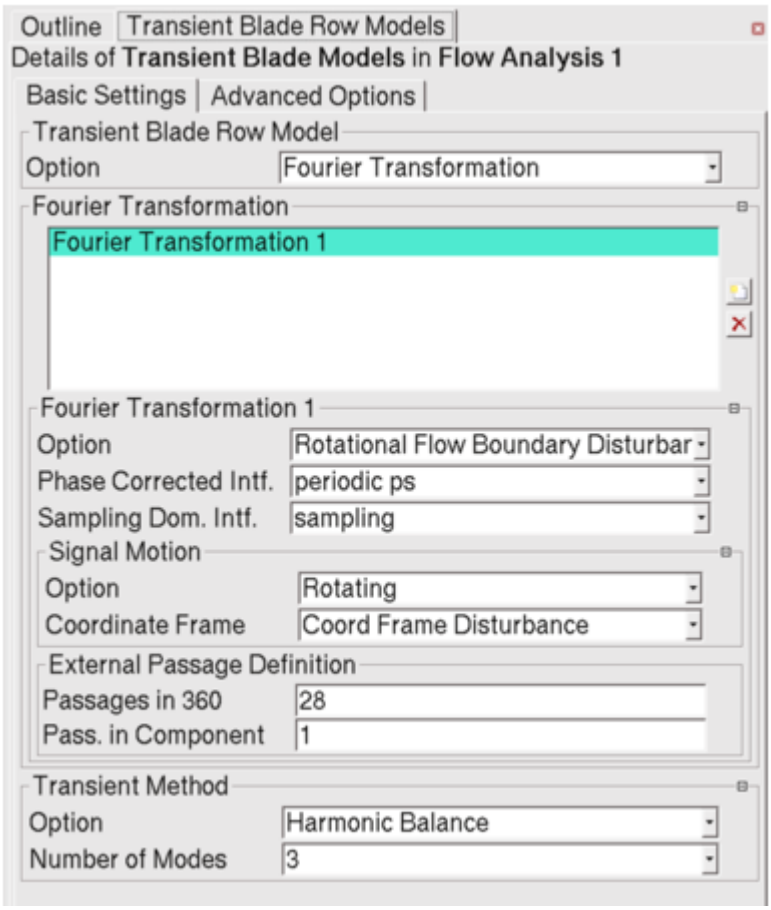


6.1.8.2.1. Flow Boundary Disturbance using Harmonic Analysis

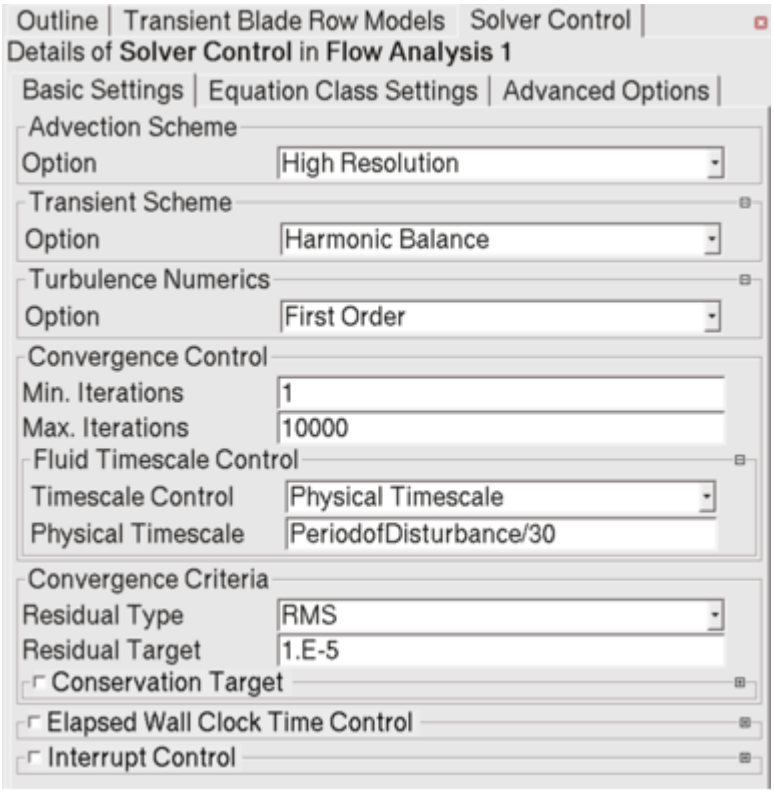
As described above, a Flow Boundary Disturbance calculation can be performed on two blade passages using the Fourier-Transformation pitch-change method. The solution is obtained faster than for a full-wheel calculation. A significant further reduction in computational time can be achieved by performing the computation using the Harmonic Analysis (HA) (see Harmonic Analysis in the *CFX-Solver Theory Guide*) solution method, which avoids the need for solving in time over many disturbance periods in order to reach a steady-periodic state.

A flow boundary disturbance simulation that uses the HA methodology is set up like a flow boundary disturbance simulation that uses the time-marching method, except:

- In the **Transient Blade Row Models** details view, on the **Basic Settings** tab, you must set **Transient Method** > **Option** to `Harmonic Balance` and **Transient Method** > **Number of Modes** to an appropriate value.



- In the **Solver Control** details view, on the **Basic Settings** tab, ensure that **Transient Scheme** > **Option** is set to `Harmonic Balance`, specify the maximum number of iterations (**Max. Iterations**), and set the pseudo-time step size (**Fluid Timescale Control** settings). Typically, the size of the pseudo-time step (**Physical Timescale**) is equal to the disturbance period divided by the intended number of pseudo-time steps per period.



A typical number of pseudo-time steps per period is between 15 and 30. Recall that the lower the number of pseudo-time steps that can be used while maintaining a stable solution, the more efficient the calculation is with respect to the time marching method.

6.1.8.2.2. Multiple Disturbances

A multi-disturbance gust analysis can be modeled using the Fourier Transformation method. In this analysis, the blade is subjected to two simultaneous disturbances; one on the inlet and the other on the outlet of the blade passage. Typically, the inlet and outlet profiles are initially extracted from a steady-state stage simulation of a 1.5 stage configuration. The profiles are then imposed on the inlet and outlet of the rotor passage in a transient gust simulation. This modeling strategy can be used as an alternative for modeling a 1.5 stage compressor or turbine. It accounts for blade passing frequencies from the upstream and downstream rows on the modeled blades. The use of the Fourier Transformation method also allows for very large pitch variations such as with a fan subjected to inflow distortion.

In general, the procedure to set up a multi-disturbance gust analysis with the Fourier Transformation method closely follows the setup of an inlet disturbance problem with the Fourier Transformation method (see Fourier Transformation Disturbance Settings in the *CFX-Pre User's Guide*). Note the following important steps:

- After setting up the first disturbance, you must add a second disturbance from the same **Transient Blade Row Models** details view.
- Using the Transient Analysis Method:
  - Specifying the time step in the **Transient Blade Row Models** details view requires two steps: (1) Specifying a meaningful time period under **Transient Method** > **Time Period**. This can

be the period of one of the disturbances or the common period between all the disturbances; (2) Specifying **Timesteps/Period**.

- Using the Harmonic Analysis Method:
  - In the **Transient Blade Row Models** details view, the number of modes needs to be specified under **Transient Method** > **Number of Modes**. This represents the number of harmonics of each of the disturbances that will be used in the simulation. For example, specifying 3 modes for a case that has 2 boundary disturbances will result in 6 modes and will require a total of 13 time planes to solve.
- Ensure that all the boundary disturbances are set up correctly. For example, in a two-disturbance case (inlet and outlet) both inlet and outlet boundary conditions must now be specified with a disturbance either through CEL or specified profiles.
- For stationary disturbances, ensure that you create and refer to a stationary coordinate frame in the **Boundary** details view as well as the **Transient Blade Row Models** details view.

Note:

- It is not ideal to use this modeling technique if strong outlet recirculation is present at the boundary.
- If the case is not periodic by nature then this modeling method will either artificially make the solution periodic or the solution may fail to converge.

6.1.8.3. Case 3: Blade Flutter

Aerodynamic damping calculations are used in the design of turbines, compressors, and fans to predict failures due to blade flutter and to estimate blade life cycles.

Blade flutter and damping calculations can be performed in Ansys CFX by solving for flow over vibrating blades with prescribed blade displacements. Blade flutter usually occurs at the natural frequency of the blade-disc assembly. This natural frequency can be determined, along with blade displacements (mode shapes), using Ansys Mechanical prior to performing a flow analysis on the blade. In general, the natural disc frequency of a rotor-disc assembly consists of many modes. However, only a limited number of these modes contribute to flutter and only these modes are considered during a blade flutter simulation setup.

For blade flutter calculations, it is important to consider the aerodynamic effects of adjacent blades. For a rotor-disc assembly consisting of N_{BL} blades, there is a finite number of disc nodal diameters. At nodal diameter $ND=0$, all the blades vibrate in phase with an Interblade Phase Angle (IBPA) of 0. However, for other nodal diameters, each blade will be out of phase with respect to the others by a finite interblade phase angle value. The interblade phase angle can be computed from the nodal diameter as:

$$IBPA=\sigma=\frac{2\pi ND}{N_{BL}} \tag{6.1}$$

where $0\leq ND\leq N_{BL}/2$ for an even number of blades and $0\leq ND\leq\left(N_{BL}-1\right)/2$ for an odd number of blades.

In CFX you can set up a full domain transient model to perform a blade flutter simulation. Alternatively, you can use the Fourier Transformation method to perform a blade flutter analysis using a truncated domain with two passages.

The main objective of a blade flutter analysis is to obtain the aerodynamic damping, or work per vibration cycle. Once a transient periodic solution is achieved, you can use the energy balance method to perform the damping calculation. This calculation yields a measure of the system stability at each nodal diameter and blade vibration frequency. The work per vibration cycle can be computed as:

$$W_{cycle} = \int_{t_0}^{t_0+T} \int_{A} p\vec{v} \cdot \hat{n} \, dA dt \tag{6.2}$$

Where:

- $T = 2\pi / \omega$ (that is, the period of one vibration cycle)
- t_0 is the time at the start of the vibration cycle.
- ω is the vibration frequency (that is, the **Frequency** value specified on the **Boundary Details** tab; see Periodic Displacement in the *CFX-Solver Modeling Guide* (p. 48))
- p is the fluid pressure
- v is the velocity of the blade due to imposed vibrational displacements
- A is the surface area (in this case, the surface of the blade)
- $\hat{n}$ is the surface normal unit vector.

Note:

Positive values for work indicate that the vibration is damped (for the frequency being studied), whereas negative values indicate that the vibration is undamped.

Aerodynamic damping monitors can be used to monitor aerodynamic damping during a run. The monitor value is calculated according to Equation 6.2 (p. 278) subject to the following modifications and conditions:

- The integration limits are set to encompass an *integration interval* rather than a vibration cycle. The integration interval size is a multiple of the period of one vibration cycle, the multiple being the value specified for the **Integration Multiplier** setting, described at [Aerodynamic Damping Name]: Integration Multiplier in the *CFX-Pre User's Guide*. The resulting integral, which represents the work done over the course of one integration interval, is divided by the integration multiplier. For an integration multiplier of one (default), the result is the (unnormalized) aerodynamic damping value. For an integration multiplier greater than one, the result is an average value of (unnormalized) aerodynamic damping (work per vibration cycle) as evaluated over the integration interval (that is, multiple vibration cycles).
- The upper time limit on the integral can be at the end of the last *complete* integration interval or at the current time step, depending on whether the aerodynamic damping option (described

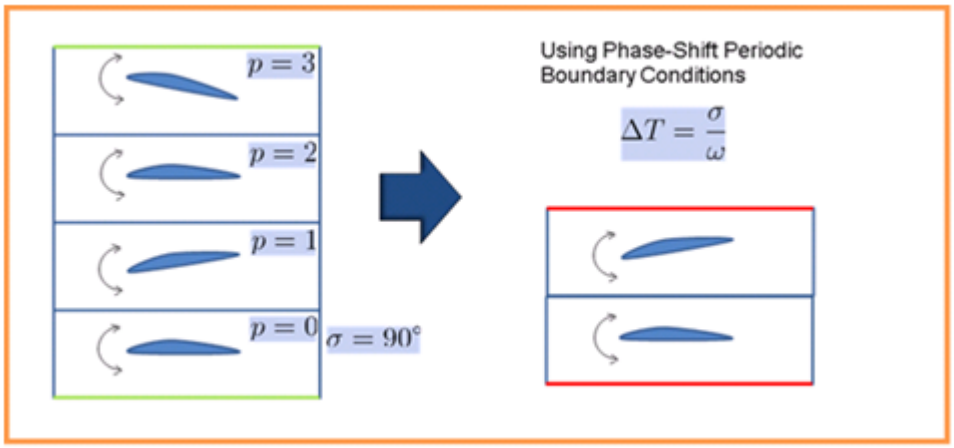
at [Aerodynamic Damping Name]: Option in the *CFX-Pre User's Guide*) is set to `Full Period Integration` or `Moving Integration Interval`, respectively.

- The resulting value (unnormalized aerodynamic damping) is then divided by the specified normalization value, described at [Aerodynamic Damping Name]: Normalization in the *CFX-Pre User's Guide*, to yield a normalized value of aerodynamic damping for the aerodynamic damping monitor.

Note:

After a new run is started, an aerodynamic damping monitor reports a value of zero until the required integration interval has passed.

Figure 6.6: Application of Phase-Shifted Boundaries to a Blade Flutter Case



For each component, you must model two instances of the passage, or group of passages, that is repeatable (see Fourier Transformation: Workflow Requirements (p. 265)).

The vibration of the blades with **Interblade Phase Angle** (IBPA) results in a traveling wave pattern.

To complete the model, you must specify the following information:

- The modal **Frequency**.
- Information about the traveling wave.

  This information can be specified using one of the **Phase Angle** options for a boundary that has the `Periodic Displacement` mesh motion option (see Periodic Displacement (p. 48)). For example, with the `Nodal Diameter (Phase Angle Multiplier)` option, you can specify the traveling wave direction: `Forward` or `Backward`.

For the traveling wave direction, the direction considered to be forward is (by default):

- For a model that has a rotating component: the direction of machine rotation. A Forward Traveling Wave (FTW) is associated with a positive IBPA value. Parts (a) and (b) of Figure 6.7: Direction of Forward Traveling Wave (p. 280) show the forward direction for a rotating component.
- For a model that has only stationary components (for example, a model of a stand-alone stator): the direction of increasing sector tag number. Part (c) of Figure 6.7: Direction of Forward Traveling

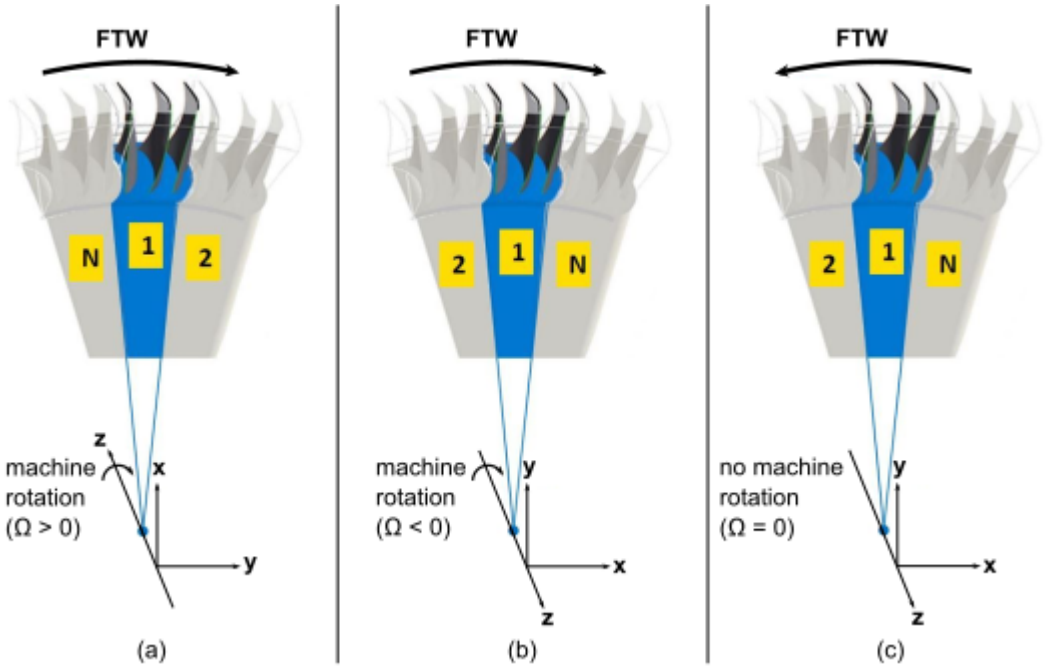
Wave (p. 280) shows the forward direction for a component in a model having no rotating components.

Note:

You can change which direction is considered to be forward (corresponding to a positive phase angle). For details, see the description for expert parameter `meshdisp phase angle convention` given in Physical Model Parameters (p. 618).

In CFX, sector tagging is independent of machine rotation. The sector tag number starts at 1 (one) for the original sector and, as shown in Figure 6.7: Direction of Forward Traveling Wave (p. 280), increases around the wheel in a direction established using the right hand rule and the global coordinate frame.

Figure 6.7: Direction of Forward Traveling Wave



To compute the magnitude of the nodal diameter, divide the magnitude of the IBPA by the blade pitch in radians. The magnitude of the IBPA can be defined as $2\pi[\mathrm{rad}](\Delta T/T)$, where ΔT is the time shift between adjacent blades. The blade pitch in radians is $2\pi/N_{\mathrm{BL}}$, where N_{BL} is the number of blades. The magnitude of the nodal diameter can therefore be computed as $N_{\mathrm{BL}}\Delta T/T$.

Note:

The size of the computational model for the analysis of a blade flutter case that uses standard periodicity boundary conditions, such as the one illustrated in Figure 6.6: Application of Phase-Shifted Boundaries to a Blade Flutter Case (p. 279), is heavily dependent on the number of passages and the nodal diameter. For most cases, a large sector of the blade row must be modeled. When you account for the phase shift at periodic

boundaries by using the Fourier Transformation method, the computational model only needs to include two passages for any nodal diameter.

Note:

If splitter blades are involved, then for the purpose of calculating the magnitude of the nodal diameter, the blade pitch should be measured from one main blade to the next.

Note:

For cases where the mesh motion on a given boundary straddles stationary interfaces, there is a risk of mesh folding. In order to prevent this problem, the Ansys CFX-Solver automatically blends the mesh motion displacement amplitude down to zero on boundary nodes approaching the stationary interfaces. This strategy, referred to as "Zero Displacement Approximation", is always used because the Fourier Transformation Blade Flutter model requires stationary interfaces. The "Zero Displacement Approximation" might be the cause of solution discrepancies between a Fourier Transformation Blade Flutter model and its equivalent Symmetric Blade Flutter case, for example, when the displacements across stationary interfaces are large compared to displacements elsewhere in the domain.

6.1.8.3.1. Setting Up a Blade Flutter Simulation

The fundamental input, which is the modal shape of the structure, must be computed and provided in the Ansys CFX `.csv` file format (see Profile Data Format in the *CFX-Pre User's Guide*). This file should contain the coordinates, real modal displacements, and complex modal displacements (only for cyclic symmetry models) at the structural mesh locations. For cyclic symmetry cases (usually involving complex mode shapes), the harmonic index (nodal diameter) should also be provided. A sample `.csv` mode shape profile is shown below.

```
[Name]
normmodeshape

[Parameters]
Ncompt = 3
Nnodes = 4404
Harmonic Index = 4
Mass = 7.25811591 [kg]
Frequency = 1422.69639 [Hz]
Maximum Displacement = 2.43392589 [m]

[Spatial Fields]
Initial X,Initial Y,Initial Z

[Data]
Initial X [m], Initial Y [m], Initial Z [m], node [], norm imag meshdisptot x [], norm imag meshdisptot y [
2.62491941e-001, -1.55315742e-001, -2.57421546e-002, 2.48000000e+002, 3.58961582e-001, 1.76829070e-001, 7.2
2.63363600e-001, -1.53833091e-001, -2.57421546e-002, 2.49000000e+002, 3.52993071e-001, 1.80246934e-001, 7.3
...
```

A typical blade flutter calculation, whether made using the symmetric sector model or a reduced model, requires modeling several blade passages, which in turn requires expansion of the structural mode shape. The latter can be accomplished using the **Edit Profile Data** tool provided in

CFX-Pre (see Edit Profile Data in the *CFX-Pre User's Guide*). This tool can replicate as many copies as required by the model and can expand a cyclic symmetry modal shape using the Harmonic Index provided in the file.

Once the mode shape has been imported, it can be visualized to verify that it is aligned with the existing Ansys CFX model before using it in a boundary condition. In case the structural mode shape is not aligned with the model, it must be aligned either using a local coordinate frame once imported, or by external means.

It is highly recommended that you use the automated facility for generating boundary conditions from profiles (see Use Profile Data in the *CFX-Pre User's Guide*) because it will automatically fill in most of the boundary condition details from the available information in the profile. It is imperative that the settings include the intended maximum amplitude for the deformation of the boundary and the wave traveling direction. For real modal shapes, the Interblade Phase Angle (IBPA) or nodal diameter must also be provided.

While setting up a boundary condition using the structural modal shape, keep in mind that there are two ways of setting up a blade flutter case:

- For mode shape displacements that can be used for a wide range of nodal diameters (typically associated with real mode shapes):

  After expanding the profile to the appropriate number of sectors, the mesh motion defined by the mode shapes is entered using the `Periodic Displacement` mesh motion option. You can then run the case with any IBPA or nodal diameter by adjusting the specified **Phase Angle** or **Nodal Diameter**.

- For mode shape displacements corresponding to a specific nodal diameter (typically associated with complex mode shapes from cyclic symmetry calculations):

  After instancing the profile to the appropriate number of sectors, the mesh motion defined by the complex mode shapes is entered using the `Periodic Displacement` mesh motion option. The phase angle or the nodal diameter cannot be adjusted because its value is obtained from the expanded profile. To run the case with another IBPA or nodal diameter requires that you specify a corresponding mode shape displacement.

6.1.8.3.2. Running a Blade Flutter Simulation

When running blade flutter simulations, consider doing some or all of the following:

- Use the `Blended Distance and Small Volumes` stiffness model. For details, see Blended Distance and Small Volumes (p. 45).

- Run with mesh pre-coarsening, which is a partitioning option for parallel runs. This is particularly appropriate if you experience robustness problems during a run. For details, see Mesh Pre-coarsening in the *CFX-Solver Manager User's Guide*.

- Lower the residual target for mesh convergence from 1.0e-04 to 1.0e-05.

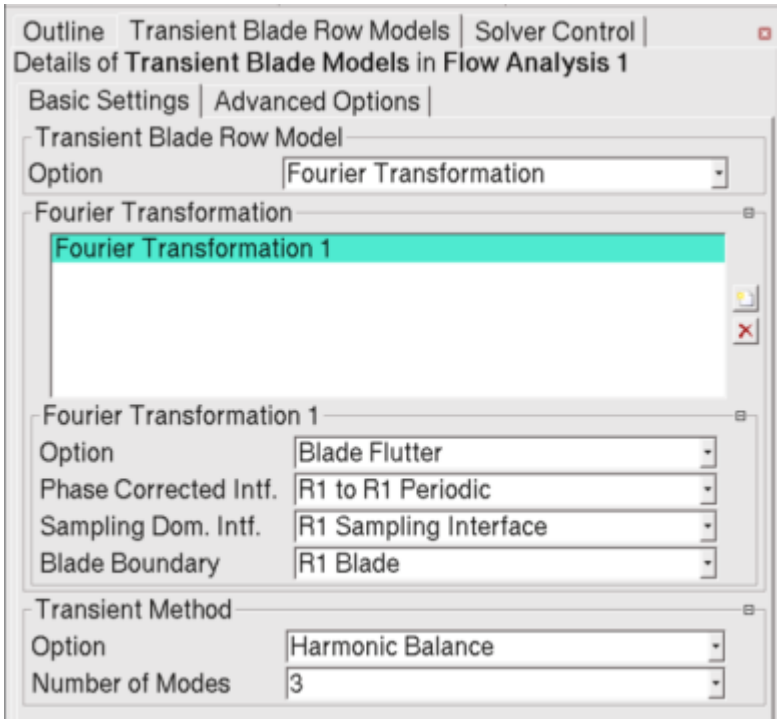
- Increase the maximum number of coefficient loops to ensure well-converged mesh motion at every time step.

6.1.8.3.3. Blade Flutter using Harmonic Analysis

As described above, a blade flutter or aerodynamic damping factor calculation can be performed on two blade passages for a range of nodal diameters using the Fourier-Transformation pitch-change method. The solution is obtained faster than for a full-wheel calculation. A significant further reduction in computational time can be achieved by performing the aerodynamic damping computation using the Harmonic Analysis (HA) (see Harmonic Analysis in the *CFX-Solver Theory Guide*) solution method, which avoids the need for solving in time over many vibration cycles in order to reach a steady-periodic state.

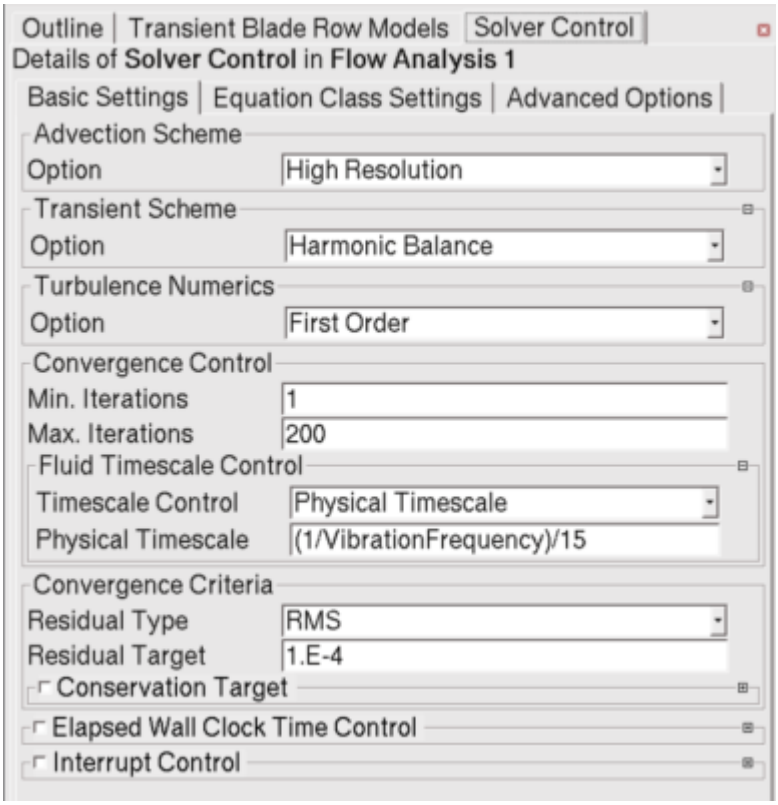
A blade flutter simulation that uses the HA methodology is set up like a blade flutter simulation that uses the time-marching method, except:

- In the **Transient Blade Row Models** details view, on the **Basic Settings** tab, you must set **Transient Method** > **Option** to `Harmonic Balance` and **Transient Method** > **Number of Modes** to an appropriate value.



  Typically, one mode is sufficient for a blade flutter calculation; however, for some transonic flow regimes, three modes may be required to accurately resolve the flow.

- In the **Solver Control** details view, on the **Basic Settings** tab, ensure that **Transient Scheme** > **Option** is set to `Harmonic Balance`, specify the maximum number of iterations (**Max. Iterations**), and set the pseudo-time step size (**Fluid Timescale Control** settings). Typically, the size of the pseudo-time step (**Physical Timescale**) is equal to the period of the vibration cycle divided by the intended number of pseudo-time steps per period.



The period of the vibration cycle is equal to 1/(vibration frequency). A typical number of pseudo-time steps per cycle is between 15 and 30. Recall that the lower the number of pseudo-time steps that can be used while maintaining a stable solution, the more efficient the calculation is with respect to the time marching method.

- The **Mesh Deformation** option cannot be set to `Periodic Regions of Motion`. For details on the `Periodic Regions of Motion` option, see Periodic Regions of Motion (p. 52).

6.1.8.4. Case 4: Harmonic Forced Response

The Harmonic Forced Response analysis requires two inputs from the CFD analysis. They are the forcing function due to the unsteady pressure, and the contributions to the aerodynamic damping matrix. In both cases, the information is required in the form of the first harmonic of the surface pressure field on the blade being analyzed. For details, see Harmonic Analysis in the *Theory Reference*.

The forcing function in the Harmonic Forced Response analysis is obtained from transient blade row interactions, while the contributions to the aerodynamic damping matrix is obtained from the blade flutter analysis. You can use either a full-wheel model or a reduced geometry using the Time-Transformation or Fourier-Transformation methods as a pitch-change model. These unsteady pressure values will be exported in the complex form obtained from their Fourier representation. In order for the solver to complete these calculations, you must specify the excitation frequency. The excitation frequency can be specified as an engine order or obtained from the blade vibration frequency. For details, see [Export Surface Name]: Excitation Frequency in the *CFX-Pre User's Guide*.

The engine order represents the number of disturbances (or pulses) per revolution. Therefore, the excitation frequency can be computed as $F=\frac{EO\times\Omega}{2\pi}$, where F is the excitation frequency, EO is the engine order, and Ω is the angular velocity of the machine.

For blade flutter cases that have the **Periodic Displacement** mesh motion option, the CFX-Solver writes the passage number to the field data of the exported data.

- If the periodic mesh displacement input vector components are specified using `Cartesian Components`, then the CFX-Solver will export a profile file that contains both the nodal diameter and mode multiplier, where the value of the mode multiplier is the same as `Scaling`.
- If the periodic mesh displacement input vector components are specified using `Normalised Cartesian Components`, then:
  - In a new case, the CFX-Solver will export a profile file that contains both the nodal diameter and mode multiplier, where mode multiplier is defined as `Amplitude/Maximum Displacement`.
  - In an existing case that was made from a release earlier than Release 18.0, the CFX-Solver will export a profile file that does not contain the mode multiplier. However, an existing case can be upgraded as follows:
    1. With the case loaded in CFX-Pre, edit the boundary that has periodic mesh displacement (so that its details view appears).
    2. If the existing case was made from a release earlier than Release 17.2, select the **Use Profile Data** option (and, if there is more than one profile, select the applicable profile).
    3. Click **Apply**.

       This updates the case in CFX-Pre by adding the required CCL.
    4. Save the case file (and/or write the solver input file).

Note that if the above procedure is not followed, the maximum displacement will not be in the solver input file, and the exported profile file will not contain the mode multiplier.

Note:

CFX can export profile data to the mesh coordinates used by Ansys Mechanical. For details, see Mapping Profile Data to a File in the *CFX-Pre User's Guide*.

Note:

When exporting complex pressures on a boundary with mesh motion (for example, a Fourier Transformation blade flutter case), the results will be exported using the initial mesh coordinates from the CFX-Solver run.

Note:

CFX and Mechanical use different representations for harmonic pressure loads, $P(t)$, where pressure P is a function of time t.

CFX represents harmonic pressure loads using real-number Fourier coefficients as:

$$P(t)=A_0+\sum_{k=1}^{N}\left(A_k\cos(k\omega t)+B_k\sin(k\omega t)\right) \tag{6.3}$$

where k represents the index of the mode, N represents the total number of modes, A_k and B_k represent the real-number Fourier coefficients, and ω represents the base frequency.

In contrast, Mechanical represents harmonic pressure loads using imaginary and real pressure components as:

$$P(t)=P_{1,0}+\sum_{k=1}^{N}\left(\left(P_{1,k}+P_{2,k}\right)e^{ik\omega t}\right) \tag{6.4}$$

where $P_{1,k}$ and $P_{2,k}$ represent the real and imaginary pressure components, respectively, of the contribution for mode k, and $P_{1,0}$ represents the real, time averaged, value.

By comparing Equation 6.3 (p. 286) and Equation 6.4 (p. 286), it can be seen that $P_{1,k}$ is analogous to A_k, while $P_{2,k}$ is analogous to $-B_k$.

When exporting to Mechanical, CFX exports $P_{1,k}$ and $P_{2,k}$.

For more details, see Harmonic Analysis in the *Theory Reference*.

6.1.8.5. Case 5: Transient Rotor Stator Multi-Stage Cases

Ansys CFX enables you to use Time-Transformation Transient Rotor Stator interfaces to model multistage turbomachinery configurations. In general, the Time Transformation method is used for

small-to-moderate pitch-change, multistage turbomachines. Note that the Time Transformation method cannot be used with incompressible cases. The Time Transformation method is not recommended when the rotation rate of the rotor is very low.

The multistage modeling capability of the Time Transformation method in Ansys CFX is demonstrated by the examples in the following sections.

6.1.8.5.1. Modeling a Single-pitch-ratio Multistage Turbomachine using the Time Transformation Method

6.1.8.5.2. Modeling 1.5 Stages with Two Different Pitch Ratios using the Time Transformation Method

6.1.8.5.3. Modeling a multistage turbomachine with Time Transformation TRS and other interfaces: Profile Transformation & Stage

6.1.8.5.4. Modeling a multistage turbomachine with Time Transformation TRS and single-sided Time Transformation interfaces (STT-TRS)

6.1.8.5.1. Modeling a Single-pitch-ratio Multistage Turbomachine using the Time Transformation Method

In this *special situation*, where every other row of the turbomachine has same number of blades, we have technically the configuration of a single disturbance being imposed in each passage. Therefore, there is only one pitch ratio present in the configuration. An example of such a case would be a machine in which all stator rows have the same number of blades and all rotors rows have the same number of blades, but the number of blades differs between the stator rows and the rotor rows. This special case can be modeled accurately by using TT correction on all the TRS interfaces (TT-TRS) between rows. Figure 6.8: 1.5 stage machine with equal number of blades in rows 1 & 3 but different than in row 2. Flow can be accurately modeled via a sequence of TT-corrected TRS interfaces. (p. 288) demonstrates an example of a 1.5 stage turbomachine with an equal number of blades in the first and third rows, but a different blade count in the middle row. TT-TRS interfaces have been applied sequentially to this configuration.

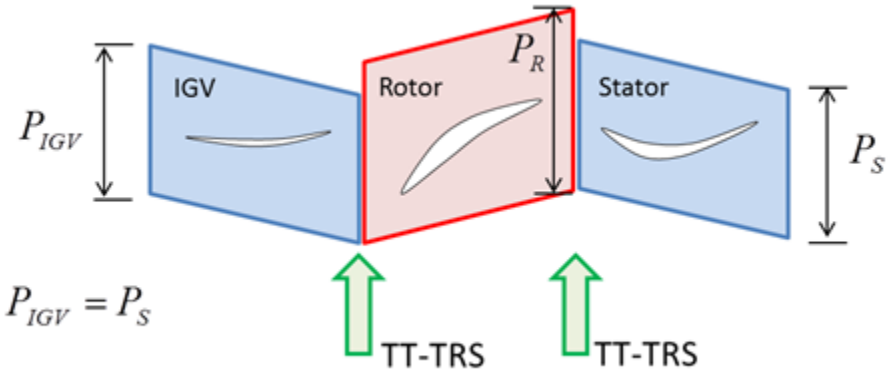
To set up this case:

1. Set the two rotor/stator interfaces (in the example of Figure 6.8: 1.5 stage machine with equal number of blades in rows 1 & 3 but different than in row 2. Flow can be accurately modeled via a sequence of TT-corrected TRS interfaces. (p. 288), the IGV/Rotor and the Rotor/Stator interfaces) as Transient Rotor Stator (TRS).

2. In the **Outline** tree view, edit the `Transient Blade Row Models` object under the applicable flow analysis.

3. Create two disturbances of the same type (TT) and assign each of them to the respective TRS interfaces created in step 1.

Figure 6.8: 1.5 stage machine with equal number of blades in rows 1 & 3 but different than in row 2. Flow can be accurately modeled via a sequence of TT-corrected TRS interfaces.

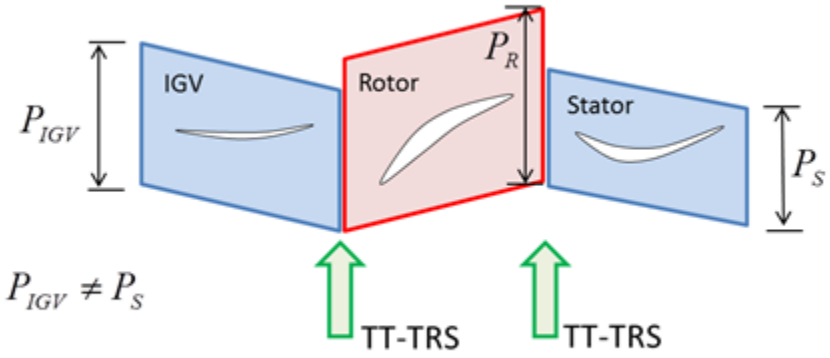


This modeling method will allow for capturing accurate blade passing signals in all the rows as well as obtaining accurate machine performance. The limitation on acceptable pitch ratio with use of the TT method still applies to this modeling technique.

6.1.8.5.2. Modeling 1.5 Stages with Two Different Pitch Ratios using the Time Transformation Method

In this configuration, each row has a different pitch value or two distinct pitch ratios across the machine, as shown in the following figure. The Time Transformation TRS interface can also be applied sequentially between the IGV and Rotor, and between the Rotor and Stator. This extension of the Time Transformation TRS interface allows for capturing both upstream and downstream blade passing frequencies in the rotor passage, making this modeling method useful for forced response analysis.

Figure 6.9: 1.5 stages with differing blade numbers between rows 1, 2 & 3. Flow can be modeled via a sequence of TT-TRS interfaces.



The following procedure is necessary to model a 1.5 stage turbomachine with two different pitch ratios:

1. Set all the rotor/stator interfaces to be of the TRS type.
2. In the **Outline** tree view, edit the `Transient Blade Row Models` object under the applicable flow analysis.
3. Create Time Transformation disturbances for each of the TRS interfaces that are corrected by Time Transformation.

This modeling strategy will allow for obtaining accurate machine performance as well as capturing blade passing signals in rows 1 and 2. On the other hand, row 3 may contain polluted frequency content due to modeling approximation. The limitation on acceptable pitch ratio with use of the Time Transformation method still applies to this modeling technique.

Important:

It is not recommended that you use Time Transformation TRS for every rotor-stator interface beyond a 1.5 stage configuration. For turbomachines with more than 1.5 stages, we recommend the modeling strategy shown in Modeling a multistage turbomachine with Time Transformation TRS and other interfaces: Profile Transformation & Stage (p. 289).

6.1.8.5.3. Modeling a multistage turbomachine with Time Transformation TRS and other interfaces: Profile Transformation & Stage

Accurate multistage performance can be predicted by using a combination of Time Transformation TRS and other interface models such as Profile Transformation TRS or Stage interface (mixing-plane). For maximum accuracy, it is recommended that you use Time Transformation TRS at each rotor passage inlet, and a Profile Transformation TRS or Stage interface at the rotor passage outlet. The idea is to minimize modeling error at the TRS interfaces where flow shocks might be crossing by using Time Transformation TRS modeling at those interfaces. The next two figures illustrate two common possible combinations of interfaces for modeling multistage configurations.

Figure 6.10: Multistage modeled with a combination of TT and PT interfaces. For high speed flows, the TT-TRS interfaces should be placed where shocks are crossing between the adjacent passages. This is typically happening at the interface between the exit of the stator row and the inlet of the rotor row.

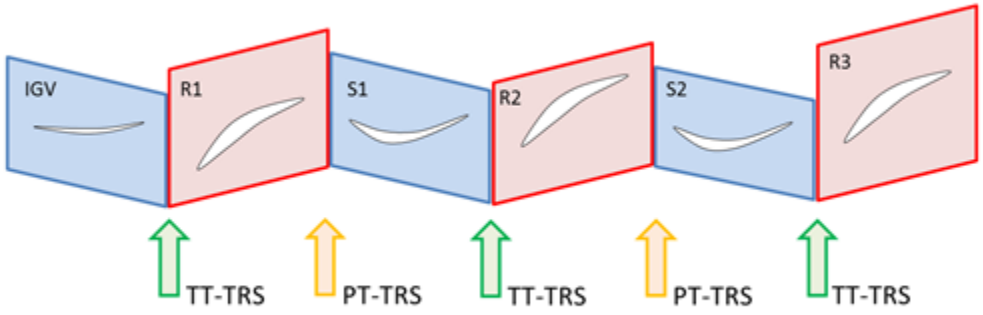
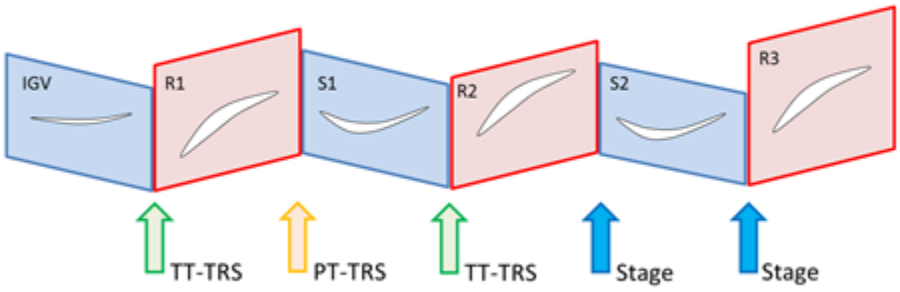


Figure 6.11: Multistage modeled with a combination of TT, PT and Stage interfaces



The following procedure is necessary to model multistage turbomachines with Time Transformation and Profile Transformation interfaces:

1. Set all the rotor/stator interfaces to be of the TRS type.
2. In the **Outline** tree view, edit the `Transient Blade Row Models` object under the applicable flow analysis.
3. Create Time Transformation disturbances for the TRS interfaces that are corrected by Time Transformation.

This modeling strategy will allow for obtaining accurate machine performance for turbines and compressors. However, this modeling strategy is not intended to be used for capturing accurate blade passing signals throughout the machine due to the modeling approximation of the Profile Transformation method. The limitation on acceptable pitch ratios using the Time Transformation method still applies to this modeling technique.

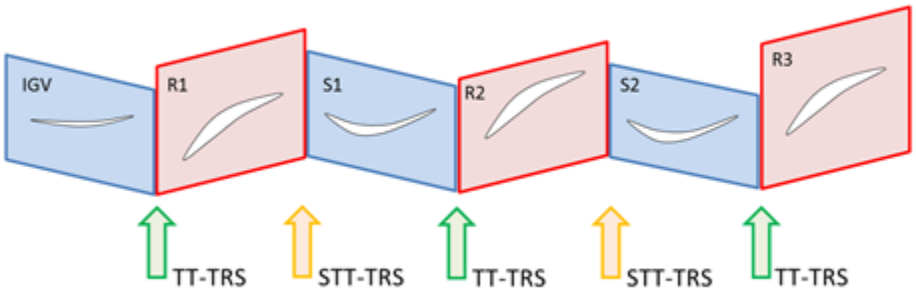
6.1.8.5.4. Modeling a multistage turbomachine with Time Transformation TRS and single-sided Time Transformation interfaces (STT-TRS)

For multistage machines, an alternative to using PT-TRS or TT-TRS models on both sides of TRS interfaces is to apply the so-called single-sided TT interface (STT-TRS). The STT-TRS interface model applies the PT model to one side of the interface (typically the upstream side) and the TT model to the other side (typically the downstream side).

Also, you can reverse this correction if necessary (for example, if you want to capture signals from the downstream component). The advantage of using STT-TRS over PT-TRS is that STT-TRS maintains the upstream blade wake thickness as it convects downstream. This modeling strategy results in more accurate accounting of machine performance and losses.

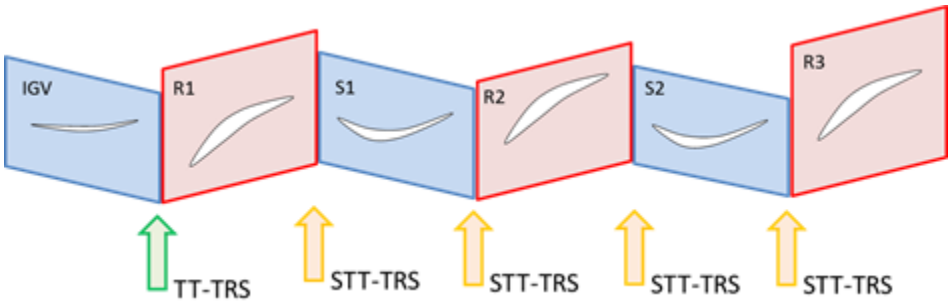
For maximum accuracy, it is recommended that you use TT-TRS at the rotor passage inlet and STT-TRS on the rotor passage exit. Figure 6.12: Multistage modeled with combination of TT-TRS and STT-TRS interfaces (p. 291) illustrates how the STT-TRS is used with TT-TRS in modeling a multistage machine.

Figure 6.12: Multistage modeled with combination of TT-TRS and STT-TRS interfaces



It is also possible to apply sequentially several STT interfaces, as shown in Figure 6.13: Multistage modeled with several STT-TRS interfaces in sequence (p. 291).

Figure 6.13: Multistage modeled with several STT-TRS interfaces in sequence



The following procedure is necessary to model multistage turbomachines with TT-TRS and STT-TRS interfaces:

1. Set all the rotor/stator interfaces to be of the TRS type.
2. In the **Outline** tree view, edit the `Transient Blade Row Models` object under the applicable **Flow Analysis** object.
3. Create TT disturbances for all of the TRS interfaces that are corrected by either TT or STT.
4. For the STT-TRS interfaces, the TT disturbances have to be modified on one of the domain interface sides. On the side of the interface not modeled by TT (the PT side as explained above), the side option should be changed to `None`. The setting for the other side can be left as it is; the TT model will be applied to that side. Note that the interface side name is displayed in the transient rotor stator domain interface's **Basic Settings** tab.

This modeling strategy can improve the accuracy of machine performance predictions for both turbines and compressors. However, this type of modeling is not intended for capturing accurate blade passing signals throughout the machine due to the modeling approximation of the STT-TRS method. The limitation on acceptable pitch ratio with use of the TT method still applies to this modeling technique.

6.1.8.6. Case 6: Transient Rotor Stator Cases with Asymmetric Flow

The Fourier Transformation Transient Rotor Stator (FT-TRS) interface can handle flow cases in which an impeller or rotor is attached to a full 360° domain, where the impeller or rotor may experience extremely large inflow variation, such as a once-per-revolution or three-per-revolution inflow disturbance.

Note:

An FT-TRS interface enables the modeling of just two impeller or rotor blades connected to the full 360° domain. There is little or no benefit in using the FT-TRS interface in this way for cases that involve a small number of impeller or rotor blades.

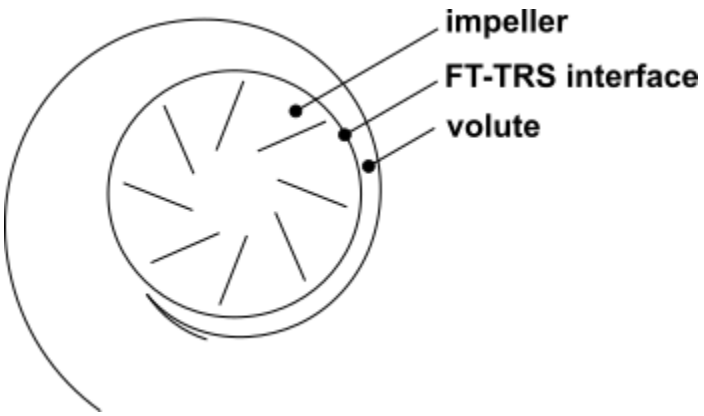
The following cases are discussed:

6.1.8.6.1. Modeling an Impeller in a Vaneless Volute

6.1.8.6.2. Modeling an Impeller in a Vaned Volute

6.1.8.6.1. Modeling an Impeller in a Vaneless Volute

In this case, the impeller experiences a long-period disturbance due to the asymmetry of the volute.



To set up an impeller in a vaneless volute:

1. Specify the FT-TRS interface as the interface between the volute and the blade passages.
2. Create a disturbance of type `Rotor Stator`:
   a. On the impeller side, select `Phase Corrected` and then select the phase-corrected and sampling interfaces (which are between impeller blades).
   b. On the volute side, select `None`.

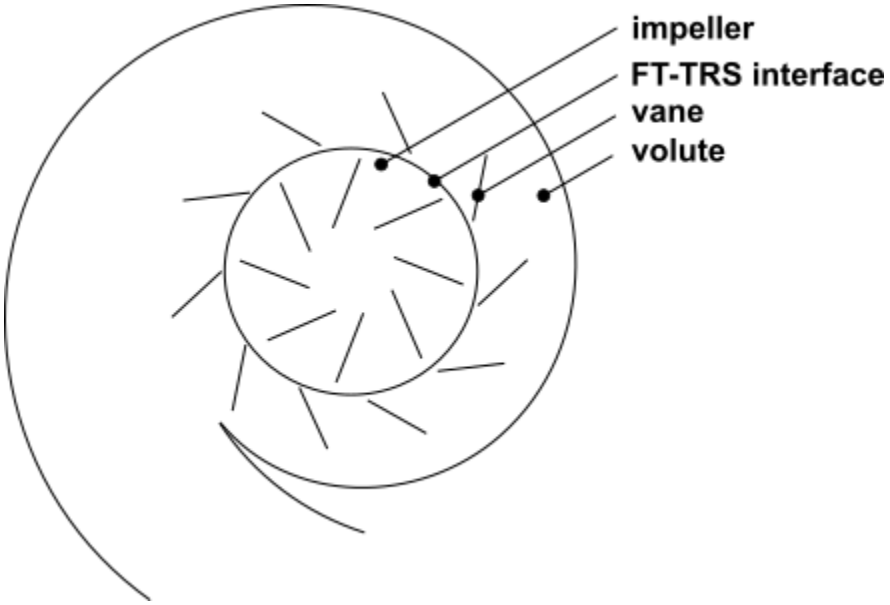
      The side of the FT-TRS interface that is attached to a 360° sector will have neither a sampling domain interface nor a phase corrected interface.

   The disturbance accounts for the signal from the volute.

Note that a similar procedure could be used to set up a case involving a fan in a crosswind.

6.1.8.6.2. Modeling an Impeller in a Vaned Volute

In this case, the impeller experiences two flow disturbances: one short-period disturbance due to the vanes, and one long-period disturbance due to the asymmetry of the volute.



To set up an impeller in a vaned volute:

1. Follow the steps for a vaneless volute.

   Because of the presence of the vanes, the disturbance is automatically configured to account for the signal from the vanes.

2. To account for the signal from the volute, create a (second) disturbance of type `Rotational Flow Boundary Disturbance`:

   a. On the impeller side, select `Phase Corrected` and then select the phase corrected and sampling interfaces (which are between impeller blades).

   b. Specify that the number of passages in 360° (under the external passage definition) is one, to capture the once-per-revolution disturbance of the volute.

   c. Set the **Signal Motion** option to `Stationary` to indicate that the signal comes from the volute side of the Rotor Stator interface.

6.2. Blade Film Cooling

Within gas turbine engines, the combustors direct hot exhaust gas over the blade rows of the turbine section. The temperatures are usually hotter than the melting point of these blades, so the blades need to be cooled directly to prevent them from being damaged. This is usually done by redirecting cooler air from the compressor to the inside of each blade, and then out through an array of small holes to create a film of cool air over the blade surface.

The following topics are discussed:

6.2.1. Options for Modeling Blade Film Cooling

6.2.1. Options for Modeling Blade Film Cooling

There are two main approaches to modeling blade film cooling in CFX:

- Use a full resolution mesh

  You can create a detailed mesh model that resolves the flow path from the plenum interior to the blade and out through individual holes in the blade surface. This approach is accurate, but can be computationally expensive in terms of setup and solution.

- Use injection regions

  You can create a coarser mesh that does not fully resolve the individual holes directly, then use one or more injection regions (see Injection Regions in the *CFX-Pre User's Guide*) to model the injection of cooling air from hole positions on the blade surfaces. This approach is computationally less expensive and allows the holes to be modified independently of the aerodynamic surfaces, facilitating rapid blade design optimization. As the optimization procedure continues, you can refine the meshes to better resolve each hole and its effects. Finally, a full resolution model can be created from the best design in order to help confirm its performance.

Chapter 7: Multiphase Flow Modeling

Multiphase flow refers to the situation where more than one fluid is present. Each fluid may possess its own flow field, or all fluids may share a common flow field. Unlike multicomponent flow, the fluids are not mixed on a microscopic scale in multiphase flow. Rather, they are mixed on a macroscopic scale, with a discernible interface between the fluids. Ansys CFX includes a variety of multiphase models to enable the simulation of multiple fluid streams, bubbles, droplets, solid particles, and free surface flows.

Two distinct multiphase flow models are available in Ansys CFX, an `Eulerian-Eulerian` multiphase model and a `Lagrangian Particle Tracking` multiphase model. Information on the use of the `Lagrangian Particle Tracking` model is available in Particle Transport Modeling (p. 353).

This chapter describes how `Eulerian-Eulerian` multiphase flow is modeled and also includes modeling advice. The theory of multiphase flow is available; see Multiphase Flow Theory in the *CFX-Solver Theory Guide*.

The corresponding information for single-phase, single-component flow is available:

- For information on the models used for single-phase flow, see Basic Capabilities Modeling (p. 39).
- For the mathematical implementation of the single-phase flow, see Basic Solver Capability Theory in the *CFX-Solver Theory Guide*.
- For advice on modeling, see Advice on Flow Modeling (p. 557).

Important:

Because this chapter extends the information given in the single-phase documentation, you are advised to read the relevant sections in the single-phase documentation before reading this chapter.

This chapter describes:

7.1. Multiphase Terminology
7.2. Multiphase Examples
7.3. Eulerian-Eulerian Multiphase Versus Particle Transport
7.4. Specifying Fluids for Multiphase Flow
7.5. The Homogeneous and Inhomogeneous Models
7.6. Buoyancy in Multiphase Flow
7.7. Multicomponent Multiphase Flow
7.8. Interphase Momentum Transfer Models
7.9. Solid Particle Collision Models
7.10. Interphase Heat Transfer

7.11. Polydispersed, Multiple Size Group (MUSIG) Model
7.12. Turbulence Modeling in Multiphase Flow
7.13. Additional Variables in Multiphase Flow
7.14. Sources in Multiphase Flow
7.15. Interphase Mass Transfer
7.16. Boundary Conditions in Multiphase Flow
7.17. Modeling Advice for Multiphase Flow
7.18. Free Surface Flow
7.19. Algebraic Slip Model (ASM)
7.20. Multiphase Flow Restrictions

7.1. Multiphase Terminology

The following terms are used in Ansys CFX multiphase flow:

- Multiphase Flow (p. 296)
- Eulerian-Eulerian (p. 297)
- Inhomogeneous Multiphase Flow (p. 297)
- Homogeneous Multiphase Flow (p. 297)
- Multicomponent Multiphase Flow (p. 297)
- Volume Fraction (p. 297)
- Free Surface Flow (p. 298)
- Surface Tension (p. 298)

7.1.1. Multiphase Flow

Multiphase flow is a flow in which more than one fluid is present. In general, the fluids consist of different chemical species, such as air-water. In some applications, they may represent different thermodynamic phases of the same species, such as steam-water.

It is important to distinguish between multicomponent and multiphase flow. A multicomponent fluid is assumed to consist of a mixture of chemical species that are mixed at the molecular level. In this case, single mean velocity and temperature fields, and so on, are solved for the fluid. Examples are gaseous mixtures, and solutes in liquids.

The fluids in a multiphase flow are assumed to be mixed at macroscopic length scales, much larger than molecular. Examples are gas bubbles in a liquid, liquid droplets in a gas or in another immiscible liquid, and so on. In this case, it is necessary to solve for different characteristics such as velocity and temperature fields for each fluid. These may interact with each other by means of interfacial forces and heat and mass transfer across the phase interfaces.

For example, if cold wet particles are injected into a fast flowing stream of hot air, the particles will be accelerated by drag, they will be heated up by heat transfer across the phase boundary, and they will be dried by evaporation of water into water vapor at the phase boundary.

7.1.2. Eulerian-Eulerian

The `Eulerian-Eulerian` model is one of the two main multiphase models that has been implemented in Ansys CFX; the other is the Lagrangian Particle Tracking Model. For details, see Particle Transport Modeling (p. 353).

Within the `Eulerian-Eulerian` model, certain interphase transfer terms used in momentum, heat, and other interphase transfer models, can be modeled using either the `Particle Model`, the `Mixture Model` or the `Free Surface Model` In particular, the calculation of the interfacial area density, used for all inhomogeneous transfer models for a given fluid pair, is calculated according to one of these models. The model used for a particular pair of phases is set on the **Fluid Pairs** tab. The available options depend on the morphology of each phase of the pair (for example, continuous, dispersed, and so on), and on settings found on the **Fluid Models** tab (homogeneous options, free surface model option).

7.1.3. Inhomogeneous Multiphase Flow

Inhomogeneous multiphase flow refers to the case where separate velocity fields and other relevant fields exist for each fluid. The pressure field is shared by all fluids. The fluids interact via interphase transfer terms. The `Particle` and `Mixture Models` are both inhomogeneous multiphase models.

7.1.4. Homogeneous Multiphase Flow

Homogeneous multiphase flow is a limiting case of `Eulerian-Eulerian` multiphase flow where all fluids share the same velocity fields, as well as other relevant fields such as temperature, turbulence, and so on. The pressure field is also shared by all fluids.

7.1.5. Multicomponent Multiphase Flow

It is possible to combine the notions of multicomponent and multiphase flows. In this case, more than one fluid is present, and each such fluid may be a mixture of chemical species mixed at molecular length scales. An example is air bubbles in water in which ozone gas is dissolved in both the gaseous and liquid phases. In this case, mass transfer of common species may occur by diffusion across the phase interface.

7.1.6. Volume Fraction

Multiphase modeling employs the notion of interpenetrating continua. Although phases are mixed at length scales much larger than molecular, they are also assumed to be mixed at length scales smaller than you want to resolve.

Thus, each phase is assumed to be present in principle in each control volume, and assigned a volume fraction equal to the fraction of the control volume occupied by that phase. This variable appears as `<fluid>.Volume Fraction` in the results file. The variable `<fluid>.Conservative Volume Fraction` is also available but should not usually be used for postprocessing. The conservative

volume fraction is used by the CFX-Solver; it may exceed 1 and may not sum to unity over all fluids during the solution. However, these criteria are enforced in a converged solution.

7.1.7. Free Surface Flow

Free Surface flow refers to a multiphase situation where the fluids (commonly water and air) are separated by a distinct resolvable interface.

7.1.8. Surface Tension

Surface tension is a force that exists at a free surface interface which acts to minimize the surface area of the interface. It gives rise to effects such as a pressure discontinuity at the interface and capillary effects at adhesive walls.

7.2. Multiphase Examples

The following are examples of multiphase flow:

- Water Droplets in Air (p. 298)
- Air Bubbles in Water (p. 298)
- Gas-Solid and Liquid-Solid Flow (p. 298)
- Three-Phase Flow (p. 299)
- Polydispersed Flow (p. 299)

7.2.1. Water Droplets in Air

Water droplets in air constitute two different fluids that are mixed at the macroscopic level and not the microscopic level. Hence, you need to use a multiphase model and define two distinct fluids, `Water` and `Air`. Each fluid contains only one component consisting of one material (air or water), whose properties can be defined. `Air` is the continuous fluid (Continuous Fluid (p. 301)), and `Water` is the dispersed fluid (Dispersed Fluid (p. 301)).

7.2.2. Air Bubbles in Water

As above, you need to use a multiphase model and define two distinct fluids: `Water` and `Air`. In this case, `Air` is the dispersed fluid (Dispersed Fluid (p. 301)), and `Water` is the continuous fluid (Continuous Fluid (p. 301)).

7.2.3. Gas-Solid and Liquid-Solid Flow

It is possible to model the motion of a large number of solid particles in a gas or a liquid as an `Eulerian-Eulerian` two phase flow. Examples occur in pneumatic conveying, sedimentation in rivers, and fluidized beds.

For example, the two phases may be `Water`, the continuous fluid (Continuous Fluid (p. 301)), and `Sand`, the dispersed fluid (Dispersed Fluid (p. 301)).

In such problems, you should assign the solid phase a small insignificant molecular viscosity. This is permissible, as the physics are dominated by interphase drag and turbulence effects. The solid phase should be assigned free slip boundary conditions at walls. If viscosity is not set, the solid material will not be available for selection in the **Fluids and Particle Definitions...** list on the **Basic Settings** tab of the Domains details view in CFX-Pre.

7.2.4. Three-Phase Flow

It is possible to have more than one dispersed phase in a continuous phase. For example, certain regimes of water-oil-gas flow in an oil pipeline may involve both oil droplets and gas bubbles immersed in a continuous water phase.

7.2.5. Polydispersed Flow

The above dispersed flow examples assume a single mean particle diameter for the dispersed phases. **Poly-dispersed** flows involve dispersed phases of different mean diameters.

One way to model such flows is to define a different phase for each particle size, as the particle size has a strong influence on interphase transfer. For example, to model the flow of air bubbles in water of different diameters, you need to create three fluids, such as `Air1`, `Air2`, `Air3`, each with the same material properties as air. Then, set the number of fluids equal to four, and select `Water`, `Air1`, `Air2` and `Air3` as the four fluids. Next, under **Fluid Models**, designate `Water` as the continuous phase, and the other three fluids as dispersed phases of desired mean diameters.

Information on an alternative way to model polydispersed flows is provided by the MUSIG model. For details, see Polydispersed, Multiple Size Group (MUSIG) Model (p. 315).

7.3. Eulerian-Eulerian Multiphase Versus Particle Transport

A multiphase flow containing dispersed particles may be modeled using either the particle transport (`Lagrangian Particle Tracking`) model or the `Eulerian-Eulerian` multiphase model. Some advantages and disadvantages of the `Eulerian-Eulerian` multiphase model are given below to aid in the choice of model. The equivalent section for the particle transport model is available in Particle Transport Versus Eulerian-Eulerian Multiphase (p. 354).

Table 7.1: Eulerian-Eulerian Multiphase vs. Particle Transport

Advantages of Eulerian-Eulerian Multiphase	Disadvantages of Eulerian-Eulerian Multiphase
Complete global information for the particle phase is available	Expensive if many sets of equations are used; that is, if there are many particle sizes. However, the homogeneous MUSIG model is an `Eulerian-Eulerian` model that uses a single velocity field for multiple size groups.
Applicable for wide range of volume fractions	Knowledge of the diffusion coefficients is incomplete

Advantages of Eulerian-Eulerian Multiphase	Disadvantages of Eulerian-Eulerian Multiphase
Relatively cheap for one additional set of equations	Difficult to get accuracy over a range of particle sizes for combustive flows
Turbulence is included automatically	When there is phase change, the particle diameter must be user-specified rather than calculated automatically by the model. This can decrease accuracy. (The droplet condensation model is an exception.)

7.4. Specifying Fluids for Multiphase Flow

You should first use the **Material** details view in CFX-Pre to define the properties of the materials in your simulation. You can then select these materials, in addition to predefined library materials, on the **Basic Settings** tab of the **Domain** details view. For details, see Basic Settings Tab in the *CFX-Pre User's Guide*. Defining more than one fluid implies a multiphase simulation.

Models that apply to all fluids in the simulation are set on the **Fluid Models** tab. For details, see Fluid Models Tab in the *CFX-Pre User's Guide*. Fluid specific attributes are defined on the **Fluid Specific Models** tab in CFX-Pre. For details, see Fluid Specific Models Tab in the *CFX-Pre User's Guide*. Interactions between the fluids are defined on the **Fluid Pairs** tab. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*.

Multiphase flows enable low speed compressibility effects to be modeled and you can mix compressible and incompressible fluids in one simulation. An example would be modeling an air-water system in which the water can be considered incompressible but the air is modeled as compressible due to the hydrostatic water pressure.

You should be aware of significant increases in memory and CPU time when using multiple fluids; these increase rapidly as more fluids are added. For details, see CPU and Memory Requirements in the *CFX-Solver Manager User's Guide*.

7.4.1. Morphology

Morphology is used to describe the connectivity or distribution of the fluid - whether it forms a single continuous medium, or whether it is present, for example, in small droplets that are not connected.

You can select from:

7.4.1.1. Continuous Fluid

7.4.1.2. Dispersed Fluid

7.4.1.3. Dispersed Solid

7.4.1.4. Particle Transport Fluid

7.4.1.5. Particle Transport Solid

7.4.1.6. Polydispersed Fluid

7.4.1.7. Droplets (Phase Change)

7.4.1.1. Continuous Fluid

A continuous phase or continuous fluid is one which forms a continuous connected region. An example is air, when modeling rain drops in air.

7.4.1.2. Dispersed Fluid

A dispersed fluid is a fluid that is present in discrete regions that are not connected. Examples are water droplets in air or gas bubbles in a liquid.

7.4.1.3. Dispersed Solid

A dispersed solid is one that is present in discrete regions that are not connected, for example, solid particulates.

You must select the morphology of all fluids as continuous if you want to use the `Mixture` model on the **Multiphase Options** tab. If dispersed fluids or solids are included, then the `Particle` model will be the only valid option.

Examples of multiphase flow with different morphologies are available. For details, see Multiphase Examples (p. 298).

7.4.1.4. Particle Transport Fluid

This is used for fluids when using the Particle Transport Model. For details, see Particle Morphology Options (p. 356).

7.4.1.5. Particle Transport Solid

This is used for solids when using the Particle Transport Model. For details, see Particle Morphology Options (p. 356).

7.4.1.6. Polydispersed Fluid

A polydispersed fluid is a fluid that is present in discrete regions that are not connected and with varying sizes of the discrete regions. Typically, this morphology is used for modeling air bubbles of varying sizes in a liquid.

7.4.1.7. Droplets (Phase Change)

Droplets (Phase Change) are required if you want to use the Droplet Condensation Model. For details, see Droplet Condensation Model (p. 339).

7.4.2. Mean Diameter

For a `Dispersed Solid` or `Dispersed Fluid`, you must supply the mean diameter of the dispersed particles, droplets or bubbles constituting the phase. The mean diameter is the actual diameter for spherical particles. For non-spherical particles, you should provide the Sauter mean diameter, that is, the diameter of a sphere possessing the same volume as the particles.

7.4.3. Minimum Volume Fraction

For each phase, you can optionally set a minimum volume fraction. A minimum value is used for numerical robustness purposes, as the volume fractions are not allowed to go to machine zero at any location. A default value of 1.0E-15 is used if you do not set a value; this is almost always appropriate. If this causes numerical problems, then set the value to be several orders of magnitude less than the minimum expected physical value, but much larger than machine zero.

7.4.4. Maximum Packing

For details, see Maximum Packing (p. 312).

7.5. The Homogeneous and Inhomogeneous Models

Two distinct models are available for `Eulerian-Eulerian` multiphase flow: the homogeneous model and the inter-fluid transfer or inhomogeneous model.

7.5.1. The Inhomogeneous (Interfluid Transfer) Model

Each fluid possesses its own flow field and the fluids interact via interphase transfer terms. In the inhomogeneous multiphase model, there is one solution field for each separate phase. Transported quantities interact via interphase transfer terms. For example, two phases may have separate velocity and temperature fields, but there will be a tendency for these to come to equilibrium through interphase drag and heat transfer terms.

Three different sub-models that differ in the way they model the interfacial area density and the interphase transfer terms are available. A model is selected for each fluid pair on the **Fluid Pairs** tab. The available options depend on the morphology of each phase of the pair (for example, continuous, dispersed, and so on), and on settings found on the **Fluid Models** tab (homogeneous options, free surface model option).

Descriptions of the three models follow.

7.5.1.1. The Particle Model

This model is available when one of the phases is continuous and the other is dispersed or polydispersed. The dispersed phase particles or droplets are assumed to be spherical. It is suitable for modeling simple dispersed multiphase flow problems, for example, the dispersion of:

- Gas bubbles in a liquid.
- Liquid droplets in a gas or in immiscible liquid.
- Solid particles in a gas or in a liquid.

The MUSIG model is a variation of the particle model specialized for polydispersed phases. For details, see Polydispersed, Multiple Size Group (MUSIG) Model (p. 315).

7.5.1.2. The Mixture Model

This is a very simple model that treats both phases symmetrically. It may be appropriate as a first approximation or as the basis of user supplied interfacial transfer models for a calculation of non-disperse liquid-liquid or gas-liquid two phase flow. It requires both phases to be continuous. It can be used to model more complex multiphase flow problems, for example, Churn flow.

An interfacial length scale must be specified on the tab in CFX-Pre to relate the phasic volume fractions to the interfacial contact area. For details, see:

- Fluid Pair Models Tab in the *CFX-Pre User's Guide*.
- The Mixture Model in the *CFX-Solver Theory Guide*.

7.5.1.3. The Free Surface Model

This model is applicable to free surface flows. The particle model or mixture model may also be used for these flows if there is entrainment of one phase inside another.

7.5.2. The Homogeneous Model

The homogeneous model can be viewed as a limiting case of `Eulerian-Eulerian` multiphase flow in which the interphase transfer rate is very large. This results in all fluids sharing a common flow field, as well as other relevant fields such as turbulence. It is valid in the following circumstances:

- In a flow under gravity, where the phases have completely stratified, for example, a free surface flow where the interface is well defined. In this case, the volume fractions of the phases are equal to one or zero everywhere except at the phase boundaries, and it makes sense to use a single velocity field.
- If the flow is drag dominated, that is, α is very large, and there are no body forces, the phase velocities will tend to equalize over very short spatial length scales. This can occur in dispersed flows of extremely small particles.
- The approximation does *not in general* apply to drag dominated multiphase flow under gravity that is not stratified, for example, droplets falling under gravity in a gas. In this case, the droplets will quickly attain a fixed slip velocity relative to the continuous phase, where the interphase drag balances the differences in body forces. In this case, the homogeneous model should only be used if the resulting slip velocities are very small relative to the mean flow.

Free surface flow is a common application of the homogeneous model. The section provides additional details for this specific application. For details, see Free Surface Flow (p. 346).

To choose the homogeneous model for the momentum equation, select `Homogeneous Model` under **Multiphase Options**. Once this is done, the bulk momentum equation is solved to obtain the shared velocity field. The use of inhomogeneous turbulence models is also disabled.

When the homogeneous model (under **Multiphase Options**) is selected, a model for interphase momentum transfer is no longer needed, although the **Interphase Transfer Model** setting (on the **Fluid Pairs** tab) may still need to be specified due to other processes that require interphase transfer modeling. Some examples are:

- The heat transfer model may be inhomogeneous. This is required when applying the `Thermal Phase Change` model for interphase mass transfer. For details, see Thermal Phase Change Model (p. 324).
- Other mass transfer models (other than cavitation) also require use of the interfacial area density.
- `Interphase Additional Variable Transfer` models. For details, see Additional Variable Interphase Transfer Models (p. 321).

In these situations, the **Interphase Transfer Model** setting for fluid pairs (on the **Fluid Pairs** tab, for a particular pair of fluids) must be set to `Particle Model`, `Mixture Model` or `Free Surface Model`. Additional information on these models is available; for details, see:

- The Inhomogeneous (Interfluid Transfer) Model (p. 302)
- Interfacial Area Density in the *CFX-Solver Theory Guide*.

7.6. Buoyancy in Multiphase Flow

You should be familiar with buoyancy in single phase flows. For details, see Buoyancy (p. 58).

In addition to the buoyancy forces that can exist in single phase flows, the difference in density between phases produces a buoyancy force in multiphase flows (including particle tracking). For this reason, buoyancy is almost always important in multiphase flows.

For information on modeling low-speed compressible multiphase flow, see Buoyancy and Pressure (p. 60).

For multiphase flows, it can be important to correctly set the buoyancy reference density. For a flow containing a continuous phase and a dilute dispersed phase, you should set the buoyancy reference density to that of the continuous phase. This is because the pressure gradient is nearly hydrostatic, so the reference density of the continuous phase cancels out buoyancy and pressure gradients in the momentum equation.

This is much the same as in single phase, where any non-buoyant calculation implicitly assumes a buoyancy reference density of the fluid, thereby avoiding potential roundoff problems. For non-dilute cases (which include all free surface cases), all terms can be equally important for each fluid, so roundoff errors will be introduced for one of the fluids if there is a significant difference in density. You should choose the density of the lighter fluid because this gives an intuitive interpretation of pressure (that is, constant in the light fluid and hydrostatic in the heavier fluid). This simplifies pressure initial conditions, pressure boundary conditions and force calculations in postprocessing.

7.6.1. Fluid Buoyancy Model

7.6.1.1. Density Difference

This is the default option for all multiphase flows. The additional buoyancy force is modeled by considering the difference in density between phases. If you are using a constant density fluid, you should consider the Boussinesq model.

$$\boldsymbol{F}_{\alpha}=\left(\rho_{\alpha}-\rho_{\mathrm{ref}}\right)\boldsymbol{g} \tag{7.1}$$

7.6.1.2. Boussinesq

This option may be selected for constant density fluids for multiphase flows with heat transfer, but it is not the default option. You should use this model to model natural convection effects within such a fluid (that is, if density fluctuations due to temperature changes are important). Note that the default option, `Density Difference`, neglects such effects if the fluid density is constant.

$$\boldsymbol{F}_{\alpha}=\left(\rho_{\alpha}\left(1-\beta_{\alpha}\left(T_{\alpha}-T_{\mathrm{ref}}\right)\right)-\rho_{\mathrm{ref}}\right)\boldsymbol{g} \tag{7.2}$$

Note further that the `Boussinesq` option may be more difficult to converge than the `Density Difference` option, because of coupling between heat transfer and momentum. It may be advantageous to obtain a converged solution first using the `Density Difference` option. This may then be used as an initial guess for a solution using the `Boussinesq` option.

Multiphase simulations use the full buoyancy model because there is usually a significant difference in density between the phases. However, you can select to use the Boussinesq buoyancy model within each phase to account for buoyancy effects due to temperature differences within the phase.

7.7. Multicomponent Multiphase Flow

Ansys CFX allows the modeling of multiphase multicomponent flows. Information on multicomponent, single-phase flows is available; for details, see Multicomponent Flow (p. 64). One component in each phase must be calculated using a constraint component in the same way as for single-phase multicomponent flow. The limitations that currently apply to single-phase multicomponent flow also apply to multiphase multicomponent flow. You should note that the molecular diffusion coefficients are calculated using the **Kinematic Diffusivity** set on the **Fluid-Specific Details** tab for a domain in CFX-Pre. For details, see Component Details in the *CFX-Pre User's Guide*.

Source terms in multicomponent multiphase flow behave in the same way as multicomponent mass sources, but on a per fluid basis. For details, see Sources in Multiphase Flow (p. 322).

7.8. Interphase Momentum Transfer Models

The models described in this section only apply to inhomogeneous multiphase flow. When using the homogeneous model, momentum transfer between phases is assumed to be very large.

7.8.1. Interphase Drag

For low Mach number flows, the drag exerted on an immersed body by a moving fluid arises from two mechanisms only. The first is due to the viscous surface shear stress, and is called skin friction. The second is due to the pressure distribution around the body, and is called the form drag. The total drag force is most conveniently expressed in terms of the dimensionless drag coefficient, C_D. The calculation of C_D is available in Interphase Drag in the *CFX-Solver Theory Guide*.

7.8.1.1. Interphase Drag for the Particle Model

For a particle of simple shape, immersed in a Newtonian fluid and which is not rotating relative to the surrounding free stream, the drag coefficient, C_D, depends only on the particle Reynolds number. The function $C_D\left(\mathrm{Re}_{\alpha}\right)$ may be determined experimentally, and is known as the drag curve.

Ansys CFX offers several different models for the drag curve, and also allows you to specify the drag coefficients directly.

This section describes drag correlations specific to dispersed multiphase flow.

7.8.1.1.1. Specifying a Drag Coefficient

You can choose to specify the dimensionless **Drag Coefficients** C_D directly. This is done by selecting to use the `Drag Coefficient` option on the **Fluid Pairs** tab in CFX-Pre, and entering the appropriate **Drag Coefficient**. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*.

You can supply your own drag coefficient correlation as an expression and you are free to define your own interfacial Reynolds number, or any other dimensionless group.

7.8.1.1.2. Sparsely Distributed Solid Particles

7.8.1.1.2.1. Sparsely Distributed Solid Particles: Schiller Naumann Drag Model

This should only be used for solid spherical particles, or for fluid particles that are sufficiently small that they may be considered spherical. For non-spherical particles, you should supply the drag curve from experiment.

As the Schiller Naumann correlation is derived for flow past a single spherical particle, it is only valid in the dilute limit of very small solid phase volume fractions.

You can select this drag curve by selecting to use the `Schiller Naumann Drag Model` on the **Fluid Pairs** tab in CFX-Pre. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*.

7.8.1.1.3. Densely Distributed Solid Particles

7.8.1.1.3.1. Densely Distributed Solid Particles: Wen Yu Drag Model

The Wen Yu correlation is valid for solid phase volume fractions at least up to 0.2, and probably higher.

You can select this Drag Curve by selecting to use the Wen Yu Drag Model on the **Fluid Pairs** tab in CFX-Pre (Fluid Pair Models Tab in the *CFX-Pre User's Guide*). Theoretical information about the Wen Yu drag model is given at Densely Distributed Solid Particles: Wen Yu Drag Model in the *CFX-Solver Theory Guide*.

Note:

Although the Wen Yu drag law implemented in Ansys CFX follows the implementation by Gidaspow [18] and its subsequent wide use, this implementation of the drag law is, in fact, quite different from that given in the original Wen and Yu paper [181].

7.8.1.1.3.2. Densely Distributed Solid Particles: Gidaspow Drag Model

For very dense gas-solid or liquid-solid flows, such as occur in fluidized bed applications, the Gidaspow correlation is recommended.

You can select this drag curve by selecting to use the `Gidaspow Drag Model` on the **Fluid Pairs** tab in CFX-Pre (Fluid Pair Models Tab in the *CFX-Pre User's Guide*).

Theoretical information about the Gidaspow drag model is given at Densely Distributed Solid Particles: Gidaspow Drag Model in the *CFX-Solver Theory Guide*.

7.8.1.1.4. Sparsely Distributed Fluid Particles (drops and bubbles)

7.8.1.1.4.1. Sparsely Distributed Fluid Particles: Ishii-Zuber Drag Model

This is applicable to general fluid particles (drops and bubbles), for any pair of phases. Ansys CFX automatically takes into account the spherical particle and spherical cap limits and dense fluid particle effects.

You can select this Drag Curve by selecting to use the `Ishii-Zuber Model` on the **Fluid Pairs** tab in CFX-Pre. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*. This model is only available when a buoyant flow is specified and a **Surface Tension Coefficient** has been set.

7.8.1.1.4.2. Sparsely Distributed Fluid Particles: Grace Drag Model

This model was developed using air-water data and produces better results for air-water systems. Ansys CFX automatically takes into account the spherical particle and spherical cap limits. You can set a **Volume Fraction Correction Exponent** for this model for use with high bubble volume fractions, see below for details.

You can select this Drag Curve by selecting to use the `Grace Model` on the **Fluid Pairs** tab in CFX-Pre. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*. This model is only available when a buoyant flow is specified and a `Surface Tension Coefficient` has been set.

7.8.1.1.4.3. Sparsely Distributed Fluid Particles: Availability

Both the `Ishii Zuber` and `Grace Drag Models` make explicit use of the gravity vector and surface tension coefficient. Hence, both are only available for buoyant multiphase flows when a surface tension coefficient has been specified. The fluid morphologies must be `Continuous` and `Dispersed Fluid` respectively.

7.8.1.1.5. Densely Distributed Fluid Particles

7.8.1.1.5.1. Densely Distributed Fluid Particles: Ishii-Zuber Drag Model

The Ishii Zuber drag model automatically takes into account compact particle effects and is therefore suitable for modeling flows containing high fluid particle volume fractions.

A user-defined maximum packing value can be set. This is defaulted to unity for a dispersed fluid phase. For details, see Maximum Packing (p. 312).

You can select this drag curve by selecting to use the `Ishii Zuber Model` on the **Fluid Pairs** tab in CFX-Pre. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*. This model is available only when a buoyant flow is specified and a surface tension coefficient has been set.

7.8.1.1.5.2. Densely Distributed Fluid Particles: Grace Drag Model

The Grace drag model is formulated for flow past a single bubble. For details, see Sparsely Distributed Fluid Particles (drops and bubbles) (p. 307).

You can select this drag curve by selecting to use the Grace model on the **Fluid Pairs** tab in CFX-Pre. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*. This model is only available when a buoyant flow is specified and a **Surface Tension Coefficient** has been set.

In the dilute limit, leave the **Volume Fraction Correction Exponent** at its default value of zero. For non-dilute bubble volume fractions, set a non-zero value, depending on the bubble size as discussed below.

Small bubbles tend to rise more slowly at high void fraction, due to an increase in the effective mixture viscosity. To capture this effect, a negative **Volume Fraction Correction Exponent** should be used. The Ishii Zuber correlation uses an exponent of -1 in this limit. A value of -0.5 has also been used successfully by some investigators.

Large bubbles, on the other hand, tend to rise faster at high void fractions, because they are dragged along by the wakes of other bubbles. This effect may be modeled using a positive **Volume Fraction Correction Exponent**. The Ishii Zuber correlation uses an exponent of 2 in this regime. A value of 4 has been used successfully by some investigators [74]. If you use a value of 4 and experience poor convergence for the mass equations, try reducing the value to 2.

7.8.1.2. Interphase Drag for the Mixture Model

The mixture model is primarily a tool to permit advanced users to specify their own interfacial transfer models for complex situations. Hence the only drag model available is the specified Drag Coefficient model. This may be a constant, or a function defined using expression language or User Fortran.

7.8.2. Lift Force

The lift force refers to the shear-induced lift force acting on a dispersed phase in the presence of a rotational continuous phase. Hence, it is applicable to the `Particle Model` only. Additional information on the theory is available.

You may set a non-dimensional lift coefficient C_L either as a constant, or an expression. It should be set to 0.5 for inviscid flow around a sphere. For viscous flow, the coefficient varies from 0.01 to 0.5 in a way that is only partially understood.

Several models for the lift coefficient have been proposed in the literature. See, for example, [86] and [87], and this is still a matter for current research. Ansys CFX has the following built-in lift models:

- The Saffman Mei Lift Force Model in the *CFX-Solver Theory Guide*
- The Legendre and Magnaudet Lift Force Model in the *CFX-Solver Theory Guide*
- The Tomiyama Lift Force Model in the *CFX-Solver Theory Guide*

The lift force is proportional to the continuous phase density. Hence, it is mainly significant when the dispersed phase density is either less than, or of the same order of magnitude as the continuous phase density. Also, it is proportional to the continuous phase shear rate. Hence, it is most significant in shear layers whose width is comparable to the dispersed phase mean diameter.

For example, the lift force is important for bubbly flow in a vertical pipe, when the pipe diameter is comparable to the bubble diameter. In this case, the lift force induced by the continuous phase boundary layer is responsible for pushing the bubbles towards the wall. On the other hand, for bubbly downflow, the lift force tends to push bubbles towards the pipe center, leading to the phenomenon of void coring.

7.8.3. Virtual Mass Force

The virtual mass force is proportional to relative phasic acceleration. The Ansys CFX implementation is applicable to the particle model only. For details, see Virtual Mass Force in the *CFX-Solver Theory Guide*. You may specify a non-dimensional virtual mass coefficient C_{VM}. C_{VM} =0.5 for inviscid flow around an isolated sphere. In general, C_{VM} depends on shape and particle concentration. At the time of this writing, there were no universally accepted models. Such models may be implemented using CEL for C_{VM}.

The virtual mass force is proportional to the continuous phase density, hence, is most significant when the dispersed phase density is less than the continuous phase density. Also, by its nature, it is only significant in the presence of large accelerations, for example, in transient flows, and in flows through narrow restrictions.

7.8.4. Wall Lubrication Force

Under certain circumstances, for example, bubbly upflow in a vertical pipe, the dispersed phase is observed to concentrate in a region close to the wall, but not immediately adjacent to the wall. This effect may be modeled by the wall lubrication force, which tends to push the dispersed phase away from the wall.

The wall lubrication force is usually modeled in conjunction with the wall lift force (see Lift Force (p. 308)). In situations where the lift force pushes bubbles towards the wall, the wall lubrication force acts in the opposite direction to ensure that bubbles accumulate a short distance away from the wall.

Currently, Ansys CFX has the following wall lubrication force models:

- The Antal Wall Lubrication Force Model in the *CFX-Solver Theory Guide*

  Note that the Antal model requires a fine mesh. Grid convergence can be expected only on extremely fine meshes.

- The Tomiyama Wall Lubrication Force Model in the *CFX-Solver Theory Guide*

  Note that the Tomiyama model is available only if the surface tension has been specified for the fluid-dispersed phase pair.

- The Frank Wall Lubrication Force Model in the *CFX-Solver Theory Guide*

Note that the Frank model is available only if the surface tension has been specified for the fluid-dispersed phase pair.

The wall lubrication force uses the near wall distance function. This is computed by CFX-Solver for each phase, based on walls that are specified as no slip wall for that phase. Therefore, walls for which the lubrication force is active must be specified as no-slip walls for the continuous phase.

7.8.5. Interphase Turbulent Dispersion Force

Turbulent Dispersion Forces result in additional dispersion of phases from high volume fraction regions to low volume fraction regions due to turbulent fluctuations. This is caused by the combined action of turbulent eddies and interphase drag. For example, in a dispersed two phase flow, dispersed particles get caught up in continuous phase turbulent eddies, and are transported by the effect of interphase drag. The effect is to move particles from areas of high to low concentration. Hence, this effect will usually be important in turbulent flows with significant interphase drag.

Note:

If you encounter convergence difficulties when using a turbulence dispersion model in conjunction with `Segregated` volume fraction coupling, then you should consider reducing the physical time scale, or switching to `Coupled` volume fraction coupling. For details, see the "Multiphase Control" section under Advanced Options Tab in the *CFX-Pre User's Guide*.

7.8.5.1. Favre Averaged Drag Model

This is based on the Favre or Mass weighted average of the interphase drag force. For details, see Interphase Drag in the *CFX-Solver Theory Guide*. This model has been shown to have a wide range of universality

σ_{tc} is the turbulent Schmidt number for continuous phase volume fraction, currently taken to be 0.9.

Expert users may also modify the model by using a CEL expression for the non-dimensional **Turbulent Dispersion Coefficient**. Its default value is unity.

For theoretical details, see Favre Averaged Drag Model in the *CFX-Solver Theory Guide*.

7.8.5.2. Lopez de Bertodano Model

This is included for backwards compatibility reasons. It requires a **Turbulent Dispersion Coefficient** to be specified. Unfortunately however, there does not exist a universally valid value of the non-dimensional **Turbulent Dispersion Coefficient**. Values of 0.1 - 0.5 have been used successfully for bubbly flow with bubble diameters of order a few millimeters. See Lopez de Bertodano (1998) [21] for a general discussion on recommended values of the turbulent dispersion coefficient.

Note:

- The Lopez de Bertodano turbulence model may not converge well for a coupled volume fraction case that has Turbulent Force discretization. You may be able to

overcome the problem by initializing with a segregated volume fraction case then restarting with coupled volume fraction.

- The Lopez de Bertodano turbulence model cannot be set with the LES model. Instead use the Favre Averaged Drag turbulence dispersion model, which can be used in combination with the LES model.

For theoretical details, see Lopez de Bertodano Model in the *CFX-Solver Theory Guide*.

7.9. Solid Particle Collision Models

The following topics will be discussed:

- Solid Pressure Force Model (p. 311)
- Maximum Packing (p. 312)
- Kinetic Theory Models (p. 312)

7.9.1. Solid Pressure Force Model

The `Solid Pressure Force Model` is available for dispersed solid phases in a multiphase flow. The forces due to solid collisions are taken into account by introducing additional solids pressure and solids stress terms into the solid phase momentum equations based on either the Gidaspow model or by specifying the elasticity modulus directly. Additional theoretical information on these models is available in Solid Particle Collision Models in the *CFX-Solver Theory Guide*.

You can select the Gidaspow Model or specify the Elasticity Modulus for solids pressure on the **Fluid Specific Models** tab, for a particular dispersed solid, when creating a domain in CFX-Pre. For details, see Multiphase Options in the *CFX-Pre User's Guide*. The Gidaspow model requires the **Reference Elasticity Modulus** and the **Compaction Modulus** to be specified. These are used to calculate an Elasticity Modulus. For details, see Solid Particle Collision Models in the *CFX-Solver Theory Guide*. There are no universally accepted values for these. The values used by Bouillard et al. [17] are:

- Reference Elasticity Modulus = 1 Pa
- Compaction Modulus = 20 to 600

For information on setting the **Maximum Packing** parameter in CFX-Pre, see Multiphase Options in the *CFX-Pre User's Guide*.

Results tend to be insensitive to the details of the solids pressure model. The solids pressure gradient is only activated in regions close to the maximum packing, where its tendency is to prevent solid volume fractions from becoming too large.

Using a solid pressure force model in conjunction with the Gidaspow drag model, it is possible to model the large scale features of bubbling fluidized beds. For details, see Densely Distributed Solid Particles (p. 306).

This `Solid Pressure Force Model` is very numerically stiff, prone to convergence problems and may cause divergence. It is needed in some two-phase flow situations, such as a fluidized bed simulation. It should be used with care as it has known issues with robustness. This behavior will be improved in future releases.

There is a more complex set of models that use the `Kinetic Theory of Granular Flow` to model solid pressures and stresses. For details, see Kinetic Theory Models (p. 312).

7.9.2. Maximum Packing

For dispersed phases, you can specify a maximum packing parameter. This is the volume fraction of the phase at its state of maximum packing. It is most commonly used for compact solid dispersed phases, for example, as you would find in fluidized beds.

The maximum packing parameter is unity by default for a dispersed fluid. For a dispersed solid phase, it may range from 0.5 to 0.74, the latter being the maximum possible packing for solid spheres. For most applications, the default value 0.62 suffices.

The maximum packing parameter is used in correlations for certain drag laws, and models for particle collision forces. Unfortunately, it is not possible to numerically guarantee that volume fractions are bounded above by the maximum packing parameter. Consequently, you may observe volume fractions higher than the maximum packing.

7.9.3. Kinetic Theory Models

When using solid particle collision models, the following pieces of advice may be useful:

- Only run in transient mode with very small timesteps, typically 1E-4 [s].
- Adaptive timestepping may help.

Details of the implementation of solid particle collision models are found in Solid Particle Collision Models in the *CFX-Solver Theory Guide*.

7.10. Interphase Heat Transfer

In the multiphase model, there are separate enthalpy and temperature fields for each phase.

7.10.1. Inhomogeneous Interphase Heat Transfer Models

For heat transfer to occur between fluids, at least one fluid must use the `Thermal Energy`, `Total Energy`, or `Small Droplet Temperature` model and at least one other fluid must use the `Isothermal`, `Thermal Energy`, `Total Energy`, or `Saturation Temperature` model. In the case of two-phase flow, a `Thermal Energy`/`Total Energy` - `None` combination results in the modeling of heat transfer only in the phase that uses the `Thermal Energy`/`Total Energy` model. For a `Thermal Energy`/`Total Energy` - `Isothermal`/`Saturation Temperature` combination, heat transfer is not modeled in the isothermal phase but interphase heat transfer can still occur (that is, the isothermal fluid acts like a heat source or sink).

The phases are not in general in thermal equilibrium due to temperature differences across phase boundaries, so heat is transferred across phase interfaces via interphase transfer terms. Heat transfer

across a phase boundary is usually described in terms of a **Heat Transfer Coefficient**, which is the amount of thermal energy crossing a unit area per unit time per unit temperature difference. Ansys CFX contains a number of models for calculating the heat transfer coefficient.

You must specify an interphase heat transfer model if any of the following conditions are true:

- You selected the `Thermal Energy` or `Total Energy` model on the **Fluid Models** tab. In this case heat is transferred to / from both fluids.
- You selected the `Fluid Dependent` heat transfer model on the **Fluid Models** tab, then selected the `Thermal Energy` or `Total Energy` model for both fluids on the **Fluid Specific Models** tab. This case is the same as the previous.
- You selected the `Fluid Dependent` heat transfer model on the **Fluid Models** tab, then selected the `Thermal Energy` or `Total Energy` model for one fluid and the `Isothermal` (or `Saturation Temperature`) model for the other fluid on the **Fluid Specific Models** tab. In this case, heat is transferred to / from the fluid that uses the `Thermal Energy`/`Total Energy` model; that is, the Isothermal fluid acts like a constant temperature heat source or sink.

All other options available for multiphase flow result do not result in interphase heat transfer. Note that selecting `Thermal Energy` or `Total Energy` for one fluid and `None` for the second fluid models heat transfer within one fluid, but no interphase heat transfer occurs.

The `Interphase Heat Transfer` models available for the `Particle` and `Mixture` models are described below, starting with models based on an **Overall Heat Transfer Coefficient**.

7.10.1.1. Particle Model Correlations for Overall Heat Transfer Coefficient

In order to specify the interphase heat transfer term, it is necessary to specify the interfacial area per unit volume and an overall heat transfer coefficient.

Information on the interfacial area density is available. For details, see The Particle Model (p. 302).

It is often convenient to express the heat transfer coefficient in terms of a dimensionless Nusselt number. For steady-state laminar forced convection around a spherical particle, theoretical analysis shows that $Nu = 2$ For a particle in a moving incompressible Newtonian fluid, the Nusselt number is a function of the particle Reynolds number Re and the surrounding fluid Prandtl number Pr.

The following models are available in Ansys CFX:

- `Ranz-Marshall Correlation`. The most well tested correlation for flow past a spherical particle is that of Ranz and Marshall [75]:

  $$Nu = 2 + 0.6\, Re^{0.5}\, Pr^{0.3} \quad 0 \le Re < 200 \quad 0 \le Pr < 250$$

  This is based on boundary layer theory for steady flow past a spherical particle. Its restriction to particle Reynolds number and Prandtl number should be noted.

- `Hughmark Correlation`. Hughmark [7] proposed the following empirical correlation for flow past a spherical particle.

  $$\begin{aligned} &2 + 0.6\, Re^{0.5}\, Pr^{0.33} \quad 0 \le Re < 776.06 \quad 0 \le Pr < 250 \\ &2 + 0.27\, Re^{0.62}\, Pr^{0.33} \quad 776.06 \le Re \quad 0 \le Pr < 250 \end{aligned} \tag{7.3}$$

It extends the Ranz Marshall Correlation and can therefore be applied to a wide range of Reynolds numbers. The Reynolds number cross over point is chosen to guarantee continuity. It should not be used outside the recommended Prandtl number range.

- Nusselt Number. Specify the dimensionless **Nusselt Number** directly. This should be based on empirical correlations, if available, for your application.
- Heat Transfer Coefficient. Specify the **Heat Transfer Coefficient** directly as a value or expression. This should be based on empirical correlations, if available.
- Interface Flux. This is an advanced option which permits experienced users to implement interphase heat transfer models that are not of the simple form of a heat transfer coefficient multiplied by a bulk temperature difference. For details, see Particle Model Correlations in the *CFX-Solver Theory Guide*.

7.10.1.2. Mixture Model Correlations for Overall Heat Transfer Coefficient

If you are using the mixture model, interphase heat transfer may be specified using either:

- A specified overall **Heat Transfer Coefficient**.
- A specified **Nusselt Number**.
- The **Interface Flux** model.

These models are as described above for the particle model.

7.10.1.3. Two Resistance Model for Fluid Specific Heat Transfer Coefficients

This applies to both the particle and mixture models. It is designed for advanced applications in which it is necessary to consider interphase heat transfer on each side of the interface. The Two Resistance Heat Transfer model is an essential ingredient when modeling combined heat and mass transfer due to phase change. The primary application that requires this in Ansys CFX is the Thermal Phase Change model for interphase mass transfer. For details, see Thermal Phase Change Model (p. 324).

In this class of models, you need to specify two heat transfer coefficients: one for each fluid of a specified phase pair. The following points should be noted when using this model.

- The Two Resistance model can be used in conjunction with the Particle or Mixture models.
- Any model that is available for the overall heat transfer coefficient is also available for either of these fluid specific **Heat Transfer Coefficients**, within the limitation described below.
- A fluid specific **Heat Transfer Coefficient** may be specified directly for each fluid.
- A fluid specific **Nusselt Number** may be specified directly for each fluid. In this case, the **Nusselt Number** is always defined relative to the physical properties of the fluid to which it pertains. For details, see The Two Resistance Model in the *CFX-Solver Theory Guide*.
- The Ranz Marshall and Hughmark correlations should only be used on the continuous phase side of a continuous-dispersed phase pair, using the Particle Model.

- It is possible to specify a zero resistance condition on one side of the phase interface. This is equivalent to an infinite fluid specific **Heat Transfer Coefficient**. Its effect is to force the interfacial temperature to be the same as the phase temperature.

Note that all the **Particle Model Overall Heat Transfer** coefficient correlations are equivalent to a `Two Resistance Model`, with the named heat transfer correlation on the continuous phase side, and zero resistance on the dispersed phase side. Hence, one would typically want to use a `Two Resistance Model` in situations where heat transfer in the dispersed phase is significant, for example:

- Transient flows, with small dispersed phase thermal conductivity.
- Condensation onto droplets.

7.10.2. Homogeneous Heat Transfer in Multiphase Flow

Unlike other transport processes, fluid-specific energy equations are solved in homogeneous multiphase flow. A separate equation will appear in the solver manager for each fluid. The results file will also contain fluid-specific temperatures, although you should find that they are essentially equal.

7.11. Polydispersed, Multiple Size Group (MUSIG) Model

The MUSIG model tracks a dispersed phase that is represented by a set of size groups. Each size group represents bubbles of a particular diameter. MUSIG models changes in the size of each group due to:

- Coalescence and breakup
- Thermal phase change
- Sources

The Polydispersed (MUSIG) model has two variants:

- Homogeneous MUSIG

  All size groups for a given polydispersed (MUSIG) fluid move according to the same velocity field.

- Inhomogeneous MUSIG

  The size groups for a given polydispersed (MUSIG) fluid are arranged into multiple velocity groups, with each velocity group associated with a velocity field.

For details on the Polydispersed (MUSIG) model, see Multiple Size Group (MUSIG) Model in the *CFX-Solver Theory Guide*.

The following topics are discussed:

7.11.1. Setting up a Polydispersed (MUSIG or IMUSIG) Simulation
7.11.2. MUSIG Modeling Advice

7.11.1. Setting up a Polydispersed (MUSIG or IMUSIG) Simulation

Important:

Before considering the MUSIG or IMUSIG model as a modeling option, you may want to read MUSIG Modeling Advice (p. 319), which outlines the applicability and limitations of the model.

A partial procedure for setting up a MUSIG or IMUSIG simulation follows:

- Edit the applicable domain that contains the polydispersed fluid(s) and the continuous fluid.

  The details view for the domain appears.

- On the **Basic Settings** tab, under **Fluid and Particle Definitions**, for each fluid in the list that represents a polydispersed fluid, select the fluid in the list then set **Morphology** > **Option** to `Polydispersed Fluid`.

- On the **Polydispersed Fluids** tab, in the list under **Polydispersed Fluid**, define at least one polydispersed fluid.

  Each polydispersed fluid in this list will represent one or more of the polydispersed fluids listed on the **Basic Settings** tab.

- For each polydispersed fluid in the list, (select the fluid in the list then) set **Option** to either `Homogeneous MUSIG` or `Inhomogeneous MUSIG`, as appropriate.

- – For a polydispersed fluid that uses the `Homogeneous MUSIG` option, set **Polydispersed Fluid** (below **Option**) to the corresponding (same basic material) polydispersed fluid (which is available due to being listed on the **Basic Settings** tab).

  – For a polydispersed fluid that uses the `Inhomogeneous MUSIG` option, set **Polydispersed Fluids** (below **Option**) to a selection of some or all of the corresponding (same basic material) polydispersed fluids (which are available for selection due to being listed on the **Basic Settings** tab).

  Each polydispersed fluid in this selection represents a velocity group to which one or more size groups can be assigned.

  The set of polydispersed fluids (each constituting a velocity group) that define a polydispersed fluid should have similar (ideally identical) material properties because they represent the same material. In some cases, a slight difference in material properties is appropriate; for example large bubbles might generally be hotter with a lower density, requiring slightly different thermal properties.

- Under **Size Group**, select each listed size group and then configure that group's **Polydispersed Fluid** and **Size Group Distribution** settings. Note that:

  – For Homogeneous MUSIG, all size groups are forced to belong to the same polydispersed fluid.

  – For Inhomogeneous MUSIG, each size group can belong to any one of the available polydispersed fluids (which are available for selection due to being listed under **Polydispersed Fluids**). All size

groups that belong to the same polydispersed fluid are part of the velocity group represented by that polydispersed fluid, and so share the same velocity field.

  - The Tomiyama diameter may serve as a guide for dividing small bubbles from large bubbles.

- Configure the **Breakup Model** and **Coalescence Model** settings.
- Specify boundary conditions, initial conditions, and sources as appropriate.

  Note that you can use the RPI wall boiling model to drive phase change at walls.

The following topics are discussed:

7.11.1.1. Creating a Polydispersed (MUSIG) Fluid
7.11.1.2. Boundary Conditions
7.11.1.3. Initial Conditions
7.11.1.4. Sources
7.11.1.5. Postprocessing Variables

7.11.1.1. Creating a Polydispersed (MUSIG) Fluid

On the **Polydispersed Fluid** tab, one or more polydispersed fluids can be added per Polydispersed (MUSIG) Fluid Definition. Selecting more than one polydispersed fluid (only for IMUSIG) enables different size groups to move with different velocity fields.

The **Size Group Distribution** setting can be set to `Equal Mass`, `Equal Diameter`, `Geometric`, or `User Defined`. For details, see Size Group Discretization in the *CFX-Solver Theory Guide*. When the **Size Group Distribution** setting is set to `User Defined`, the **Size Group Diameter** parameter must be defined for every **Size Group** object with the groups ordered so that the diameters are increasing monotonically. The **Reference Density** must be set if the density of the polydispersed fluid is not constant; this reference density is used only for assigning the masses of the size groups. When the size group diameter is needed to calculate coalescence and breakup rates, the diameter is calculated from the group mass and the local fluid density.

The group names will have the format `Group <N>`, where `<N>` is an integer denoting the group number (sorted in order of increasing size).

Coalescence and breakup models must be set on the **Polydispersed Fluids** tab. For details, see Multiple Size Group (MUSIG) Model in the *CFX-Solver Theory Guide*.

7.11.1.2. Boundary Conditions

Inlet and Opening boundary conditions require the specification of size fractions for each of the size groups. The size fractions may be set to `Value` or `Automatic`. All size fractions set to `Automatic` are calculated to have the same value such that the overall sum of size fractions (including those that are specified by value) is unity. If all size fractions are set to `Value`, you must ensure that the specified size fractions sum to unity.

7.11.1.3. Initial Conditions

Initial conditions can be set to either `Value` or `Automatic` in the same manner as at inlet and opening boundary conditions.

7.11.1.4. Sources

There are two types of sources that are relevant for polydispersed (MUSIG) fluids:

- The first is a continuity source, which is commonly used to model an inlet without resolving the inlet geometry. The source can be applied to a subdomain, a point, or a boundary. When specifying a continuity source, the size fraction distribution must also be provided. The specified size fractions must sum to unity. By default, if the continuity source is negative (that is, a mass sink), the specified size distribution is ignored and the local size fractions are used instead. If you would like the specified size distribution to be used even if the continuity source is negative, set the setting **MUSIG Sink Option** to `Specified Size Fractions`.
- Sources may also be added directly to the size fraction equations without affecting the continuity equation. Consistency requires that the size fraction sources sum to zero when summed over all size groups.

7.11.1.5. Postprocessing Variables

The MUSIG-specific variables available for postprocessing are:

- `Conservative Size Fraction`

  For a given small volume of the domain, the `Conservative Size Fraction` is the volume of all bubbles contained in a given size group divided by the volume of all bubbles contained in all size groups *of the containing velocity group*.

  Conservative size fractions add to one for all size groups in a given velocity group.

- `Size Fraction`

  `Size Fraction` is derived from the computed value of `Conservative Size Fraction` and represents, for a given small volume of the domain, the volume of all bubbles contained in a given size group divided by the volume of all bubbles contained in all size groups (of all velocity groups) for a polydispersed fluid.

  Size fractions add to one for all size groups (without regard for velocity group).

- `Cumulative Size Fraction` is computed as:

  cumulative size fraction 1 = size fraction 1
  cumulative size fraction 2 = size fraction 1 + size fraction 2
  cumulative size fraction 3 = size fraction 1 + size fraction 2 + size fraction 3
  ...

- `Mean Diameter` (Sauter mean diameter)
- `Interfacial Area Density`

7.11.2. MUSIG Modeling Advice

The MUSIG model is best applied to flows with polydispersed, medium-size bubbles. Medium-sized in this context means bubbles that are large enough to be in the distorted particle regime, but not large enough to be in the spherical cap regime. Within the distorted particle regime, the asymptotic slip velocity of such bubbles is sufficiently small relative to inertial time scales that it can be attained almost instantly. Within this assumption, it is reasonable to assume that all the bubble size groups share a common velocity field. This is the fundamental assumption in the homogeneous MUSIG model.

The assumptions used in deriving the homogeneous MUSIG model break down in the following situations:

- Small spherical bubbles, or large spherical cap bubbles. In these cases, the asymptotic slip velocity is proportional to the square root of the particle diameter.
- Liquid drops (sprays) or solid particles in a gas. In this case, not only are the asymptotic slip velocities dependent on particle size, but the particle relaxation times may be comparable to, or larger than, the inertial time scale. The MUSIG model would not predict segregation of solid particles of different sizes in a fluidized bed.
- Flows where non-drag forces are significant and depend on bubble size. For example, the lift force exerted on bubbly flows in pipes has a separating effect, moving large bubbles to the pipe center and small bubbles to the outside.

For flows where some bubbles are expected to move independently of larger bubbles, consider using IMUSIG. The Tomiyama diameter may serve as a guide for dividing size groups.

7.12. Turbulence Modeling in Multiphase Flow

This section describes the use of turbulence models in multiphase simulations. You should be familiar with the contents of Turbulence and Near-Wall Modeling (p. 197), which describes the use of turbulence models in single-phase flows, before reading this section.

7.12.1. Phase-Dependent Turbulence Models

Phase dependent turbulence models can be used in conjunction with the inhomogeneous model (particle and mixture models) only. Each phase can use a different turbulence model. All models available for single-phase calculations are also available for continuous phases in multiphase calculations, with the exception of the LES model. The models available for dispersed phases are limited to the laminar model or zero equations models because other models are not considered to be appropriate. For example, a recommended model for dilute dispersed two-phase flow uses a two-equation model, such as the SST model or k–ε model, for the continuous phase, and an algebraic eddy viscosity model for the dispersed phase, which simply sets the dispersed phase viscosity proportional to the continuous phase eddy viscosity.

For flows where the phases tend to separate, homogeneous turbulence is recommended. See the sections below for details.

7.12.1.1. Algebraic Models

If using the zero-equation algebraic turbulence model, there are several options:

- The default zero-equation model uses a formula based on geometric length scale and the mean solution velocity. This should be used with caution for multiphase flow, as it is correlated for single-phase turbulent pipe flow.
- You may supply your own prescription for eddy viscosity, using a constant or an expression.
- For dispersed phases only, you may invoke the `Dispersed Phase Zero Equation` turbulence model. This is the recommended algebraic model for a dispersed phase. It is only available for the disperse fluid when the continuous fluid is set to use a turbulence model (that is, not laminar).

  An **Eddy Viscosity Prandtl Number** can be specified. The default value of 1 is appropriate for bubbles or very small solid particles. For large solid particles in a gas phase is may be better to use a value greater than 1. This is highly model dependent. Several models are available in the literature.

7.12.1.2. Two Equation Models

The k–ε model in multiphase is very similar to the single-phase model. It is recommended for use in continuous phases. The other two equation turbulence models can be used for continuous phases, if desired.

7.12.1.3. Reynolds Stress Models

The multiphase versions of Reynolds stress models are equivalent to the single phase version.

7.12.2. Homogeneous Turbulence in Inhomogeneous Flow

When using the inhomogeneous model (particle and mixture models), you can select to solve a homogeneous turbulence field. In this case, a single turbulence field is solved for using a single turbulence model. This option is recommended for free surface flow using the inhomogeneous model, separated flow, stratified flow and any situation where the phases tend to separate out.

Note that using the fluid-dependent turbulence model option, when the same turbulence model is used in each phase, is not the same as using the homogeneous turbulence option. In the latter, a single turbulence field is solved instead of two separate fields that happen to use the same model.

The turbulence model chosen should be based on the advice given for single phase flow. For details, see Turbulence and Near-Wall Modeling (p. 197).

7.12.3. Turbulence Enhancement

In flows with a dispersed phase, large particles in the dispersed phase tend to increase turbulence in the continuous phase due to the presence of wakes behind the particles. This is known as particle induced turbulence. The `Sato Enhanced Eddy Viscosity` model can be used to model this effect. It is only available for fluid pairs using `Continuous` | `Dispersed` morphology. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*.

7.12.4. Turbulence in Homogeneous Multiphase Flow

A common turbulence field exists in homogeneous multiphase flow, therefore a single turbulence field is solved using a single turbulence model. For this type of flow, the single phase flow modeling advice is applicable. For details, see Turbulence and Near-Wall Modeling (p. 197).

7.13. Additional Variables in Multiphase Flow

Additional Variables can exist in all fluids or only a limited number of fluids in a multiphase simulation. In addition, both homogeneous and phase-specific Additional Variables are supported. Phase-specific Additional Variables may be transferred between two phases provided that, for both phases, a transport equation is solved for the Additional Variable and a kinematic diffusivity has been set. It is also possible to have interphase transfer from a transported Additional Variable into an algebraic Additional Variable (that is, the algebraic Additional Variable acts like an Additional Variable source or sink).

7.13.1. Additional Variable Interphase Transfer Models

It is possible for an Additional Variable Φ_α to be coupled to a different Additional Variable Ψ_β across a phase interface between fluids α and β. Such a situation may arise, for example, when modeling the evaporation of water in a solid phase to water vapor in a gaseous phase. The only restriction is that Φ_α and Ψ_β have the same physical dimensions. Two such coupled Additional Variables constitute an Additional Variable pair $\Phi_\alpha \mid \Psi_\beta$ associated with the phase pair $\alpha \mid \beta$.

Transfer of an Additional Variable across a phase boundary is described by an **Additional Variable Transfer Coefficient**, which is analogous to the **Heat Transfer Coefficient**.

It is often convenient to express the Additional Variable transfer coefficient in terms of a dimensionless Sherwood number *Sh*, analogous to the **Nusselt Number** in heat transfer. The Sherwood number is calculated based on the kinematic diffusivity for a volumetric variable, or the dynamic diffusivity for a specific variable.

7.13.1.1. Particle Model Correlations

For laminar forced convection around a spherical particle, theoretical analysis shows that *Sh = 2*. For a particle in a moving incompressible Newtonian fluid, the Sherwood number is a function of the particle Reynolds number *Re* and the Additional Variable Prandtl number *Pr*.

The following models are available in Ansys CFX:

7.13.1.1.1. Ranz-Marshall Correlation

This model is valid in the following ranges:

$$Sh = 2 + 0.6\, Re^{0.5}\, Pr^{0.3} \quad 0 \le Re < 200 \quad 0 \le Pr < 250 \tag{7.4}$$

7.13.1.1.2. Hughmark Correlation

This model is valid in the following ranges:

$$Sh = \begin{cases} 2 + 0.6\, Re^{0.5}\, Pr^{0.33} & 0 \le Re < 776.06 \quad 0 \le Pr < 250 \\ 2 + 0.27\, Re^{0.62}\, Pr^{0.33} & 776.06 \le Re \quad 0 \le Pr < 250 \end{cases} \tag{7.5}$$

7.13.1.1.3. Sherwood Number

You specify the Sherwood number directly. This should be based on empirical correlations, if available, for your application.

7.13.1.1.4. Additional Variable Transfer Coefficient

You specify the Additional Variable Transfer Coefficient directly. This should be based on empirical correlations, if available.

7.13.1.1.5. Interface Flux

This is an advanced option that permits experienced users to implement interphase Additional Variable transfer models that are not of the simple form of a transfer coefficient multiplied by a bulk Additional Variable difference. You must specify the Additional Variable flux coefficients for both fluids and Additional Variable flux value from Fluid 1 to Fluid 2. For details, see Particle Model Correlations in the *CFX-Solver Theory Guide*.

7.13.1.2. Mixture Model Correlations

If you are using the mixture model, interphase transport for Additional Variables may be specified using one of the following:

- A specified **Interphase Transfer Coefficient**
- A specified **Sherwood Number**
- The **Interface Flux Model**

7.13.2. Homogeneous Additional Variables in Multiphase Flow

Homogeneous Additional Variables are available for both homogeneous and inhomogeneous flow. However, in most cases, the homogeneous assumption is inappropriate for volumetric Additional Variables. For this reason, only specific Additional Variables are permitted to be homogeneous in Ansys CFX.

7.14. Sources in Multiphase Flow

Sources in multiphase flow are very similar to those in single phase flow. Information on the single-phase source documentation is available. For details, see Sources in the *CFX-Solver Theory Guide*. Information on setting source terms in CFX-Pre is available in Sources Tab in the *CFX-Pre User's Guide*.

There are two types of sources in multiphase flow:

- Fluid-specific sources. For details, see Fluid-Specific Sources (p. 323).
- Bulk sources. For details, see Bulk Sources (p. 323).

7.14.1. Fluid-Specific Sources

Fluid-specific sources are applied directly to fluid as specified and are not multiplied by volume fraction; that is, the sources are assumed to be specified per unit volume of geometry, not per unit volume of the phase. If you have a source expression that is per unit volume of the phase, you must either multiply the source by volume fraction or else use **Bulk Sources**, which are discussed below.

Fluid-specific sources are not available for equations in homogeneous multiphase flow that span all fluids; **Bulk Sources** are the only appropriate option for these situations.

Examples of the application of fluid-specific sources:

- Momentum sources:

  Momentum sources must be multiplied by the volume fraction. For details on momentum sources, see Momentum Sources in the *CFX-Solver Theory Guide*.

- Isotropic and directional loss models:

  The linear resistance coefficient must be multiplied by the corresponding phase volume fraction. The quadratic resistance coefficient should be multiplied by the square of the corresponding phase volume fraction.

- Permeability:

  Resistance coefficients must be multiplied by the corresponding phase volume fraction. The permeability must be divided by the corresponding phase volume fraction.

- Quadratic loss coefficients:

  Quadratic loss coefficients must be multiplied by the square of the corresponding phase volume fraction.

7.14.2. Bulk Sources

Bulk Sources automatically multiply the specified source by volume fraction, ensuring that the source reduces to zero as the fluid volume fraction reduces to zero. They can be used for equations that span all fluids as well as those that are fluid-dependent, such as mass fractions and Additional Variables. However, they may not be used for mass sources.

7.15. Interphase Mass Transfer

Interphase mass transfer occurs when mass is carried from one phase into another. It is applicable to both the inhomogeneous and homogeneous multiphase models. Some of the model libraries available in CFX-Pre contain simulation definitions for interface mass transfer cases, including boiling water and cavitation.

Possible causes of interphase mass transfer are:

- Change of thermodynamic phase. For example, melting/solidification in liquid-solid systems, evaporation/condensation in gas-liquid systems, and cavitation in gas-liquid systems.

- Diffusion of a dissolved species across a phase boundary. This may or may not involve a change of phase of the dissolved species. Examples are gas dissolution, and evaporation of a liquid into a gas containing its vapor.
- Breakup and coalescence may be treated as a mass transfer process between two phases representing different size groups of the same species.

7.15.1. Double Precision Solver

In some cases, using the double precision solver executable can improve convergence for interphase mass transfer cases.

7.15.2. User Specified Mass Transfer

For advanced applications, it is possible to directly specify the interphase mass transfer sources. You may specify the interfacial mass flux or the interfacial mass flow (volumetric mass source) between any pair of phases. The latter is more appropriate for volumetric mass transfer processes such as breakup and collision. For details, see User Defined Interphase Mass Transfer in the *CFX-Solver Theory Guide*.

7.15.3. Thermal Phase Change Model

This model describes phase change induced by interphase heat transfer; it may be used to simulate boiling and condensation, or melting and solidification. For example, it may be used to model condensation of saturated vapor bubbles in sub-cooled liquid, or evaporation of saturated bubbles in super-heated liquid. It is only applicable to phase changes of pure substances.

This section provides modeling information. For a discussion of the theory, see The Thermal Phase Change Model.

The Thermal Phase Change model requires a number of related items of information to be provided.

7.15.3.1. Saturation Temperature

There are two possible ways of defining the saturation temperature:

- Using the optional parameter **Saturation Temperature** (on the **Fluid Pair Models** tab under **Phase Change Model** with **Option** set to `Thermal Phase Change`). It may be set as a constant, or an expression. If the latter, it should be defined as a function of absolute pressure.

  Note:

  This option will not be available if you have created a homogeneous binary mixture definition.

- Alternatively, you may define a homogeneous binary material (HBM) in the **Material** details view. For details, see Material Details View: Homogeneous Binary Mixture in the *CFX-Pre User's Guide*. The HBM should consist of the two materials undergoing phase change, and the saturation conditions should be defined as material properties of the HBM.

You must use one of these methods to define the Saturation Temperature, otherwise an error will be generated. If you use both methods, then the **Saturation Temperature** parameter (of the `Thermal Phase Change` option) takes precedence.

7.15.3.2. Wall Boiling Model

Wall boiling starts when the wall temperature becomes sufficiently large to initiate the activation of wall nucleation sites. This activation temperature is typically a few degrees above the saturation temperature. However, at this stage, the average temperature of the liquid in the vicinity of the heated wall is still well below the saturation temperature, hence in the sub-cooled boiling regime.

Wall boiling can be controlled in CFX-Pre from:

- The domain details view, on the **Boundary Models** tab.
- The boundary details view for a wall boundary, on the following tabs:
  - The **Boundary Details** tab, which has the **Wall Boiling Model** > **Option** setting
  - The **Fluid Pair Values** tab, which is available when, on the **Boundary Details** tab, **Wall Boiling Model** > **Option** is set to **RPI Model**
  - The **Fluid Values** tab, which has fluid-specific settings related to wall boiling.

Ansys CFX implements an RPI boiling model. Usage information is provided in RPI Model (p. 325). For a theoretical discussion of the wall boiling model in CFX, see Wall Boiling Model.

The following topics are discussed:

7.15.3.2.1. RPI Model

7.15.3.2.2. Using a Wall Boiling Model

7.15.3.2.1. RPI Model

The RPI model for near-wall boiling can be controlled by several settings as described next:

- **Bubble Diam. Influence Factor**

  Enables you to specify the bubble diameter influence factor. For details, see Partitioning of the Wall Heat Flux.

- **Max. Area Frac. of Bubble Influence**

  Enables you to specify the maximum area fraction of bubble influence. For details, see Area Influence Factors and Evaporation Rate.

- **Onset of Boiling Superheating**

  CFX-Solver assumes that boiling starts when T_{wall} exceeds T_{sat} by more than the specified **Onset** value.

- **Mass Source Under-Relaxation**

This factor under-relaxes only $\dot{m}$, which is the evaporation mass transfer rate per unit wall area. It is recommended that you set the **Mass Source Under-Relaxation** factor and the **Wall Heat Flux Partitioning Under-Relaxation** factor to the same value to avoid inconsistencies in the wall energy fluxes that could lead to convergence problems.

- **Wall Superheating Under-Relaxation**

  This factor under-relaxes only ΔT_{sup}, which is the unknown in wall heat partitioning. You may specify this factor independently of any applied partitioning under-relaxation or mass source under-relaxation.

- **Wall Heat Flux Partitioning Under-Relaxation**

  This under-relaxation factor permits consistent relaxation of the individual heat partitions (convection to liquid, quenching, and evaporation) in order to fulfil the given wall heat flux boundary condition. It is recommended that you set the **Wall Heat Flux Partitioning Under-Relaxation** factor and the **Mass Source Under-Relaxation** factor to the same value to avoid inconsistencies in the wall energy fluxes that could lead to convergence problems.

- **Bubble Departure Diameter** enables you to specify the bubble departure diameter. The available options are:

  – `User Defined`

  – `Tolubinski Kostanchuk`

    For details, see Bubble Departure Diameter.

  – `Delta Function`

    The `Delta Function` option is available only when using the MUSIG/IMUSIG approach for the dispersed phase; there must be at least one polydispersed fluid in the fluid pair.

    Specify a value for **Delta Diameter**. The size group with the nearest diameter will have its size fraction equation influenced by the RPI boiling model, resulting in the production of bubbles within that size group.

  – `Distribution Function`

    The `Distribution Function` option is available only when using the MUSIG/IMUSIG approach for the dispersed phase; there must be at least one polydispersed fluid in the fluid pair.

    Specify a CEL function for **Distribution Function**. Given a diameter, this function must evaluate to the probability (between 0 and 1) that a newly departing bubble has that diameter. The integral of the probability over all diameters must be one. For example, you could specify that bubbles with diameters between 1 mm and 5 mm are equally likely, and that no other diameters are allowed, by using the following CEL expression:

    ```
    if (Fluid1|Fluid2.Bubble departure diameter < 1[mm], 0, if (Flu
    id1|Fluid2.Bubble departure diameter < 5[mm], 0.25, 0))
    ```

    The CFX-Solver discretizes this probability function into groups corresponding to the defined size groups.

- **Bubble Departure Temperature** enables you to specify the bubble departure temperature. The available options are:

  – `User Defined`

    With this option, you can define the bubble departure temperature using an expression.

  – `Saturation Temperature`

    With this option (which is the default), the bubble departure temperature is made equal to the saturation temperature.

  – `Superheat Fraction`

    With this option, the bubble departure temperature is made equal to the saturation temperature plus the specified fraction of the superheat.

- **Wall Nucleation Site Density** enables you to specify the wall nucleation site density. The available options are:

  – `User Defined`

  – `Lemmert Chawla`

  For details, see Wall Nucleation Site Density.

- **Bubble Detachment Frequency** enables you to specify the bubble detachment frequency. The available options are:

  – `User Defined`

  – `Terminal Velocity over Departure Diameter`

  For details, see Bubble Detachment Frequency.

- **Bubble Waiting Time** enables you to specify the bubble waiting time. The available options are:

  – `User Defined`

  – `Proportional to Detachment Period`

  For details, see Bubble Waiting Time.

- **Liquid Quenching Heat Transfer Coefficient** enables you to specify the liquid quenching heat transfer coefficient. The available options are:

  – `User Defined`

  – `Del Valle Kenning`

  For details, see Quenching Heat Transfer.

- **Flux Transfer Model** enables you to model heat transfer from the wall to the phase in contact with the wall. The only available option, `Convective`, requires **Critical Vol. Frac.** to be set

to a value for the critical volume fraction of the applicable fluid. Whenever the volume fraction of that fluid is greater than the critical value, the simulation allows convective heat transfer to that fluid.

With the vapor volume fraction below the critical level for the vapor phase, the wall is treated as if it is covered by a liquid film. Above the critical level, the liquid film is assumed to break down rapidly, leaving the wall exposed to the vapor so that some of the wall heat flux contributes directly to convective heating of the vapor phase.

The default value of **Critical Vol. Frac.** for a gas is 1.0, which models the vapor near the wall as being saturated, with no part of the wall heat flux contributing to superheating of the vapor phase.

For details, see Area Influence Factors in the *CFX-Solver Theory Guide* and Convective Heat Transfer in the *CFX-Solver Theory Guide*.

- Fluid-specific **Wall Boiling Model** settings, which are found in the domain details view on the **Boundary Models** tab and in the boundary details view on the **Fluid Values** tab, enable you to specify information about the boundary layer temperature via **Boundary Layer Liquid Temperature** settings. The available options are:

  – `Fixed Yplus`, which enables you to specify a fixed Y^+ value at which the temperature is estimated for use in the bubble departure diameter and quenching heat transfer correlations. For details, see Bubble Departure Diameter.

  – `User Line Cloud Average`

    This option uses a line averaged liquid temperature as set up in the specified line cloud for calculating the liquid quenching flux and the bubble departure diameter. When setting up the line averaging for the liquid temperature in the line cloud, the distance over which the average is taken should be based on the bubble departure diameter.

    The variable `<liquid name>.Temperature` must be available in the list of variables for line averaging, as specified under **Line Variables List** in the settings of the specified user line cloud.

    For details on user line clouds, see User Line Cloud in the *CFX-Pre User's Guide*.

Note:

If you are using the `User Defined` option for any of these settings, and any of these user-defined settings use CEL expressions that involve wall superheat, then it is important to specify the temperature using the `Tsuperheat` variable and not the phase-specific temperature variables such as `water.Temperature`.

7.15.3.2.2. Using a Wall Boiling Model

Walls that have boiling may have a specified temperature, specified heat flux, or specified heat transfer coefficient. Walls that have boiling may not have fluid-specific heat transfer boundary conditions. A wall boiling model has its own algorithms for computing wall contact area fractions for each phase; for the purpose of computing wall heat transfer, any user-specified wall contact model is overridden.

Wall boiling must be switched on explicitly for the walls at which it is expected to occur. On the **Boundary Details** tab for a boundary, set **Wall Boiling Model** > **Option** to one of the following values:

- `None`

  No wall boiling model is applied at the wall.

- `From Domain`

  A boiling model is applied at the wall using the boiling model settings of the domain.

- `RPI Model`

  The RPI model is used to model wall boiling on the wall. The primary details of the wall boiling model are defined on the domain form, but these may be overridden on the wall boundary by using settings on the boundary's **Fluid Pair Values** tab.

A wall boiling model cannot be used:

- For a laminar continuous phase.

  Laminar dispersed phases are permitted only if they do not have convective heat transfer with the wall.

- On a boundary where the continuous phase has a free slip wall condition.

- In conjunction with the MUSIG model.

- In conjunction with thermal radiation models.

7.15.3.3. Latent Heat

This is not specified directly. Rather, it is obtained *indirectly* as the difference between the static enthalpies of the two phases. For example:

$$L = H_{\text{gas}}\left(T_{\text{sat}}\right) - H_{\text{liquid}}\left(T_{\text{sat}}\right) \tag{7.6}$$

Hence, it is essential that the static enthalpy fields contain the absolute enthalpies of the two phases. This should be taken care of automatically if you select materials from the materials properties data base. On the other hand, if you define your own material properties, you should ensure that the enthalpies are defined correctly by setting the following fields in the **Material** details view:

- Reference Temperature, T_{ref}
- Reference Pressure, p_{ref}.
- Reference Specific Enthalpy h_{ref}.

Definitions of these quantities are available. For details, see Library Materials (p. 97).

For example, if you know the latent heat L at a reference state (T, P), then one way of ensuring that the correct latent heat is obtained at all temperatures and pressures is to set:

$$T_{\text{ref}}(\text{liq})=T\,,\quad P_{\text{ref}}(\text{liq})=P\,,\quad H_{\text{ref}}(\text{liq})=0$$
$$T_{\text{ref}}(\text{gas})=T\,,\quad P_{\text{ref}}(\text{gas})=P\,,\quad H_{\text{ref}}(\text{gas})=L \tag{7.7}$$

7.15.3.4. Heat Transfer Models

In general, both phases should be assigned either the Thermal Energy or Total Energy model for heat transfer.

However, in the case of subcooled boiling, you should find that the vapor phase temperature remains fixed at saturation conditions. Hence, in the case of constant saturation conditions, it is possible to run the vapor phase as Isothermal, with Reference Temperature set equal to the vapor saturation temperature.

7.15.3.5. Interphase Heat Transfer Correlations

The Thermal Phase Change Model assumes:

- That thermodynamic equilibrium prevails at the interface between the two phases. That is, that the interfacial temperature equals the saturation temperature.
- That heat transfer either side of the phase interface may be modeled by two independent heat transfer coefficients. For details, see Two Resistance Model for Fluid Specific Heat Transfer Coefficients (p. 314).

Hence, the Thermal Phase Change model requires the use of the Two Resistance model for interphase heat transfer. Selection of appropriate heat transfer correlations depends on the situation being modeled. Suggestions are provided below. First, some terminology:

- A phase is said to be **saturated** if its temperature equals the saturation temperature.
- It is said to be **subcooled** if its temperature is below saturation.
- It is said to be **superheated** if its temperature is above saturation.

7.15.3.6. Modeling Advice

7.15.3.6.1. Saturated Vapor Bubbles in Subcooled or Superheated Liquid

Vapor temperature is expected to remain at saturation conditions. There is negligible resistance to heat transfer on the dispersed phase side. Hence:

- Under *constant* saturation conditions, you should set **Zero Resistance** on the **dispersed** phase side.
- Under *variable* saturation conditions, you should set a large but finite **Nusselt Number** (of the order of 1000) on the **dispersed** phase side.

The primary resistance to heat transfer is on the continuous liquid phase side. Hence:

- For spherical bubbles, you should set either the **Ranz Marshall** or **Hughmark** correlation on the **continuous** phase side.

- For non-spherical bubbles, you may want to replace these by a user-defined correlation for **Heat Transfer Coefficient** or **Nusselt Number**.

7.15.3.6.2. Subcooled or Superheated Droplets in Saturated Vapor

The situation here is entirely the opposite to the above, as there is now negligible resistance to heat transfer on the *continuous* phase side. Hence:

- Under constant saturation conditions, you should set **Zero Resistance** on the **continuous** phase side.
- Under **variable** saturation conditions, You should set a large but finite **Nusselt Number** (of the order of 1000) on the **continuous** phase side.

The primary resistance to heat transfer is on the *dispersed* droplet phase side. This is a fundamentally transient phenomenon for which there are no universally established correlations.

The simplest possible model on the **dispersed** phase side is:

$$Nu \approx 6 \tag{7.8}$$

This is based on the analytic solution of transient heat conduction inside a solid sphere. It assumes:

- That advection effects inside the drop may be neglected.
- That the time-dependent temperature field inside the sphere may be considered to be spatially constant.

7.15.3.6.3. Superheated Vapor Bubbles in Liquid

In this case, there is significant resistance to heat transfer on both the dispersed and continuous phase sides. Hence:

- For spherical bubbles, use either the **Ranz Marshall** or **Hughmark** correlation on the **continuous** phase side.
- On the dispersed phase side, use a correlation appropriate to heat transfer inside a spherical fluid particle, such as the $Nu \approx 6$ discussed above.

7.15.3.6.4. Thermal Energy and Total Energy Models

- Use Thermal Energy in cases where low-speed flow exists and where viscous effects are negligible.
- Use Total Energy in the case of high-speed flow and where viscous effects are not negligible.

Note:

The **Incl. Viscous Work Term** option should be selected for a case involving MFR (multiple frames of reference) and the Total Energy equation. For details, see The Total Energy Equation in the *CFX-Solver Theory Guide*.

7.15.4. Cavitation Model

Cavitation refers to the process by which vapor forms in low pressure regions of a liquid flow. It occurs in a variety of applications such as throttles, pumps and injectors.

In Ansys CFX, cavitation can be modeled using different approaches, as described in Approaches to Modeling Cavitation in the *CFX Reference Guide*. This section describes a cavitation model for interphase mass transfer.

Most simulations can use the homogeneous multiphase model because the vapor velocity field is often assumed to be the same as that of the liquid. However, the inhomogeneous model can be used if desired.

When solving a cavitating flow problem, you should first obtain a converged solution with the cavitation model turned off at the conditions where you would expect to see cavitation. The cavitation model can be turned on and off in the domain details view, on the **Fluid Pair Models** tab under **Mass Transfer**. The initial guess should use a liquid volume fraction of 1 and a gas vapor volume fraction of 0.

It should be emphasized that if a poor initial guess is supplied with the cavitation model active (that is, a non-zero initial vapor volume fraction), then it is possible to arrive at a physically impossible situation where most of the flow domain is cavitating.

For cavitation problems, the pressure level should be set at one of the boundaries, or by setting pressure level information on the **Advanced** tab of the **Solver Control** settings in CFX-Pre. For details, see Pressure Level Information (p. 576). This is because the cavitation rate is driven by the difference between the local pressure and vapor pressure, so the pressure level is important. Inflow boundaries would normally use a vapor volume fraction of 0 because the vapor is generated within the domain.

When solving cavitation problems, several variables in the results file are new or are modified from their standard behavior:

- **Interphase Mass Transfer Rate**: The cavitation rate is calculated by the solver.
- **Pressure**: The cavitation model is a model for the interphase mass transfer rate, and does not guarantee that all absolute pressures calculated by the solver will be positive. They will, however, be less negative than if cavitation is not modeled.

  In order to make postprocessing more convenient, the variable called `Pressure` that is written to the results file is clipped in such a way that negative values for absolute pressure are clipped to zero. For example, if the reference pressure is 1e5 Pa, the `Pressure` variable is clipped to be not smaller than -1e5 Pa. The actual (non-clipped) pressure field computed by the solver can be visualized using the variable called `Solver Pressure`.

  Note, however, that the solver will output `Pressure` with non-clipped values (as for `Solver Pressure`) if you refer to `Pressure`:

  – in an expression

  – for output to a selected variables transient file

  – in the definition of a monitor point

the solver will interpret this as `Solver Pressure` (non-clipped values). To use the clipped pressure in these contexts, use the variable name `Clipped Pressure` instead.

- **Absolute Pressure**: To avoid robustness issues associated with negative absolute pressure (for example if the simulation includes an ideal gas equation state), the variable called `Absolute Pressure` is clipped to be no smaller than the saturation pressure. However, the true absolute pressure must be used to calculate the cavitation rate. This variable is not available in the results file, but may be accessed in CEL (for example, for user defined cavitation models) with the name `Nonclipped Absolute Pressure`.
- **Density**: To enhance numerical stability, the vapor density field is clipped in a user-controlled fashion through the `Maximum Density Ratio` parameter, described below. The vapor density field in the results file shows the clipped values. The unclipped density (which is used to calculate the cavitation rate itself) is available in the variable field called `Nonclipped Density`.

The cavitation model appears as an interphase mass transfer option. If you choose cavitation as a mass transfer model, you have the choice between the Rayleigh Plesset model and a user-defined model:

7.15.4.1. Rayleigh Plesset Model

The Rayleigh Plesset cavitation model requires the following parameters:

- **Saturation Pressure**: The saturation pressure must be specified as a value or expression. Note that this is an absolute, not relative, pressure. In Ansys CFX, saturation data is *not* read from material property files (such as `.rgp` files used for defining real fluids). There are two possible ways of defining the saturation pressure:
  - Using the optional parameter Saturation Pressure under **Rayleigh Plesset Model** on the Fluid Pairs tab. It may be set as a constant, or an expression. If the latter, it should be defined as a function of temperature.
  - Alternatively, you may define a Homogeneous Binary Material (HBM) under the **Material** details view. The HBM should consist of the two materials undergoing phase change, and the saturation conditions should be defined as material properties of the HBM.

    You must use one of these methods to define the Saturation Pressure; otherwise, an error will be generated. If you use both methods, then the Saturation Pressure parameter of the **Rayleigh Plesset Model** takes precedence.
- **Mean Diameter**: The mean nucleation site diameter must be specified. The default value of 2e-06 m is a reasonable value.

The following optional parameters can also be set. If not specified, the default values shown are used. These defaults are appropriate for most cavitation simulations.

- **Cavitation Condensation Coefficient**: This is an empirical factor to account for the fact that condensation usually occurs slowly. The default value is 0.01.
- **Cavitation Vaporization Coefficient**: This is an empirical factor to account for the fact that vaporization usually occurs quickly. The default value is 50.

- **Maximum Density Ratio**: This is used to clip the vapor density for all terms except the cavitation source term itself, which must use the true density specified as the material property. The default value is 1000.
- **Nuclei Volume Fraction**: This is the volume fraction of the nucleation sites. The default value is 5e-4.
- **Cavitation Rate Under-Relaxation Factor**: The default value is 0.25.

Additional theoretical information on this model is available. For details, see The Rayleigh Plesset Model in the *CFX-Solver Theory Guide*.

7.15.4.2. User Defined Cavitation Models

A user-defined cavitation model can be implemented for advanced users. It requires the following parameters:

- **Cavitation Rate**: This should be set through CEL or User Fortran. The cavitation rate is simply $\dot{m}_{fg}$, the interphase mass transfer rate per unit volume from the liquid to the vapor phase, assuming the first fluid in the fluid pair is the liquid and the second is the vapor. If the order in the pair is reversed, the sign must be reversed. As an example, the interphase mass transfer rate for the Rayleigh Plesset model in Ansys CFX is given by The Rayleigh Plesset Model in the *CFX-Solver Theory Guide*.
- **Saturation Pressure**: The saturation pressure must be specified as a value or expression. This may be set in the same way as explained above for the Rayleigh-Plesset Model.

The following optional parameters can also be set. If not specified, the default values shown are used. These defaults are appropriate for most cavitation simulations.

- **Maximum Density Ratio**: This is used to clip the vapor density for all terms except the cavitation source term itself, which must use the true density specified as the material property. The default value is 1000.
- **Cavitation Rate Under-Relaxation Factor**: The default value is 0.25.

When developing an expression for a user-defined cavitation rate:

- It was noted above that the cavitation model does not guarantee that all absolute pressures will be positive. If you calculate the vapor density as a CEL function of pressure, make sure you use the variable `pabs` in your expression. This variable is clipped to the saturation pressure, which is numerically well-behaved even when the absolute pressure becomes negative.
- The cavitation rate is driven by $p_v - p$. The pressure used in this expression should be the non-clipped absolute pressure (variable name is `pabsnc`).
- The cavitation rate is typically proportional to the vapor density. The vapor density, however, may have been clipped by the **Maximum Density Ratio** parameter, which is inappropriate in this context. Instead, the true vapor density should be used, which is available as the variable `densitync`.

Solver information about user-defined models is available. For details, see User Defined Cavitation Models in the *CFX-Solver Theory Guide*.

7.15.5. Interphase Species Mass Transfer

This describes mass transfer of dissolved species by diffusion across a phase interface. Examples are:

- **Evaporation** of a liquid into a gaseous mixture containing its vapor; for example, the evaporation of water droplets into air + water vapor.
- **Absorption/Dissolution** of a dissolved gas in a liquid from a gaseous mixture; for example, the absorption of ammonia by water from a mixture of air+ammonia.

Additional theoretical information is available in General Species Mass Transfer in the *CFX-Solver Theory Guide*.

Species transfer requires a number of related items of information to be provided.

7.15.5.1. Component Pairs

In Ansys CFX, interphase species transfer (or component transfer) may be specified between any two components, *A* and *B*, in fluids, α and β, respectively, subject to the following conditions:

- The mass fractions of *A* and *B* must both be determined from transport equations.
- It is not possible to specify the species transfer of a component whose mass fraction is determined algebraically, or from the constraint equation.
- Unless the **Mass Transfer** option `Sum Species Mass Transfers` is selected, the resulting interphase mass flow rate is sufficiently small that continuity equation sources may be neglected, and latent heat effects in the energy equation are negligible.

The coupled components *A* | *B* constitute a *Component Pair* associated with the fluid pair α | β. Component pairs are analogous to Additional Variables pairs. For details, see Additional Variables in Multiphase Flow (p. 321).

For more information on setting up component pairs in CFX-Pre, see Eulerian | Eulerian Pairs in the *CFX-Pre User's Guide*.

7.15.5.2. Two Resistance Model

The species transfer model assumes:

- That dynamic equilibrium prevails at the interface between the two phases.
- That mass transfer either side of the phase interface may be modeled by two independent mass transfer coefficients.

Hence, species transfer is implemented as a Two Resistance Model, analogous to the Two Resistance Model for heat transfer. For details, see:

- Two Resistance Model for Fluid Specific Heat Transfer Coefficients (p. 314)
- Two Resistance Model with Negligible Mass Transfer in the *CFX-Solver Theory Guide*.

7.15.5.3. Single Resistance Model

The Single Resistance options Ranz Marshall, Sherwood Number, Hughmark and Mass Transfer Coefficient apply the corresponding correlation on the continuous fluid side and a zero resistance on the dispersed phase side of the interphase. The correlations are analogous to Additional Variable mass transfer. For details, see Particle Model Correlations (p. 321).

7.15.5.4. Interfacial Equilibrium Models

A number of interfacial equilibrium models are available. Different ones are recommended for different physical situations. Additional theoretical information is available in Equilibrium Models in the *CFX-Solver Theory Guide*.

7.15.5.4.1. Liquid Evaporation (Raoult's Law)

For evaporation of an ideal liquid into a gas containing its vapor, **Raoult's Law** should be employed to describe the equilibrium condition. This requires the following additional information:

- The **Thermodynamic Phase** (liquid or gas) must be declared on the **Materials Properties** tab for each fluid α and β.
- The **Saturation Pressure** must be declared for the vapor component. This may be a constant or an expression. In the latter case, it should be a function of absolute temperature. There are two possible ways of defining the saturation pressure:
  - Using the optional parameter **Saturation Pressure** under the option for Raoult's Law in the **Component Pair Details** frame. It may be set as a constant or an expression. If the latter, be defined as a function of temperature.
  - Alternatively, you may define a Homogeneous Binary Material (HBM) under the **Material** details view. The HBM should consist of the two components in the component pair and the saturation conditions should be defined as material properties of the HBM.
  - You must use one of these methods to define the Saturation Pressure; otherwise, an error will be generated. If you use both methods, then the **Saturation Pressure** parameter of **Raoult's Law** takes precedence.

7.15.5.4.2. Gas Absorption / Dissolution (Henry's Law)

For absorption/dissolution of a dissolved gas from a liquid into a gaseous mixture, **Henry's Law** should be employed to describe the equilibrium condition. This requires the following additional information:

- The **Thermodynamic Phase** (liquid or gas) must be declared on the **Materials Properties** tab for each fluid α and β.
- **Henry's Coefficient** must be declared for the vapor component.

Care should be exercised in defining the Henry coefficient. There is no agreed convention as to its correct definition, so a units conversion may be necessary. Ansys CFX has a choice of two conventions:

Molar Fraction Henry Coefficient, H^x, with units of pressure, or

Molar Concentration Henry Coefficient, H^c, with units of pressure per molar concentration.

7.15.5.4.3. Other Situations

In situations where the above two simple laws do not apply, such as liquid-liquid extraction, or multicomponent transfer, you may specify empirical data for the equilibrium condition, if available. For this purpose, you have a choice of three dimensionless quantities:

- **Molar Concentration Equilibrium Ratio**
- **Molar Fraction Equilibrium Ratio**
- **Mass Fraction Equilibrium Ratio**.

These specify the equilibrium ratios of the indicated concentration variables, for the first fluid divided by the second fluid.

7.15.5.5. Species Mass Transfer Coefficients

The Two Resistance model applies to both the Particle and Mixture models, which determine the interfacial area density. It remains to specify the two mass transfer coefficients on each side of the phase interface.

The available correlations for mass transfer coefficients are analogous to those for heat transfer and Additional Variables.

- A fluid specific **Mass Transfer Coefficient** may be specified directly for each fluid.
- A fluid specific **Sherwood Number** may be specified directly for each fluid. The Sherwood number is always defined relative to the physical properties of the fluid to which it pertains:

$$Sh_\alpha = \frac{k_\alpha d_{\alpha\beta}}{\rho_\alpha D_\alpha} \tag{7.9}$$

k_α is the mass transfer coefficient and $d_{\alpha\beta}$ is the interfacial length scale (the mean particle diameter for the Particle Model and the mixture length scale for the Mixture Model).

- **Ranz Marshall** and **Hughmark** correlations are available on the **continuous phase side only** of a continuous-dispersed phase pair, using the **Particle Model**.
- It is possible to specify a **Zero Resistance** condition on one side of the phase interface. This is equivalent to an infinite fluid specific mass transfer coefficient $k_\alpha \to \infty$. Its effect is to force the interfacial concentration to be the same as the bulk concentration of that phase.

The selection of physically correct mass transfer correlations is highly problem dependent. In the case of mass transfer between a continuous phase and a dispersed phase of approximately spherical particles, the following should be adequate for most problems of interest:

- **Ranz Marshall** or **Hughmark** in the continuous phase.
- **Zero Resistance** in the dispersed phase.

The latter assumption implies that mass transfer occurs very quickly between the dispersed phase and the interface. Consequently, interfacial concentrations on the dispersed phase side are assumed equal to the bulk dispersed phase concentrations. This will usually be valid for droplet evaporation, or gas absorption / dissolution in bubbles.

In cases where there is significant resistance to mass transfer on the dispersed phase side, the Zero Resistance model should be replaced by a user-specified correlation for dispersed phase Mass Transfer Coefficient or Sherwood Number.

7.15.5.6. Modeling Advice

7.15.5.6.1. Liquid Evaporation

For evaporation of a pure liquid into its vapor, use Raoult's Law for the interfacial equilibrium model. For most cases of droplet evaporation, the resistance mass transfer on the droplet side is negligible. If that is the case, use Zero Resistance on the dispersed phase side, and Ranz-Marshall or Hughmark on the continuous phase side.

If the liquid consists of just one component, you should define it as a two component mixture with a placeholder material as the Constraint material, and the pure liquid as the Transported material. You will then be able to define a component pair consisting of the pure liquid coupled to its vapor component in the gaseous phase.

7.15.5.6.2. Gas Dissolution

Use Henry's Law for the interfacial equilibrium model. If the resistance to mass transfer on the dispersed phase side is negligible, then use Zero Resistance on the dispersed phase side, and Ranz-Marshall or Hughmark on the continuous phase side.

7.15.5.6.3. Mixture Properties

By default, Ansys CFX computes thermodynamic properties of mixtures assuming that they are ideal mixtures. This is not appropriate, for example, for gases dissolved in a liquid. If the effect of the gas phase properties (for example, density) on the liquid phase properties is negligible, then you explicitly specify mixture properties in CFX-Pre to override the solver calculated mixture properties with appropriate values. The most important mixture property to override is density, but molar mass, dynamic viscosity and specific heat capacity should also be defined if you believe the solver calculated values will be inappropriate. Information on how the solver calculates mixture properties is available in Radiation Properties (p. 95).

7.15.5.6.4. Current Limitations

The Ansys CFX implementation of component mass transfer is a preliminary implementation, with some limitations that are repeated here.

- The resulting interphase mass flow rate is assumed to be sufficiently small that continuity equation sources may be neglected.
- Consequently, latent heat effects in the energy equation are neglected.

Thus, component mass transfer is assumed to transfer very small amounts of material between phases, and the driving force is assumed to be governed solely by concentration differences. Component mass transfer due to heat transfer is not yet modeled.

7.15.6. Droplet Condensation Model

To activate the Droplet Condensation Model (or Nonequilibrium Steam Model when steam is the working fluid), use the following steps:

1. Choose your vapor and liquid materials, as well as a homogeneous binary mixture to define saturation properties. For each condensed phase, there should be a binary mixture defined between each droplet phase and the continuous phase. For steam calculations, the IAPWS library is recommended.

2. Choose a multiphase simulation with one continuous vapor phase and as many dispersed liquid phases as appropriate. (Each droplet phase is assumed to be monodispersed at each point in space, although the diameter can differ through the domain.)

3. For many condensing droplet applications, the homogeneous multiphase model is appropriate. When the droplets become bigger (for example, 1 μm or larger), the inhomogeneous multiphase model should be considered.

4. On the **Fluid Models** tab, for the heat transfer model, ensure that **Homogeneous Model** is de-activated and select the `Fluid Dependent` option.

5. On the **Basic Settings** tab, set the droplet phase morphology option to `Droplets (Phase Change)`.

6. On the **Fluid Specific Models** tab, set the continuous phase heat transfer model option to `Total Energy`.

7. Subsequent modeling choices depend on whether the droplets are small or not.

   For small (less than 1 μm) droplets, the following recommendations apply:

   - On the **Fluid Specific Models** tab, with the droplet phase selected, set **Heat Transfer Model** > **Option** to `Small Droplet Temperature` (recommended) or `Total Energy`. Note that the droplet temperature setting implies that no transport equation is solved for the energy state of the droplet, but is determined from an algebraic relationship valid for sub-micron droplets.

   - On the **Fluid Specific Models** tab, with the droplet phase selected, activate the homogeneous nucleation model if you expect nucleation to occur in the domain. For multidomain applications (for example, multistage turbines), the nucleation model can be controlled independently for each domain. This allows you to model polydispersed droplets where a new set of droplets nucleates in each stage component (rotor or stator). To do this, one droplet phase can be defined for each turbine component (or group of components). For a particular component (group), the nucleation model can be activated only for the corresponding droplet phase. Each droplet phase will continue to grow through the entire machine.

     With **Nucleation Model** > **Option** set to `Homogeneous`, you must also specify an option for the surface tension temperature model (**Surf. Tension Model**): `Gas Temperature` or `Interface Temperature`. The surface tension for newly formed droplets is evaluated at the

temperature of the gas or interface, according to the selected option. In accordance with classical nucleation theory and based on equilibrium assumptions, the default option is `Gas Temperature`.

If you need to adjust the bulk tension of newly formed droplets, select **Nucleation Bulk Tension Factor** and set **Value** to a scaling factor.

- On the **Fluid Pair Models** tab, set **Mass Transfer** > **Option** to `Phase Change` and set **Phase Change Model** > **Option** to `Small Droplets`.
- Also on the **Fluid Pair Models** tab, set **Heat Transfer** > **Option** to `Small Droplets`, then configure the **Thermal Diffusion Model** settings.

  The **Thermal Diffusion Model** options are `Young` and `Gyarmathy`. For details, see The Droplet Condensation Model in the *CFX-Solver Theory Guide*.

For larger droplets, the following recommendations apply:

- On the **Fluid Specific Models** tab, with the droplet phase selected, set **Heat Transfer Model** > **Option** to `Total Energy` (recommended) or `Thermal Energy`.
- On the **Fluid Pair Models** tab, set **Heat Transfer** > **Option** to `Two Resistance`. On the continuous phase side, select the Ranz Marshall correlation, while on the droplet side, a Nusselt number of 6 is appropriate. For mass transfer, select `Phase Change` and choose the thermal phase change model

In addition, keep the following in mind when setting boundary conditions and initial conditions:

- Droplets may or may not appear through inlets and openings. If they do appear, choose the droplet volume fraction and droplet diameter (or droplet number) as appropriate. If they do not enter the boundary, choose zero for the volume fraction and droplet number; the droplet diameter is irrelevant
- The gas temperature at inlets, openings, and for initial conditions should not extend too far in the supercooled (or superheated) region when a second phase is present. A good rule of thumb is that these temperatures should not be more than 5 [K] below (or above) the saturation temperature, or else nonphysical mass transfer rates will develop, which could cause divergence.

The following procedure is often helpful to solve droplet condensation problems:

1. Obtain a solution with the nucleation model turned off. This allows the supercooling to develop naturally in the flow, sets up a realistic expansion rate, and provides a very good initial guess for the full calculation.
2. Activate the nucleation model.

7.16. Boundary Conditions in Multiphase Flow

The use of boundary conditions for multiphase flow is very similar to that for single-phase flow. For details, see Boundary Condition Modeling (p. 111). The main differences are:

- Boundary conditions need to be specified for all fluids for all variables except the shared pressure field.
- Volume fractions of all phases must be specified on inlet and opening boundary conditions. These must sum to unity.
- A degassing boundary condition is available for multiphase flow containing a dispersed phase. For details, see Degassing Condition (Multiphase only) (p. 136).

7.16.1. Wall Boundaries in Multiphase

For multiphase flows, it is convenient to enable two distinct methods for you to specify wall boundary conditions:

- Bulk wall boundary conditions are defined as boundary condition attributes of the wall, and subsequently shared among the phases in a well-defined manner. The algorithm for sharing wall fluxes among fluids depends on the concept of a wall contact model (which will be explained shortly).
- Alternatively, wall boundary conditions may be defined on a per fluid basis. This is a more advanced option, giving you complete flexibility in the definition of multiphase wall boundary conditions.

7.16.1.1. Bulk Wall Boundary Conditions

This is the simplest and most useful option. In most applications, wall boundary conditions are known as attributes of the wall. It is then part of the modeling procedure to decide how the consequent wall fluxes are allocated among the phases in contact with the wall.

For example, for bulk heat transfer, you may specify either:

- A bulk heat flux to the wall, Q_{wall}. The heat fluxes Q_α to each phase α are determined as follows:

$$Q_\alpha = A_\alpha \, Q_{\text{wall}} \tag{7.10}$$

  Where A_α is the contact area fraction of phase α at the wall, calculated from the Area Contact Model as described below.

- A bulk heat transfer coefficient h_{wall} and outer temperature T_o. The heat fluxes to each phase are determined as follows:

$$Q_\alpha = A_\alpha \, h_{\text{wall}} \left(T_o - T_\alpha \right) \tag{7.11}$$

  The difference between the specified wall temperature and the calculated phase temperature together with the heat transfer coefficient determines the heat flux to the phase. The contact area fraction is also taken into account.

- A bulk wall temperature T_{wall}. The specified wall temperature is allocated to all phases in contact with the wall, and resulting wall heat fluxes are partitioned among phases using the contact area fraction.

7.16.1.1.1. Area Contact Model

By default, the contact area fraction A_α of phase α at the wall is identical to the volume fraction in the control volume adjacent to the wall:

$$A_\alpha = r_\alpha \tag{7.12}$$

However, there are situations in which an alternative model may be required, in which case it can be overridden by providing a value or an expression for each phase. For example, for annular gas-liquid flow, the liquid forms a thin film at the wall and is therefore sensible to assign:

$$A_{\text{liquid}} = 1 \quad A_{\text{gas}} = 0 \tag{7.13}$$

Similarly, in dispersed gas-solid flows with spherical solid particles, the wall contact area of the dispersed phase may be considered to be negligible relative to the continuous phase.

Hence, Ansys CFX includes a facility for you to override the default wall contact area model by providing a value or an expression for the contact area fraction for each phase. Note:

- Contact area fractions must sum to unity over all phases.
- Specified contact area fractions are applied to all transport processes at the wall, that is, wall stresses, heat fluxes, Additional Variable fluxes, and so on.
- You should only use the simple model:

  $$A_\alpha = 1 \quad A_\beta = 0 \tag{7.14}$$

  in situations where phase α does not vanish at the wall. Otherwise, it is possible that you will be applying a non-zero wall flux to a region of zero volume of phase α, which could cause the calculation to overflow. For the same reason, your initial guess should not set the fluid volume fraction to be zero at a wall when that phase uses a non-zero area fraction.
- In general it, is desirable, though not essential, to express contact area fractions as functions of volume fraction such that:

  $$A_\alpha \to 0 \quad \text{as} \quad r_\alpha \to 0 \tag{7.15}$$

  This ensures that finite wall fluxes do not enter regions of zero volume of phase α.

7.16.1.2. Fluid Dependent Wall Boundary Conditions

This option is included to give the advanced user complete flexibility in assigning wall boundary conditions to individual phases. For example, for **heat transfer**, you may specify either:

- The temperature, T_α of phase α, or
- The heat flux Q_α to phase α, or
- Phase specific heat transfer coefficient h_α outer temperature $T_{o\alpha}$, in which case:

  $$Q_\alpha = h_\alpha \left(T_{o\alpha} - T_\alpha \right) \tag{7.16}$$

Note:

- These options may be mixed arbitrarily among the phases.
- Heat fluxes arising from phase specific isothermal boundary conditions are automatically multiplied by contact area fraction.

- However, specified heat fluxes Q_α are **not** automatically multiplied by contact area fractions A_α. You should include these as multiplicative factors in your expressions for Q_α or h_α if this is the effect that you require.
- Care should be exercised to ensure:

$$Q_\alpha \to 0 \quad \text{as} \quad r_\alpha \to 0 \tag{7.17}$$

to prevent overflow errors due to finite sources being applied to zero fluid volumes.

7.16.1.3. Wall Deposition

This option is only available for the Algebraic Slip Model. For details, see Wall Deposition (p. 351).

7.16.2. Mass Flow Inlet

Information on the mass flow rate inlet boundary condition for single phase flow is available in Mass Flow Rate (p. 120).

For multiphase flow, mass flow inlets may be specified either for a particular phase or for the bulk flow. In both cases, the flow direction and the volume fraction profile at the inlet must be specified.

If the mass flow rate for a particular phase is specified, then the mass flux (defined as mass flow per unit area of the phase) is assumed to be uniform over the entire boundary.

Boundary conditions for other phases are also required. These may also be phasic mass flow conditions, or any other available velocity specifications. The phasic mass flow inlet condition is not available for homogeneous multiphase simulations.

If a bulk mass flow inlet condition is applied, then the bulk mass flux is assumed to be uniform over the entire boundary. The mass flux of each phase is calculated from the bulk mass flux according to the specified volume fraction profile.

The bulk mass flow inlet condition is available for both homogeneous and inhomogeneous simulations.

7.16.3. Mass Flow Outlet

Information on the mass flow rate outlet boundary condition for single-phase flow is available in Mass Flow Rate (Bulk Mass Flow Rate for Multiphase) (p. 133). For multiphase flow, mass flow outlets can be specified either for a particular phase or for the bulk flow. In all cases, the volume fraction and pressure profiles at the outlet are an implicit part of the calculation. The option to specify a pressure profile together with the mass flow rate for single-phase flows is not available for multiphase simulations.

If the mass flow rate for a particular phase is specified, then the mass flow distribution is a function of the mass flow and volume fraction profiles just upstream of the outlet boundary, adjusted to enforce the specified mass flow rate of that phase.

Boundary conditions for the other phases are also required. These may also be phasic mass flow conditions or any of the other available velocity conditions. Two particular cases that may be appropriate are a zero velocity specification, in which case the boundary is closed to the phase, and a no

slip velocity condition, which is possible using a CEL expression. The phasic mass flow outlet condition is not available for homogeneous multiphase simulations.

If a bulk mass flow rate outlet condition is applied, then the bulk mass flow distribution is again a function of the mass flow and volume fraction profiles just upstream of the boundary, adjusted to enforce the specified bulk mass flow rate.

The bulk mass flow outlet condition is available for both homogeneous and inhomogeneous multiphase simulations.

7.17. Modeling Advice for Multiphase Flow

This section describes:

- Turbulence Models (p. 344)
- Minimum Volume Fraction Setting (p. 344)
- Buoyancy (p. 345)
- Initial Conditions (p. 345)
- Timestepping (p. 345)
- Convergence (p. 345)
- Transient Simulations (p. 345)

For related advice, see Modeling Advice for Free Surface Flow (p. 348).

7.17.1. Turbulence Models

For flows in which a dilute phase is dispersed in a continuous phase, you should use the `Fluid Dependent` turbulence option with the k–ε model for the continuous fluid and the `Dispersed Phase Zero Equation` option for the dispersed phase. At wall boundaries, the dispersed phase should also use the `Free Slip Wall` option for **Mass And Momentum**. This type of flow can occur in bubble columns.

The turbulent Prandtl number for an algebraic model should be unity, except for large heavy particles, in which case it should be greater than unity.

For flows in which the phases tend to separate (for example, inhomogeneous free surface flows and oil-water separators), the homogeneous turbulence model is recommended.

7.17.2. Minimum Volume Fraction Setting

The Default option can almost always be used. If this causes numerical problems, then set the value to be several orders of magnitude less than the minimum expected physical value, but larger than machine zero.

7.17.3. Buoyancy

For details, see Buoyancy (p. 58).

7.17.4. Initial Conditions

You should ensure that the default values for the initial fluid volume fraction are appropriate for your simulation. The default volume fraction for dispersed fluids is zero, and for continuous fluids it is 1/N, where N is the number of continuous phases. Note that you can leave one fluid to use the `Automatic` option - the remaining volume fraction will be assigned to this fluid such that the volume fractions sum to unity. For cases involving free-surface flows, there are additional considerations for setting the initial conditions. For details, see Initial Conditions (p. 349).

7.17.5. Timestepping

For bubble columns, use a fraction of the bubble rise time for a total of one rise time, then increase the timestep to a fraction of the fluid circulation time. If species mass transfer is modeled, it can take a considerable amount of time for some species to come to equilibrium. Try setting the timestep size for the mass fraction equations to a large value if convergence is very slow for this equation class.

For dilute particle cases, use a fraction of the dynamical time scale for the momenta. This is essentially a fluid travel time through or around the region of interest, but is not always trivial. If volume fraction convergence is slow, then the time scale for these equations can be increased suitably.

7.17.6. Convergence

If convergence stalls completely or becomes very slow, check the conservation and imbalances at the end of the CFX-Solver Output file. If these are acceptable, plot the residual fields. It may be that the residual fields are good almost everywhere except for a few 'hot spots.' In this case, the solution is valid. You can also monitor some important quantity; for example, in a bubble column you can monitor if gas hold-up has reached steady-state.

For a case with a very dilute phase, the continuous phase will have a volume fraction very close to unity everywhere. Rounding errors will affect the residuals at a much higher level for the continuous phase in this case. By default, the volume fraction residual target is 10 times the global target residual.

For some multiphase simulations, notably free surface and interphase mass transfer cases, using the double precision solver executable can improve convergence significantly. For details, see Advection Schemes for Multiphase Volume Fractions (p. 574).

7.17.7. Transient Simulations

The default of 10 coefficient loops per timestep is usually adequate to resolve the strong nonlinearities present in multiphase flows. It may sometimes be necessary to use a larger number during the initial transient. If you feel that convergence is not sufficient with this number of coefficient loops, you should reduce the timestep size rather than increase the number of coefficient loops.

If possible, a transient should be started from a small perturbation of a converged steady-state result. This may reduce the number of coefficient loops required during the initial transient.

7.18. Free Surface Flow

Free surface flow refers to a multiphase flow situation where the phases are separated by a distinct interface. Examples of free surface flows include open channel flow, flow around ship hulls, tank filling problems, Pelton turbines, and many others.

Free surface flow with the homogeneous model should be used when possible. Free surface flow with the inhomogeneous model can be used to enable the two phases to separate. This will be required if entrainment of one phase within another occurs and you want to allow the phases to separate again.

7.18.1. Interface Compression Level

This controls the volume fraction advection scheme details for free surface flows, which in turn controls the interface sharpness. Allowable values are:

Setting	Value
0	No special compression — The advection scheme is the same as the global advection scheme setting.
1	Intermediate compression
2	Aggressive compression — This is the default.

7.18.2. Supercritical and Subcritical Flow

When referring to free surface flows, a supercritical flow has a Froude number > 1 and a subcritical flow has a Froude number <1.

$$\text{Froude number} = \frac{U}{\sqrt{g\,l}} \tag{7.18}$$

where U is the local fluid velocity, g is gravity and l is a length scale. For shallow flow, the length scale is the depth of the fluid. The term $\sqrt{g\,l}$ is the surface wave speed.

Supercritical flow has a local velocity greater than the surface wave speed, while subcritical flow has a velocity less than the wave speed. This is analogous to supersonic and subsonic flow where the local fluid velocity is compared to the speed of an infinitesimal pressure wave to determine the flow regime.

7.18.3. Multiphase Model Selection

Free surface flows can be simulated when using the inhomogeneous (particle or mixture models) or the homogeneous model. Use the following rules-of-thumb to decide which model is appropriate:

- Use the homogeneous model if the interface is distinct and well-defined everywhere.
- If the interface is not well-defined in some locations, perhaps because one phase is entrained in the other, an inhomogeneous model may be more appropriate. For example, if there is significant splashing, air becomes entrained in the water and behaves as a dispersed phase. In this case, the particle model, using air as the dispersed phase, can give superior results because the unequal velocity fields allow the air and water phases to separate and form a distinct interface. The disad-

vantage of the inhomogeneous models is that they require more CPU time and memory to run. Note also that whichever inhomogeneous model you choose has the same limitations as discussed in the previous chapter; for example, the accuracy of the particle model decreases at high dispersed phase volume fractions.

7.18.4. Surface Tension

7.18.4.1. Background

Surface tension is the force that holds the interface together. If an interface is cut, the magnitude of the surface tension force per unit length is called the surface tension coefficient, σ. Surface tension has a number of important physical effects, including:

- If the interface is curved, it induces a force normal to the interface. For a droplet at rest, this induces a pressure rise within the droplet of $\Delta p=\sigma\kappa$, where κ is the droplet curvature. The effect of this normal force is to smooth regions of high curvature; it tends to reduce the surface area of droplets.
- When the surface tension coefficient is not constant, the surface tension force has a tangential component that tends to move fluid along the interface toward regions of high surface tension coefficient. This is often called Marangoni convection.
- When the free surface interface touches a wall, the wall may attract the liquid (wetting wall) or repel the liquid (non-wetting wall). In a static situation, the wall contact angle may be measured. This phenomenon, known as wall adhesion, gives rise to effects such as the meniscus and capillary rise in tubes.

For free surface flows, you have the option of including surface tension effects. When it is enabled, a surface tension coefficient must be specified. If the surface tension coefficient is variable, the Marangoni force is automatically included. If the surface tension model is activated, using double precision is often required to avoid roundoff errors in the curvature calculation.

Additional information on the surface tension implementation in Ansys CFX is available in Surface Tension in the *CFX-Solver Theory Guide*.

7.18.4.2. Discretization Options

7.18.4.2.1. Volume Fraction Smoothing Type

When evaluating the interface normal vector for the curvature and surface tension force, it is usually beneficial to take the gradient of a smoothed volume fraction field. The volume fraction smoothing method may be optionally set in CFX-Pre. The options are `None`, `Laplacian`, and `Volume-Weighted`. `Volume-Weighted` is the default. `Laplacian` is slightly more accurate, but also slightly more expensive, than `Volume-Weighted`. Occasionally, when the surface tension force is driven only by wall adhesion and there is no interior curvature, it may be beneficial to choose `None`.

7.18.4.2.2. Curvature Under-Relaxation Factor

This is used for underrelaxing the curvature. The default value is 1 (unity), but for flows strongly driven by surface tension, it may be useful to use a smaller value (for example, 0.25).

7.18.4.3. Initial Conditions

When setting initial conditions for surface tension calculations, it may also be beneficial to assign a smeared volume fraction rather than a discontinuous one. The hyperbolic tangent function (available in CEL) is useful for this purpose. For example, the following expressions defined a smeared droplet of radius 0.3 m centered at (x,y) = (0.5 [m], 0.5 [m]):

```
rdrop = 0.3 [m]

dist = rdrop -sqrt((x - 0.5[m])^2 + (y - 0.5[m]^2)

delta = 0.01 [m]

drop = 0.5*tanh(dist/delta)+0.5
```

These expressions represent:

`rdrop` - droplet radius

`dist` - signed distance to the droplet interface

`delta` - smearing distance

`drop` - the smeared volume fraction field: 1 inside the droplet, 0 outside, and smoothly varying between 0 and 1 over a range of about 3*delta on each side of the interface.

7.18.4.4. Wall Adhesion

To account for wall adhesion, you should specify a contact angle when setting wall boundary conditions. This is the angle the interface makes with the wall through the **Primary Fluid** that is prescribed on the **Fluid Pairs** tab of the **Domain** details view when the **Surface Tension Coefficient** is set. If the **Primary Fluid** is a liquid and the secondary fluid is a gas then, for non-wetting walls, the contact angle is greater than 90 degrees, while for wetting walls the contact angle is less than 90 degrees.

7.18.5. Modeling Advice for Free Surface Flow

7.18.5.1. Domains

For the Buoyancy Reference Density, choose the density of the light fluid (air). This simplifies the definition of pressure, and hence the specification of pressure initial and boundary conditions. In the hydrostatic limit:

$$\frac{dp}{dz} = -\left(\rho - \rho_{ref}\right) g \tag{7.19}$$

For details, see Buoyancy (p. 58).

7.18.5.2. Turbulence Model

If the inhomogeneous multiphase model is selected, using the homogeneous turbulence option is recommended. For details, see Homogeneous Turbulence in Inhomogeneous Flow (p. 320).

7.18.5.3. Boundary Conditions

Make sure that a pressure level has been set by one of the boundary conditions. For "closed box" cases, you can add a small opening to achieve this.

7.18.5.3.1. Pressure Reference

A pressure level must be set at one or more boundary conditions by using the **Pressure Level Information** option on the **Solver Control** tab in CFX-Pre. For details, see Turbulence Control (p. 576). To set a pressure reference value using a boundary condition for "closed box" cases, this can be achieved by adding a small pressure opening to the top wall.

7.18.5.3.2. Outlets

At pressure outlets, the elevation of the water must usually be known as part of the problem definition. The mechanism used to enforce this elevation is to set a pressure profile consistent with the known elevation. In some circumstances, the elevation is not known in advanced (for example supercritical outlets). In this case, it is best to specify a pressure profile consistent with an estimate for the outlet elevation. This assumption should not affect the flow other than possibly inducing a small kink in the flow just upstream of the outlet condition.

7.18.5.4. Initial Conditions

The initial conditions for the pressure field and volume fraction must be consistent (that is, the pressure field is hydrostatic in the heavy phase and uniform in light phase). This is easily done using CEL step functions. For example, the functions:

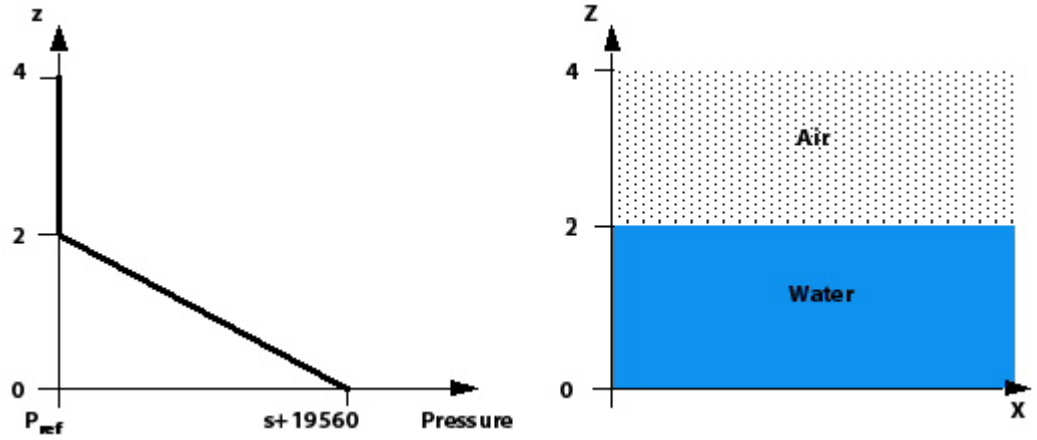
$$h = 2\,[m] \tag{7.20}$$

$$\text{initial VOF Air} = \text{step}\left(\left(z - h\right) / \ 1\,[\text{m}]\right) \tag{7.21}$$

$$\text{initial VOF Water} = 1 - \text{initial VOF Air} \tag{7.22}$$

$$\text{initial } P = 998\,[\text{kg m\^{}}-3] * g * \left(h - z\right) * \text{initial VOF Water} \tag{7.23}$$

would be appropriate to initialize the *relative* pressure field and volume fractions as shown below.



In the equation for *initialVOFAir*, the division is required because the step function must operate on dimensionless quantities.

7.18.5.5. Timestep

The timestep for free surface flows should be based on a L/U (Length/Velocity) scale. The length scale should be a geometric length scale. The velocity scale should be the maximum of a representative flow velocity and a buoyant velocity, which is given by:

$$U_{\text{buoyant}}=\sqrt{g\,L} \tag{7.24}$$

In addition, it is often helpful to reduce the timestep for the volume fraction equations by an order of magnitude below that of the other equations.

7.18.5.6. Mesh Adaption

When using free surface with adaption, make sure the interface is no longer moving before doing an adaption step, otherwise the adaption will be a wasted effort.

7.18.5.7. Body Forces

The **Body Force Averaging Type** controls how body forces (buoyancy in particular) are averaged to elements. The default option of `Volume-Weighted` should be used in most cases. For free surface flows, you may want to use the `Harmonic` option; this is a little more robust and is less prone to generate spurious velocities in the gas phase adjacent to the interface. However, harmonic averaging does not converge as well as volume-weighted averaging. The arithmetic option is very similar to the `Volume-Weighted` option.

7.18.5.8. Convergence

Convergence is often hard to achieve in a residual sense. Instead use some global quantity to check if the solution is changing. Failure to converge residuals is often manifest as small spurious waves on the free surface interface. These waves may be reduced in magnitude if the timestep for the volume fraction equation is an order of magnitude smaller than for the momentum equation.

In some cases, using the double precision solver executable can improve convergence.

- Hydro Control (p. 576)
- Advection Schemes for Multiphase Volume Fractions (p. 574)

7.18.5.9. Parallel

Free surface may not be robust in parallel if any portion of a partition boundary is aligned with the free surface. In this case, consider trying another of the partitioning methods (recursive bisection or specified direction). If GGI boundaries are present and the domains are of the same physical type (fluid/porous and solid), consider using coupled partitioning. Finally, reducing the value of the expert parameter 'overlap relaxation fluids' from its default value of 1 may be helpful.

7.19. Algebraic Slip Model (ASM)

In multiphase flow, each phase has its own velocity field, governed by conservation of momentum for the phase. In `Eulerian-Eulerian` multiphase flow, the full momentum equation is considered, in-

cluding inertial effects of the phase. In some cases, however, if the time scales to reach the equilibrium slip velocity are small, it is appropriate to use simplified models for calculating the velocity of dispersed phases. This is the motivation behind the Algebraic Slip Model (ASM) for multiphase flow.

7.19.1. Algebraic Slip Model Specification

The ASM model is not set up in the same way as other multiphase problems. Only one fluid for a variable composition mixture is used in the **Domains** tab on CFX-Pre. When setting up a problem to run with the Algebraic Slip Model, you should first create a material for the variable composition mixture. For details, see Material Details View: Variable Composition Mixture in the *CFX-Pre User's Guide*. When setting the thermodynamic state of the mixture, the choice of liquid or gas is not important. When specifying the domain options, define the fluid in **Fluids and Particle Definitions...** and select the fluid **Material**.

7.19.1.1. Fluid Models

The continuous phase should be set to `Constraint` on the **Component Details** tab. The remaining phases can be set to either `Algebraic Slip` or `Transport Equation`. `Transport Equations` and `Algebraic Slip` components can be combined, meaning that a mixture of components may be present in the continuous phase. At least one component must be set to `Constraint`. For any dispersed phases, choose the `Algebraic Slip` option.

The **Algebraic Slip** option can be set to `Drag Force Balance` or `Slip Velocity`. Drag Force Balance uses the closed relationship for the slip velocity. For details, see Derivation of the Algebraic Slip Equation in the *CFX-Solver Theory Guide*.

$$\left|u_{S\alpha}\right| u^i_{S\alpha} = \frac{4}{3}\frac{d_p}{\rho_c C_D}\left(\rho_\alpha - \rho_m\right)\left(\frac{\partial u^i_m}{\partial t} - u^j_m \frac{\partial u^i_m}{\partial x^j} - g^i\right) \tag{7.25}$$

where the subscript m refers to bulk quantities and α refers to dispersed phase quantities. The drag coefficient C_D can be directly specified or the Schiller Naumann model can be used. In both cases, the mean diameter d_p must be specified.

The slip velocity may also be directly specified.

7.19.1.2. Wall Deposition

For wall boundaries, there is the option of specifying wall deposition for a dispersed phase. When deposition is selected for a phase that contacts a wall boundary, the mass of the particular phase is removed from the calculation and replaced with the mass of the ballast phase (the component that was selected as `Constraint` on the **Fluid Models** tab). The deposition rate is determined from the slip velocity of the algebraic slip component.

7.19.1.3. Limitations

As the model makes a number of assumptions, the model will not produce the best results when:

- Non-drag forces are significant.
- The dispersed phase does not reach terminal velocity quickly (that is, particles with significant mass are present).

For details, see Derivation of the Algebraic Slip Equation in the *CFX-Solver Theory Guide*.

7.20. Multiphase Flow Restrictions

Radiation is not supported in multiphase flow in Ansys CFX.

Chapter 8: Particle Transport Modeling

This chapter describes how Particle Transport is modeled:

8.1. Model Validity
8.2. Particle Transport Versus Eulerian-Eulerian Multiphase
8.3. Forces Acting on the Particles
8.4. Creating Particle Materials
8.5. Particle Domain Options
8.6. Particle Boundary Options and Behavior
8.7. Subdomains
8.8. Particle Injection Regions
8.9. Particle Output Control
8.10. Particle Solver Control
8.11. Multiphase Reactions and Combustion
8.12. Restrictions for Particle Transport
8.13. Restrictions for Particle Materials
8.14. Convergence Control for Particle Transport
8.15. Expert Parameters for Particle Tracking
8.16. Particle Diagnostics
8.17. Integrated Particle Sources for the Coupled Continuous Phase
8.18. Transient Simulations: What is Different for Particles?

Related information is provided in Particle Transport Theory in the *CFX-Solver Theory Guide*.

Multiphase flow refers to the situation where more than one fluid is present. Each fluid may possess its own flow field, or all fluids may share a common flow field. Unlike multicomponent flow, the fluids are not mixed on a microscopic scale in multiphase flow. Rather, they are mixed on a macroscopic scale, with a discernible interface between the fluids. Ansys CFX includes a variety of multiphase models to enable the simulation of multiple fluid streams, bubbles, droplets, solid particles and free surface flows.

Two distinct multiphase flow models are available in Ansys CFX: an **Eulerian-Eulerian** multiphase model and a **Lagrangian Particle Tracking** multiphase model. Additional information on the `Eulerian-Eulerian` model is available in Multiphase Flow Modeling (p. 295).

The Particle Transport model is capable of modeling dispersed phases which are discretely distributed in a continuous phase. The modeling involves the separate calculation of each phase with source terms generated to account for the effects of the particles on the continuous phase. Water droplets dispersed in air from the liquid sprays of a cooling tower, and solid particles dispersed in air from the pneumatic transport of solids, or transport of airborne particulates are such practical occurrences.

The implementation of particle transport modeling in Ansys CFX can be thought of as a multiphase flow in which the particles are a dispersed phase. Instead of using an Eulerian transport model for the dispersed phase, a Lagrangian transport model is used. All continuous phases must use the Eulerian model.

With the dispersed phase, each particle interacts with the fluid and other particles discretely. Therefore, another method is required to calculate the particle behavior. The most widely applied method available to determine the behavior of the dispersed phase is to track several individual particles through the flow field. Each particle represents a sample of particles that follow an identical path. The behavior of the tracked particles is used to describe the average behavior of the dispersed phase. This method is called *separated flow analysis*.

The separated flow analysis has been implemented as a Lagrangian tracking model to characterize the flow behavior of a dispersed phase and is available in Ansys CFX.

The term *particle* is used to describe an individual discrete element of the dispersed phase. The particle could represent a solid, droplet, or bubble.

Particle Number Rate

For applications involving tracking of discrete particles, it is not practical to track all physically existing particles. Instead representative particles, or parcels, are used to track these discrete particles. Each representative particle characterizes a certain number of actual particles. The actual number of particles represented by the representative particle is called the **Particle Number Rate**. The **Particle Number Rate** is determined from the mass flow rate assigned to the representative particle divided by the mass of an actual particle.

The known mass flow rate of particles can be specified on inlet, opening, wall and fluid-solid interface boundaries. This allows Ansys CFX to know how many particles the tracked particles represent.

All the particles represented by the sample particle will follow exactly the same track, even if the turbulent particle dispersion is used.

Initialization of the particle properties in the domain is not required.

8.1. Model Validity

Unlike in an Eulerian multiphase calculation, the fraction of volume taken by the particles is not included in the continuous phase calculation. This means that the model is only valid for quite low volume fractions. If the volume fraction of particles is calculated, then it can show the reliability of the model. Indeed, in cases where the model is unreliable, the calculated volume fraction of particles may be greater than one.

8.2. Particle Transport Versus Eulerian-Eulerian Multiphase

A multiphase flow containing dispersed particles may be modeled using either the particle transport model or the `Eulerian-Eulerian` multiphase model. Some advantages and disadvantages of the particle transport model are given below to aid in the choice of models. The equivalent section for the

Eulerian-Eulerian multiphase model is available in Eulerian-Eulerian Multiphase Versus Particle Transport (p. 299).

Table 8.1: Advantages and Disadvantages of Particle Transport

Advantages	Disadvantages
Complete information on behavior and residence time of individual particles is available.	Expensive if a large number of particles have to be tracked.
Relatively cheaper for wide range of particle sizes.	Very expensive to include turbulence.
Better detail for mass and heat transfer.	Essentially only possible as a post-process for a large number of particles.
More flexible when there is a significant size distribution leading to different particle velocities. (In Eulerian-Eulerian Multiphase, a momentum equation must be solved for each representative size which becomes very expensive.)	Restricted to low particle volume fractions.

8.3. Forces Acting on the Particles

Several different forces affect the motion of a particle in a fluid. In Ansys CFX, the forces which have been included in the particle equation of motion are viscous drag, buoyancy force, virtual mass and pressure gradient forces, and the centripetal and Coriolis forces in rotating reference frames. A description of these terms and their implementation is available.

8.3.1. Drag Force

The viscous and form drag is always important and is always calculated. For details, see Drag Force for Particles (p. 368).

8.3.2. Reference Frame Rotational Forces

These include the centripetal and Coriolis forces. They are always calculated for rotating frame simulations.

8.3.3. Buoyancy Force

In addition to the buoyancy forces that arise in single phase flows, the difference in density between the particle and the continuous phase results in a buoyancy force. This is the same as for Eulerian-Eulerian multiphase flows. For details, see Buoyancy in Multiphase Flow (p. 304).

8.4. Creating Particle Materials

If the particles you want to model are inert, the only material property required for particles is the density. Multicomponent particles can also be created: one application of these is to model liquid and oil evaporation. For details, see:

- Liquid Evaporation Model (p. 372)
- Liquid Evaporation Model: Oil Evaporation/Combustion (p. 374).

Another is for modeling coal combustion. For details, see Multiphase Reactions and Combustion (p. 414).

If modeling heat and mass transfer, the additional material properties that must be specified are specific heat capacity and reference enthalpy (although reference enthalpy is not required if latent heat is selected for the heat release method). For details, see Heat Release/Heat Release Distribution (p. 417).

You can create new materials from the **Material** details view in CFX-Pre. For details, see Materials and Reactions in the *CFX-Pre User's Guide*. Each material that you create will be available for selection from **Fluid and Particles Definitions...** on the **Basic Settings** tab when creating new domains. For details, see Basic Settings Tab in the *CFX-Pre User's Guide*.

8.5. Particle Domain Options

The model settings described in this section are set when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

8.5.1. Basic Settings

8.5.1.1. Particle Morphology Options

The morphology option can be set to `Particle Transport Fluid` or `Particle Transport Solid` in CFX-Pre. For details, see Basic Settings Tab in the *CFX-Pre User's Guide*. Gas and liquid particles should use the Fluid Particles option. Currently both models produce the same results because the same drag law is used for fluid and solid particles.

8.5.2. Fluid Models

8.5.2.1. Multiphase Reactions

Multiphase Reactions involving particle phases refer to a reaction between one or more components in a particle phase and one or more components in a continuous phase. For details, see Multiphase Reactions and Combustion (p. 414).

8.5.2.2. Buoyancy for Particles

Depending on the flow type, buoyancy (gravity) terms may need to be modeled. The density of the particle material is often much higher than that of the carrier fluid, resulting in a buoyancy force. However, other forces may dominate over the buoyancy force such that its effect is not important. The relative size of three forces must be considered:

- The drag force
- The inertial force
- The buoyancy force

Often the buoyancy force can be neglected if the drag and inertial forces dominate. In general, if settling of the particles is thought to be important, then the buoyancy term should be included.

If the buoyancy term is not important for the particle phase, you may still need to include it if convection due to density variations in the continuous phase is important. This is independent of particle transport modeling.

8.5.2.3. Turbulence for Particles

The turbulence model used in a particle tracking simulation only applies to the continuous phases. Turbulence can affect the particles through the Particle Dispersion force, but the particles can have no effect on the turbulence of the continuous phase, other than indirectly by affecting the velocity field. For details, see Turbulent Dispersion Force (p. 371).

8.5.2.4. Heat Transfer for Particles

Heat transfer between the particles and the continuous phase can be modeled. The **Fluid Models** form must use the `Fluid Dependent` option to model Heat Transfer. The heat transfer options for each phase are set on a per-material basis. For details, see Heat Transfer (p. 362).

8.5.2.5. Radiation for Particles

Radiation can be included in a particle transport calculation, but there is no radiation calculation in the particles themselves. To turn on radiation for particles, set the **Radiation** option to `Fluid Dependent`, and then select a radiation model on the **Fluid Specific Models** tab for the continuous fluid. The radiation option for particles themselves is unavailable on the **Fluid Specific Models** form (because it cannot assume any value other than `None`).

8.5.3. Fluid Specific Models

8.5.3.1. Particle Diameter Distribution

A particle diameter distribution can optionally be set for a domain. If specified, then the distribution applies to all boundaries where particles are injected in that domain. A particle diameter distribution can also be set in a boundary object, in which case it overrides the domain particle diameter distribution for that boundary only. If a domain particle diameter distribution is not set, then all boundaries where particles are injected must have a particle diameter distribution set. Note that for all selections of **Particle Diameter Distribution** (except for the **Specified Diameter** option), the larger the number of particles, the better the representation of the selected distribution.

8.5.3.1.1. Specified Diameter

This option sets a constant specified particle diameter for all particles.

8.5.3.1.2. Uniform in Diameter by Number

This option produces an equal number of particles at all diameters between the specified minimum and maximum diameters. This results in the same number of smaller particles as larger particles.

8.5.3.1.3. Uniform in Diameter by Mass

This option produces an equal mass of particles at all diameters between the specified minimum and maximum diameters. This results in a larger number of particles close to the minimum specified diameter, because many more of these are required to equal the mass of a few larger particles.

8.5.3.1.4. Normal in Diameter by Number

This option uses a normal distribution of particle diameters centered about a specified mean diameter. The shape of the normal distribution is determined by the specified standard deviation.

Maximum and minimum diameters are used to clip the normal distribution.

8.5.3.1.5. Normal in Diameter by Mass

Normal distribution requires `Mean Diameter`, `Standard Deviation in Diameter`, `Max Diameter` and `Min Diameter`.

8.5.3.1.6. Rosin Rammler

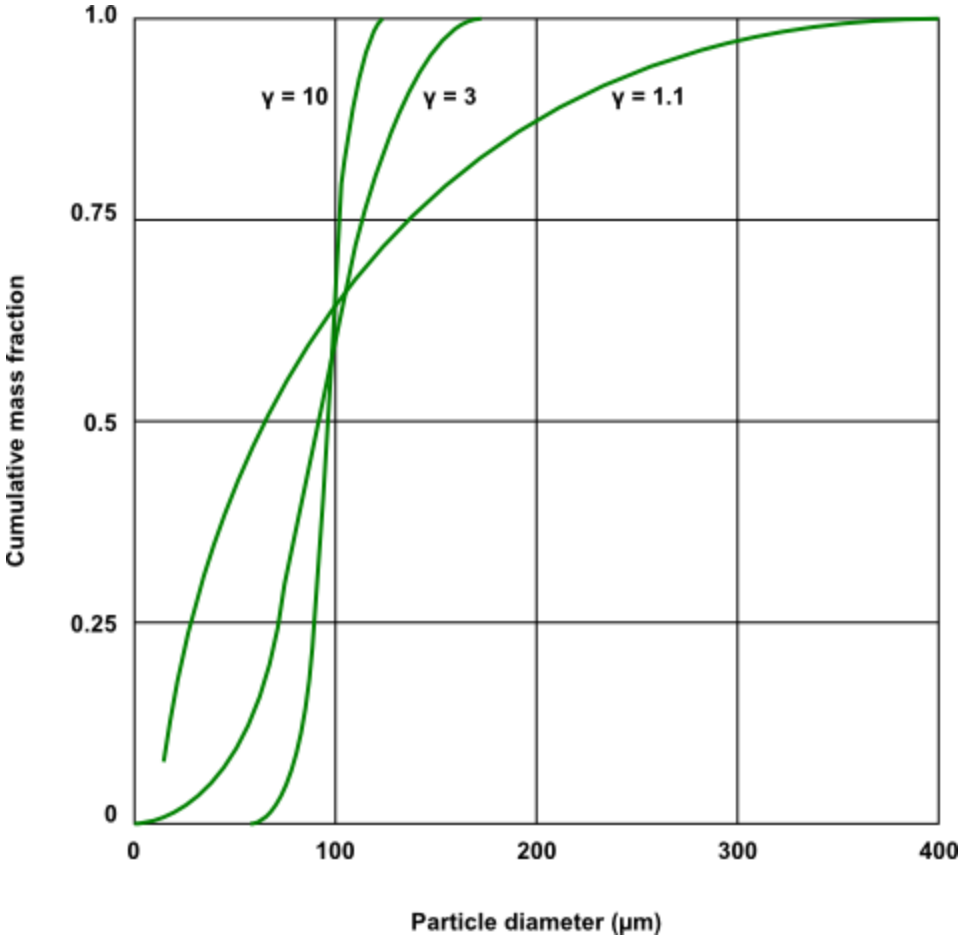
Requires a `Rosin Rammler Size` and `Rosin Rammler Power`. The Rosin Rammler distribution has been developed for pulverized solid fuels but may also be applicable to certain sprays.

A Rosin Rammler distribution can be used to determine the distribution of the mass flow among particle sizes. By default, the minimum particle diameter predicted by the Rosin Rammler distribution is clipped to 10% of the specified `Rosin Rammler Size`, and the maximum particle diameter is not clipped. However, the minimum and maximum diameters can be controlled by adding the `Minimum Diameter` and/or `Maximum Diameter` parameters into the `PARTICLE DIAMETER DISTRIBUTION` command language in the **Command Editor** dialog box. The mass fraction, R, above a given particle diameter, d, is calculated from:

$$R=\exp\left[-\left(\frac{d}{d_e}\right)^{\gamma}\right] \tag{8.1}$$

where d_e is a measure of the fineness and is equal to the diameter at which R is $1/e$ or 0.368. The spread parameter, γ, is a measure of dispersion of particle sizes, a lower value indicating a wider dispersion. A typical value of γ for pulverized fuels is 1 to 1.3 and for sprays is 1.5 to 3.0. For a nearly monosize distribution, γ may have a value of 10 to 20. Given d_e and γ, it is possible to calculate the particle size distribution. Examples of Rosin Rammler distributions for $d_e = 10^{-4}$ and various values of γ are shown in Figure 8.1: Particle Size Distributions for Various Rosin Rammler n Parameters (p. 359).

Figure 8.1: Particle Size Distributions for Various Rosin Rammler n Parameters



8.5.3.1.7. Nukiyama Tanasawa

Set the `Rosin Rammler Size`, `Rosin Rammler Power`, and the `Nukiyama Tanasawa Power`. This distribution is a generalization of the Rosin Rammler distribution. As stated for the Rosin Rammler distribution, by default, the minimum particle diameter predicted by the Nukiyama Tanasawa distribution is clipped to 10% of the specified `Rosin Rammler Size`, and the maximum particle diameter is not clipped. However, the minimum and maximum diameters can be controlled by adding the `Minimum Diameter` and/or `Maximum Diameter` parameters into the `PARTICLE DIAMETER DISTRIBUTION` command language in the **Command Editor** dialog box.

The distribution can be specified by its probability density function, $p(d)$, which is given by the following expression:

$$p(d) = c\, d^{q} \exp\left(-\left(\frac{d}{d_e}\right)^{\gamma}\right) \tag{8.2}$$

where the constant c is determined by the distribution integrating to 1.0. q is the Nukiyama Tanasawa power. If q is equal to $\gamma - 4$, then this reduces to the Rosin Rammler distribution. d_e and γ are defined as for the Rosin Rammler distribution.

8.5.3.1.8. Discrete Diameter Distribution

This option enables you to model particles with different diameters. You must provide three sets of discretized distribution data, as well as other parameters. This section explains, by example, which data are required and how they are used.

This example begins with the following input data:

```
Diameter List = 1 [mm], 2 [mm], 3 [mm], 4 [mm]
Mass Fraction List = 0.1, 0.4, 0.3, 0.2
Number Of Positions > Option = Direct Specification
Number Of Positions > Number = 20
Number Fraction List = 0.2, 0.3, 0.4, 0.1
The particle density is 1000 kg/m^3.
The total particle mass flow rate of injected particles is 1 g/s.
```

Diameter List is a list of diameter values. In this example, particles that belong to the first band have a diameter of 1 mm, particles that belong to the second band have a diameter of 2 mm, and so on.

Mass Fraction List is a list of mass fractions that apply to the total mass flow of particles at the injection site. The mass fractions must sum to unity. In this example, particles that belong to the first band constitute 10% of the total mass flow of injected particles, particles in the second band constitute 40% of the mass flow of injected particles, and so on.

Physical particles are not modeled individually. Instead, a smaller number of representative (or "computational") particles are modeled. The representative particles in a given band each represent an equal share of the physical particles in that band. The total number (or in a transient case, the rate of production) of representative particles is governed by the **Number Of Positions** specification. The number of representative particles that are generated for a particular band depends on a probability distribution specified by **Number Fraction List** (a list of values that sum to unity).

In this example, 20 representative particles would be generated. Each representative particle would be assigned to one of the 4 bands by using a random number generator. Roughly 4 (or 20%) of the representative particles will be assigned to the first band, roughly 6 (or 30%) will be assigned to the second band, and so on.

After the solver has assigned all representative particles to bands, it then assigns to each representative particle:

- a diameter from the entry in **Diameter List** that corresponds with the representative particle's band, and
- a number rate of physical particles to be represented by the representative particle

The number rate is calculated for each band as follows:

mass flow rate per band / (mass of a physical particle * number of representative particles in the band)

In this case, the number rates for representative particles in bands 1 and 2 would be calculated as follows:

- Band 1

The mass flow rate of all particles in the band is 0.1 * 0.001 kg/s = 0.0001 kg/s.

Each physical particle has a mass of (4/3) * pi * ((0.001 m)/2)^3 * 1000 kg/m^-3 = 0.524E-6 kg.

Assume that 4 representative particles were assigned to the band.

Number Rate = (0.0001 kg/s) / ((0.524E-6 kg/particle) * 4) = 48 particles/s.

- Band 2

  The mass flow rate of all particles in the band is 0.4 * 0.001 kg/s = 0.0004 kg/s.

  Each physical particle has a mass of (4/3) * pi * ((0.002 m)/2)^3 * 1000 kg/m^-3 = 4.189E-6 kg.

  Assume that 6 representative particles were assigned to the band.

  Number Rate = (0.0004 kg/s) / ((4.189E-6 kg/particle) * 6) = 16 particles/s.

Note:

If you edit CCL directly, do not enter `Number Fraction List` and `Mass Fraction List` entries using the "*" operator. For example:

Not allowed in CCL:

```
Number Fraction List = 5 * 0.1, 0.5
```

Correct syntax:

```
Number Fraction List = 0.1, 0.1, 0.1, 0.1, 0.1, 0.5
```

This requirement applies to CCL you edit directly. If you create such CCL using CFX-Pre, the CCL will be converted automatically to the correct form.

To ensure that each band receives at least one representative particle, ensure that you specify a sufficient number of representative particles. As a minimum requirement, you must inject at least as many representative particles as there are bands. To get a rough estimate for the required number, take the inverse of the smallest number fraction in the list. The solver will tolerate an empty band if the particle mass flow for the band is specified as being less than 1% of the total mass flow, but you should try to prevent this from happening. The solver will stop during injection if a band representing more than 1% of the total injected mass did not receive any representative particles.

The diameter distribution can be specified only once on a domain level; it will be applied to each injection region separately. The total number of injected particles must guarantee a reasonable number of representative particles per injection region. For simulations with many injection regions (for example, gas washers), but with only a small number of particles per injection region, the problem of empty bands can easily happen. Your options are to increase the number of representative particles, or to use Particle User Fortran.

8.5.3.2. Particle Shape Factors

Particles are assumed to be spherical by default. Ansys CFX always calculates the diameter of the particle from the mass of the particle divided by its density, assuming it is spherical.

A **Cross Sectional Area Factor** can be included to modify the assumed spherical cross-section area to enable non-spherical particles. The factor is multiplied by the cross-section area calculated assuming spherical particles. This affects the drag force and radiation interaction calculated by CFX.

The **Surface Area Factor** is analogous to the cross-sectional area factor. For a non-spherical particle, it is the ratio of the surface area to the surface area of a spherical particle with the same equivalent diameter. This factor affects both mass transfer and heat transfer correlations.

8.5.3.3. Particle Diameter Change Due to Swelling

In contrast to liquid droplets, where the diameter of the particle is determined by the mass and densities of species, the diameter of a solid particle remains relatively constant, as the density of the particle as a whole changes. The change in diameter of the particle is controlled by the mass of solid particle, and by the swelling factor. **Swelling Factor** is dimensionless and should be set with reference to the raw material, by selecting it from the **Reference Material** list in CFX-Pre. For details, see Devolatilization in the *CFX-Solver Theory Guide*.

8.5.3.4. Heat Transfer

The heat transfer model for the continuous phase is set in the same way as for single-phase simulations. When a heat transfer model is activated for the continuous fluid, the heat equation for the particles is solved by default (the **Heat Transfer** option is set to `Particle Temperature` for particles). Information on the implementation of heat transfer for particle transport in Ansys CFX is available in Heat Transfer in the *CFX-Solver Theory Guide*.

8.5.3.5. Erosion

The erosion model can be set on a per-boundary or per-domain basis. When selected for the domain, the domain settings will apply for all boundaries that do not explicitly have erosion model settings applied to them.

There is a choice of two erosion models, those of `Finnie` and `Tabakoff`. Additional information on the implementation of these models is available in Basic Erosion Models in the *CFX-Solver Theory Guide*. The choice of one model over another is largely simulation-dependent. In general, the Tabakoff model provides more scope for customization with its larger number of input parameters.

In the `<install_dir>\examples\UserFortran` directory, you can find an erosion routine, `pt_erosion.F`, and corresponding CCL template, `pt_erosion.ccl`.

8.5.3.5.1. Finnie

The velocity power factor corresponds to the variable n. It is set to 2 by default. The value of the exponent generally lies within the range 2.3 to 2.5 for metals.

The Reference Velocity corresponds to the variable V_0 in Equation 6.92 in the *CFX-Solver Theory Guide*.

For details, see Model of Finnie in the *CFX-Solver Theory Guide*.

8.5.3.5.2. Tabakoff

The Tabakoff model requires the specification of five parameters. The k_{12} constant, 3 reference velocities and the angle of maximum erosion γ_0 must all be specified.

Table 8.2: Coefficients for Some Materials Using the Tabakoff Erosion Model (p. 363) lists some values for the Tabakoff model coefficients for Quartz-Aluminum, Quartz-Steel and Coal-Steel. For details, see Model of Tabakoff and Grant in the *CFX-Solver Theory Guide*.

Table 8.2: Coefficients for Some Materials Using the Tabakoff Erosion Model

Variable	Coefficient	Value	Material
k_{12}	k_{12}	5.85 x 10^{-1}	Quartz - Aluminum
Ref Velocity 1	V_1	159.11[m/s]	
Ref Velocity 2	V_3	194.75 [m/s]	
Ref Velocity 3	V_4	190.5 [m/s]	
Angle of Maximum Erosion	γ_0	25 [deg]	
k_{12}	k_{12}	2.93328 x 10^{-1}	Quartz - Steel
Ref Velocity 1	V_1	123.72 [m/s]	
Ref Velocity 2	V_3	352.99 [m/s]	
Ref Velocity 3	V_4	179.29 [m/s]	
Angle of Maximum Erosion	γ_0	30 [deg]	
k_{12}	k_{12}	-1.321448 x 10^{-1}	Coal - Steel
Ref Velocity 1	V_1	51.347 [m/s]	
Ref Velocity 2	V_3	87.57 [m/s]	
Ref Velocity 3	V_4	39.62 [m/s]	
Angle of Maximum Erosion	γ_0	25 [deg]	

8.5.3.5.3. User Defined

This option allows you to use a User Fortran subroutine to calculate the erosion rate density. General information on creating user routines is available in User Fortran (p. 631). Input arguments and return quantities of the particle user routine are specified in the **Wall Interaction** option. For details, see Wall Interaction (p. 383).

In the `<install_dir>\examples\UserFortran` directory, you can find an erosion routine, `pt_erosion.F`, and corresponding CCL template, `pt_erosion.ccl`.

8.5.3.6. Particle-Rough Wall Model (Virtual Wall Model)

The particle-rough wall model can be set on a per-boundary or per-domain basis. When selected for the domain, the domain settings will apply for all boundaries that do not explicitly have rough wall model settings applied to them.

Information on the model theory and its implementation is available in The Sommerfeld-Frank Rough Wall Model (Particle Rough Wall Model) in the *CFX-Solver Theory Guide*.

8.5.3.6.1. Sommerfeld-Frank Model

The Sommerfeld-Frank model requires the input of parameters that describe the wall roughness structure. The input parameters are:

- The average length of a roughness element, L_r
- The average height of a roughness element, H_r
- The deviation from the average roughness height, ΔH_r

For details, see The Sommerfeld-Frank Rough Wall Model (Particle Rough Wall Model) in the *CFX-Solver Theory Guide*.

Note:

- These parameters may change in the future.
- A spatial variation of these quantities on a single boundary patch is not supported.

8.5.3.7. Particle Breakup Model

The particle breakup models enable you to simulate the breakup of droplets due to external aerodynamic forces. The droplet breakup models are set on a per fluid-pair basis. Note that Liu dynamic drag coefficient modification is activated by default for the TAB, ETAB, and CAB breakup models, whereas the Schmehl dynamic drag law is available only for the Schmehl breakup model.

There is a choice of five predefined secondary particle breakup models, those of `Reitz and Diwakar`, `Schmehl`, TAB, ETAB, and CAB as well as the possibility to define your own model via Particle User Fortran.

In the `<install_dir>\examples\UserFortran` directory, you can find a secondary breakup routine, `pt_breakup_rd.F`, and corresponding CCL template, `pt_breakup_rd.ccl`.

The choice of one model over another is largely simulation-dependent. All model constants are accessible via the user interface. For details on the models, see Spray Breakup Models in the *CFX-Solver Theory Guide*. Default values for each model are given in the following places:

- Table 6.1: Reitz and Diwakar breakup model constants and their default values
- Table 6.2: TAB Breakup Model Constants and their Default Values
- Table 6.3: ETAB Breakup Model Constants and Their Default Values
- Table 6.4: CAB Breakup Model Constants and Their Default Values

During droplet breakup, the particle shape may be distorted significantly and it might be desirable to modify the drag law to account for the influence of the particle distortion. Different models have been implemented in Ansys CFX that enable you to modify the drag coefficient based on the droplet distortion. See Dynamic Drag Models in the *CFX-Solver Theory Guide* for the description of available models and their usage.

8.5.3.8. Particle Collision Model

In classical Euler-Lagrange modeling, collisions between particles are not possible because the presence of other particles is not accounted for. The stochastic particle-particle collision model by Oesterlé & Petitjean [151], which has been extended by Frank [149], Hussmann et al. [150], and Sommerfeld [152], takes interparticle collisions into consideration while the trajectories are still calculated sequentially. The main advantage of the model is the possibility of sequential trajectory calculation by creating virtual collision partners sampled from local statistical values. This offers a high potential of parallelization and therefore facilitates – in conjunction with the highly parallelized CFX-Solver for the gas-phase – its use in industrial applications as a major advancement in the simulation of dense gas-particle flows. The model extension by Sommerfeld additionally takes into account a possible correlation between the velocity fluctuations of neighboring particles. The implementation in Ansys CFX was validated based on three published experiments: one involving a convergent channel provoking inter-particle collisions, one involving a highly loaded vertical pipe flow, and one involving a strongly swirling pipe flow.

The following steps summarize the implementation of this model in Ansys CFX:

1. Calculate the instantaneous velocity of the virtual collision partner.

   The velocity of a virtual particle has the following components:

   - A mean part from the local average values
   - Correlation function is determined by Large Eddy Simulation (LES) of a homogeneous isotropic turbulence field (see Equation 6.172 in the *CFX-Solver Theory Guide*)
   - A fluctuating part including a correlation term between the two particles due to Sommerfeld [152, 154] and a random term (see Equation 6.173 in the *CFX-Solver Theory Guide*)

   Angular velocity of the particle is calculated the same way:

- no correlation between particles

2. Determine the collision probability and decide whether or not a collision occurs.
   - *If collision occurs:*
     a. Compute position of virtual collision partner.
     b. Calculate binary collision and adjust particle velocities.
   - *If collision does not occur:*
     a. Particle velocities remain unchanged in the event of no collision.
3. Discard the virtual particle.
4. Update average particle quantities and local statistical moments in each control volume.

In the `<install_dir>\examples\UserFortran` directory, you can find a collision routine, `pt_collision.F`, and corresponding CCL template, `pt_collision.ccl`.

8.5.3.8.1. Requirements for the Applicability of Particle-Particle Collision Model

The requirements for the applicability of particle collision model are outlined below:

- High mass loading
- Moderate volumetric concentration (<~ 20%)
- The model is limited to binary collisions only and follows the following criteria:
  - Inter-particle distance is much greater than the particle diameter
  - Aerodynamic forces dominate
  - Not suitable for fluidized beds
  - $\rho_p \gg \rho_{Gas}$
- Spherical particles

For additional information, see the following topics available under Particle Collision Model in the *CFX-Solver Theory Guide*:

- Introduction to the Particle Collision Model
- Implementation of a Stochastic Particle-Particle Collision Model in Ansys CFX (includes the discussion on the implementation theory, particle variables, and virtual collision partner)
- Range of Applicability of Particle-Particle Collision Model
- Limitations of Particle-Particle Collision Model in Ansys CFX

8.5.3.9. Fluid Buoyancy Model

If you have selected the flow as buoyant and provided components for the gravity vector, then a **Fluid Buoyancy Model** can be set for each phase. The same treatment as `Eulerian-Eulerian` multiphase flow applies. For details, see Buoyancy in Multiphase Flow (p. 304).

Setting the flow as buoyant, but each fluid buoyancy model to non-buoyant is equivalent to setting the flow as non-buoyant.

8.5.4. Fluid Pairs

The models described in this section are selected on the **Fluid Pairs** tab when creating a domain containing particles in CFX-Pre. For details, see Fluid Pair Models Tab in the *CFX-Pre User's Guide*.

8.5.4.1. Particle Fluid Pair Coupling Options

Particles can be either fully coupled to the continuous fluid or can be one-way coupled. Fully coupled particles exchange momentum with the continuous phase, enabling the continuous flow to affect the particles, and the particles to affect the continuous flow.

Full coupling is needed to predict the effect of the particles on the continuous phase flow field but has a higher CPU cost than one-way coupling. One-way coupling simply predicts the particle paths as a post-process based on the flow field and therefore it does not influence the continuous phase flow field. For details, see Interphase Transfer Through Source Terms in the *CFX-Solver Theory Guide*.

The choice of one-way or full coupling for particles depends on the mass loading, that is, the ratio of the mass flow rate of particles to the mass flow rate of fluid. One-way coupling may be an acceptable approximation in flows with low mass loadings where particles have a negligible influence on the fluid flow. If the particles influence the fluid flow significantly, then you should use full coupling.

To optimize CPU usage, you can create two sets of identical particles. The first smaller set should be fully coupled and is used to enable the particles to influence the flow field. The second larger set should use one-way coupling and provides a more accurate calculation of the particle volume fraction as well as local forces on walls. When postprocessing these types of cases, you should *not*, for example, sum the forces on the wall from both sets of particles because each set fully represents all the particles.

The CPU cost of tracking particles is proportional to the number of particles tracked multiplied by the number of times tracked. One-way coupled particles are tracked only once, at the end of the solver run. The number of times fully coupled particles are tracked depends on the iteration frequency set on the **Solver Control** tab and the number of iterations required for the simulation to converge.

You can define multiple sets of one-way coupled particles without affecting the flow field. For example, if you were conducting a parametric study with various different particle sizes, you can create multiple particle materials with the same properties and then use each one to define a set of particles with different diameters. This is not true of fully coupled particles, because each set influences the flow field.

8.5.4.2. Drag Force for Particles

There are three ways in which the drag forces between the continuous phase and the particle phase can be modeled:

- Use the Schiller-Naumann, Ishii-Zuber, or Grace correlations.
- Use Particle Transport Drag Coefficient and specify the drag coefficient using one of the following options:
  - Drag Coefficient - specify a constant value (CEL expressions are not permitted)
  - User Defined - specify a drag correlation using a particle user routine. See Particle User Source Example (p. 368) for an example of how to do this.
- Set the drag to `None` and set your own drag force using a particle user routine.

A description of these particle models is available in Interphase Drag for the Particle Model (p. 305), along with models for Euler-Euler flows.

8.5.4.2.1. Particle User Source Example

This example is a basic routine used to calculate the particle drag coefficient using the Wen Yu Drag Model. The Schiller-Naumann correlation is used with a modified particle Reynolds number and a power law correction, both functions of the continuous phase volume fraction. The input arguments to the routine are Particle Reynolds number and volume fraction of the particle phase. The Fortran subroutine returns the particle drag coefficient, C_d. You can copy the routine (`pt_drag_factor.F`) from the `<install_dir>\examples\UserFortran` directory.

The Fortran for this example is:

```
#include "cfx5ext.h"
dllexport(pt_drag_factor)
      SUBROUTINE PT_DRAG_FACTOR (NLOC,NRET,NARG,RET,ARG,CRESLT,
     &                           CZ,DZ,IZ,LZ,RZ)
CC
CC --------------------
CC        Input
CC --------------------
CC
CC  NRET   - number of components in result
CC  NARG   - number of arguments in call
CC  ARG() - (NARG) argument values
CC
CC --------------------
CC      Modified
CC --------------------
CC
CC  Stacks possibly.
CC
CC --------------------
CC        Output
CC --------------------
CC
CC  RET()  - (NRET) return values
CC
CC --------------------
CC       Details
CC --------------------
CC
CC=====================================================================
```

```
C
C ------------------------------
C     Preprocessor includes
C ------------------------------
C
C ------------------------------
C        Global Parameters
C ------------------------------
C
C ------------------------------
C        Argument list
C ------------------------------
C
      INTEGER NLOC,NARG,NRET
C
      REAL ARG(NLOC,NARG), RET(NLOC,NRET)
C
      CHARACTER CRESLT*(*)
C
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C ------------------------------
C        External routines
C ------------------------------
C
C ------------------------------
C        Local Parameters
C ------------------------------
C
C ------------------------------
C        Local Variables
C ------------------------------
C
C ------------------------------
C        Stack pointers
C ------------------------------
C
C=======================================================================
C
C --------------------------
C    Executable Statements
C --------------------------
C
C=======================================================================
C
C     Argument variables stack:
C     -------------------------
C     Reynolds number          :     RE_PT      = ARG(1,1)
C     Particle volume fraction :     VOLFRN_PT  = ARG(1,2)
C
C     Return variables stack:
C     -----------------------
C     Drag coefficient         :     CD         = RET(1,1)
C
C=======================================================================
C
C-----------------------------------------------------------------------
C     Calculate the momentum source and source term coefficient
C-----------------------------------------------------------------------
C
      CALL USER_CD(RET(1,1),ARG(1,1),ARG(1,2))
      END
C
      SUBROUTINE USER_CD(CD,RE_PT,VOLFRN_PT)
C
C=======================================================================
C     Calculate the momentum source and source term coefficient
```

```
C=======================================================================
C
C ----------------------------
C     Preprocessor includes
C ----------------------------
C
#include "cfd_sysdep.h"
#include "cfd_constants.h"
C
C ----------------------------
C        Argument list
C ----------------------------
C
      REAL CD, RE_PT, VOLFRN_PT
C
C ----------------------------
C        Local variables
C ----------------------------
C
      REAL VOLFRN_C, RE_S
C
C ----------------------------
C    Executable statements
C ----------------------------
C
C---- Clip continuous phase volume fraction [0,1]
C
      VOLFRN_C  = 1. - MAX(0.0,MIN(1.0,VOLFRN_PT))
C
C---- Correct Reynolds number
C
      RE_S = RE_PT*VOLFRN_C
C
C---- Calculate the Schiller-Naumann drag coefficient
C
      IF (RE_S .LT. 1000.) THEN
        CD = 24./RE_S*(1+0.15*RE_S**0.687)
      ELSE
        CD = 0.44
      ENDIF
C
C---- Correct drag coefficient to account for high particle volume
C     loading
C
      CD = VOLFRN_C**-1.65*CD
C
      END
```

8.5.4.2.2. Linearization Blend Factor

The linearization blend factor is a value that interpolates between simple linearization and the linearization based upon accurate differentiation of the drag. The recommended value (and the default) is 0.

8.5.4.3. Particle User Source

For details, see Particle User Sources (p. 374).

8.5.4.4. Non-Drag Forces

8.5.4.4.1. Virtual Mass Force

Virtual mass force can be modeled with the specification of a virtual mass coefficient. The virtual mass force is proportional to the continuous phase density, hence, is most significant when the dispersed phase density is less than the continuous phase density. Also, by its nature, it is only significant in the presence of large accelerations, for example, in transient flows, and in flows through narrow restrictions. Additional information on the implementation and usage for this model is available in Virtual Mass Force (p. 309). For particle transport, the virtual mass coefficient defaults to 0.5.

8.5.4.4.2. Turbulent Dispersion Force

Turbulent dispersion forces result in additional dispersion of particles from high volume fraction regions to low volume fraction regions due to turbulent fluctuations. The `Particle Dispersion` model is available to account for the turbulent dispersion force. This force is only important for small particles (approximately smaller than 100 microns for water drops in air) and when you want to see the dispersion. For example, even when the particle tracks are affected by turbulence, the effect of the particles on the continuous phase is usually the important process, and this is not affected by the turbulence.

The turbulent dispersion force is only active in regions where the turbulent viscosity ratio is above the value specified by **Eddy Viscosity Ratio Limit**. The default value is 5.

Note:

Turbulent dispersion can only be used if a drag force is specified. Therefore, it is not possible to combine turbulent dispersion with a user-specified momentum source term for the drag.

8.5.4.4.3. Pressure Gradient Force

The pressure gradient force is small for particles of much higher density than the continuum fluid and need not be included when this is the case. For details, see Pressure Gradient Force in the *CFX-Solver Theory Guide*.

8.5.4.5. Interphase Heat Transfer

The only available heat transfer option between the continuous phase and the particles is `Ranz Marshall`. The Ranz Marshall correlation is also applicable to Eulerian multiphase flow. For details, see Particle Model Correlations for Overall Heat Transfer Coefficient (p. 313).

8.5.4.5.1. Particle User Source

For details, see Particle User Sources (p. 374).

8.5.4.6. Interphase Radiation Transfer

8.5.4.6.1. Opaque

If `Opaque` is chosen, the radiation transfer can be either one-way coupled or fully coupled (if the overall coupling is fully coupled). For details, see:

- Opaque (p. 467)
- Emissivity (p. 475).

8.5.4.6.2. Blended Particle Emissivity

When a reacting particle case has been defined, two parameters exist to account for the change as the reacting particle changes its composition. For details, see Multiphase Reactions and Combustion (p. 414). Base Emissivity and Blended Emissivity must both be supplied. The present model assumes a linear variation in the emissivity from the reference material (for example, raw coal) value to the value of the reaction product (for example, char). For details, see Radiative Preheating in the *CFX-Solver Theory Guide*.

8.5.4.7. Mass Transfer

There are a number of choices available for mass transfer in Ansys CFX. The choices available for mass transfer depend on the type of particles that have been selected for the simulation. For any particle, simple mass transfer can be modeled with the `Ranz Marshall` model. Mass transfer from liquid drops can be modeled with the `Liquid Evaporation Model`. Coal combustion also uses mass transfer, but is set up using multiphase reactions, so no component pair information should be set at this point. For details, see Multiphase Reactions and Combustion (p. 414).

Important:

Regardless of the mass transfer option chosen, whenever mass transfer takes place, the particle diameter is automatically changed to reflect the change in mass.

8.5.4.7.1. Ranz Marshall

This option is analogous to Additional Variable species transfer, and uses the Ranz Marshall correlation on the continuous phase side of the interface and a zero resistance on the particle phase side of the phase interface. For details, see Ranz-Marshall Correlation (p. 321). A mass fraction equilibrium ratio can be set, analogous to that described in the multiphase mass transfer section. For details, see Equilibrium Models in the *CFX-Solver Theory Guide*.

8.5.4.7.2. Liquid Evaporation Model

This option is used to model evaporation of a liquid species in particles to the respective gas phase species in the continuous phase. This model is designed for evaporation of a single species and commonly used for spray dryer and oil combustion applications. However, it can be extended to multi-component evaporation. For details, see Extension of the Liquid Evaporation Model in the *CFX-Solver Theory Guide*. A template CCL outline has been provided for general liquid evaporation. The `evaporating_drops.ccl` file (located in the `etc/model-templates/` directory of your Ansys CFX installation) can be selected when starting a new simulation using the Library

User mode. The template sets up a continuous gas phase containing H2O and Air Ideal Gas, and defines a homogeneous binary mixture for the phase change of Water at 25 C to H2O. It also creates and specifies a domain named Domain 1 that includes the continuous and particle phases. Templates have also been provided for the Spray Dryer and Oil Combustion. For details, see:

- Liquid Evaporation Model: Spray Dryer with Droplets Containing a Solid Substrate (p. 374)
- Liquid Evaporation Model: Oil Evaporation/Combustion (p. 374).

To complete the domain specification, you would need to create a material to define the particle phase (step 1, below) and import an appropriate mesh.

Should you choose not to use the template, a general outline of the steps to set up a liquid evaporation model, including Water at 25 C evaporating from a particle into a gas mixture of H2O and Air Ideal Gas follows. You can substitute the materials below for the materials you are modeling in your simulation.

1. Create a pure substance or variable composition mixture that will be the particle phase. In the case of a variable composition mixture, it should contain the solid material(s) from which water evaporates, and the evaporating liquid (Water at 25 C).

2. Create another variable composition mixture that will be the continuous fluid. This mixture must include the gas phase component of the evaporating liquid selected in step 1 (H2O from the Gas Phase Combustion material group), as well as a constraint material (Air Ideal Gas).

3. Create a homogeneous binary mixture to set the mass transfer properties of the evaporating material. For the case of water, the homogeneous binary mixture contains Water at 25 C and H2O. The rate of evaporation in the liquid evaporation model is controlled by the binary mixture settings defined in the **Material** details view. Information on the creation of homogeneous binary mixtures is available in Material Details View: Homogeneous Binary Mixture in the *CFX-Pre User's Guide*.

4. When creating the domain, define a fluid (or select an existing fluid) and set the **Morphology** > Option to Continuous Fluid.

5. Define a particle and set the **Morphology** > Option to Particle Transport Fluid.

6. On the **Fluid Models** form, set **Heat Transfer Model** to Fluid Dependent.

7. When setting up the **Fluid Specific Models** for the continuous mixture set H2O to use a Transport Equation and Air Ideal Gas to Constraint. Set **Heat Transfer** to use Thermal Energy for the continuous phase, and Particle Temperature for the particle phase.

8. On the **Fluid Pairs** tab, set **Heat Transfer** to Ranz Marshall (to enable interphase heat transfer). Under **Component Pairs**, enable the check mark for **H2O | Water at 25Ç** and set **Option** to Liquid Evaporation Model.

9. The Latent Heat setting for **Latent Heat** can be selected if the reference enthalpies of the liquid and gas materials are inconsistent.

8.5.4.7.3. Liquid Evaporation Model: Spray Dryer with Droplets Containing a Solid Substrate

The evaporation of liquid from droplets containing a solid substrate is a common industrial application. The `spray_dryer.ccl` template is the same as the general liquid evaporation model, but includes the particle phase `Coffee Mixture`, which includes a new material `Coffee` and `Water at 25 C`. For details, see Liquid Evaporation Model (p. 372).

8.5.4.7.4. Liquid Evaporation Model: Oil Evaporation/Combustion

The simulation of evaporating oil droplets is set up in a very similar way to the liquid evaporation model, with a **Modification** parameter named `Light Oil`, which is enabled to control the physical properties used in calculating the heat and mass transfer from the droplet. For details, see Liquid Evaporation Model (p. 372). This parameter should be set for all oil evaporation simulations. Enabling combustion involves creating a reaction for the gas phase mixture and then defining the gas phase mixture as a reacting mixture when setting up its material properties. For details, see Material Details View: Reacting Mixture in the *CFX-Pre User's Guide*.

The `oil_combustion.ccl` template contains the material `JetA Liquid` (as the liquid oil) as a pure substance particle that evaporates into its gas phase equivalent `JetA`. `JetA` is part of the gas phase variable composition mixture named `Gas Mixture`. The `Gas Mixture` material itself is defined using a single-step reaction `JetA Oxygen WD1` (a reaction that specifies `JetA` and oxygen as reactants, and carbon dioxide and water as products, with an additional non-reacting `N2` component). A homogeneous binary mixture, `JetAlg`, exists to model the evaporation of `JetA Liquid` to `JetA`.

A domain named `Domain 1` is set up in a very similar way to a general liquid evaporation specification, with the `Light Oil Modification` set for the mass transfer of `JetA Liquid` to `JetA`. Additional information on the `Light Oil` treatment is available in Light Oil Modification in the *CFX-Solver Theory Guide*.

8.5.4.7.5. Latent Heat

In general, you can select `From Material Properties` for the **Latent Heat** option. The reference enthalpies will be used to determine latent heat if this option is selected. You can optionally set a latent heat by entering a value (for example, if the reference enthalpies of the liquid and gas materials are not consistent).

8.5.4.8. Particle User Sources

The **Particle User Source** functionality allows you to use a User Fortran subroutine to calculate sources of momentum, heat and mass transfer. General information on creating user routines is available in User Fortran (p. 631).

In the `<install_dir>\examples\UserFortran` directory, you can find routines and CCL templates to calculate sources for momentum (`pt_mom_source.F`, `pt_mom_source.ccl`), heat (`pt_heat_source.F`, `pt_heat_source.ccl`), and mass transfer (`pt_mass_source.F`, `pt_mass_source.ccl`). These routines can be used as starting points for your own modifications. A particle user routine example is presented later in this section.

To use a particle user routine, you should first write the Fortran routine and create a shared library. Next, you should create a User Routine (**Insert** > **User Routine**) of type `Particle User Routine`

in CFX-Pre. When defining the domain, particle user sources are selected by selecting the **Particle User Source** check box for the appropriate quantity, and selecting the routine from the drop-down list. For details, see:

- Creating the Shared Libraries (p. 640)
- Particle User Routines in the *CFX-Pre User's Guide*.

The **Argument Variables** list should contain the fluid and particle variables needed to calculate the source (help on how to use variable names in Ansys CFX, as well as a list of all variables, can be found in the VARIABLES file, in the `/etc/` directory of your Ansys CFX installation).

The **Return Variables** list can only contain four quantities: the source itself, its derivative with respect to the associated variable in the particle, its derivative with respect to the associated variable in the fluid, and for mass sources only its derivative with respect to the temperature in the particle. The latter three quantities are optional, and the derivative with respect to the associated variable in the fluid can only be used when the overall coupling and the coupling set for the user source are both set to Fully Coupled. For momentum sources the associated variable is velocity, for heat transfer the associated variable is temperature, and for mass transfer the associated variable is mass fraction of species. Additional information on general source specification is available in Sources (p. 80) in Basic Capabilities Modeling (p. 39).

8.5.4.8.1. Particle User Source example

Important:

This example is a basic routine used to calculate a Schiller-Naumann drag force and is intended for demonstration purposes. If you want to use this example, you should set the **Drag Force** option to `None` on the **Fluid Pairs** tab.

This example uses a subroutine to calculate a user drag force between a particle phase `Red Sand`, and the fluid `AirSTP`. The Fortran subroutine returns a source and a source coefficient with respect to the particle and fluid. The input arguments to the routine are:

- Mean particle diameter, velocity, particle Reynolds number and slip velocity for the particle `Red Sand`.
- Density and velocity for the fluid `AirSTP`.

You can copy the routine (`pt_mom_source.F`) from the `<install_dir>\examples\User Fortran` directory. The Fortran for this example is:

```
#include "cfx5ext.h"
dllexport(pt_momsource)
      SUBROUTINE PT_MOM_SOURCE (NLOC, NRET, NARG, RET, ARG, CRESLT,
      &    CZ,DZ,IZ,LZ,RZ)
CC
CD User routine: template for particle user routine
CC
CC --------------------
CC        Input
CC --------------------
CC
CC  NRET   - number of components in result
CC  NARG   - number of arguments in call
CC  ARG() - (NARG) argument values
```

```
CC  CRESLT - Result
CC
CC --------------------
CC        Output
CC --------------------
CC
CC  RET()  - (NRET) return values
CC
CC --------------------
CC       Details
CC --------------------
CC
CC=====================================================================
C
C ----------------------------
C        Argument list
C ----------------------------
C
         INTEGER NLOC,NARG,NRET
C
         REAL ARG(NLOC,NARG), RET(NLOC,NRET)
C
         CHARACTER CRESLT*(*)
C
         INTEGER IZ(*)
         CHARACTER CZ(*)*(1)
         DOUBLE PRECISION DZ(*)
         LOGICAL LZ(*)
         REAL RZ(*)
C
C ----------------------------
C        Local Variables
C ----------------------------
C
C=====================================================================
C
C --------------------------
C    Executable Statements
C --------------------------
C
C=====================================================================
C
C     Argument variables stack:
C     -------------------------
C
C     Particle diameter   :     DIAM_PT    = ARG(1,1)
C     Particle velocity   :     VEL_PT     = ARG(1,2:4)
C     Reynolds number     :     RE_PT      = ARG(1,5)
C     Slip velocity       :     SLPVEL_PT  = ARG(1,6)
C     Fluid density       :     DENSITY_FL = ARG(1,7)
C     Fluid velocity      :     VEL_FL     = ARG(1,8:10)
C
C     Return variables stack:
C     -----------------------
C
C     Source term         :     SOURCE     = RET(1,1:3)
C     Source coefficient  :     COEF_PT    = RET(1,4)
C     Source coefficient  :     COEF_FL    = RET(1,5)
C
C=====================================================================
C
C---------------------------------------------------------------------
C     Calculate the momentum source and source term coefficient
C---------------------------------------------------------------------
C
         CALL USER_MOMENTUM_SOURCE (RET(1,1),RET(1,4),RET(1,5),
     &    ARG(1,1),ARG(1,2),ARG(1,5),ARG(1,6),
     &    ARG(1,7),ARG(1,8))
C
      END
      SUBROUTINE USER_MOMENTUM_SOURCE (SOURCE,COEF_PT,COEF_FL,
```

```
     &                                DIAM_PT,VEL_PT,RE_PT,SLPVEL_PT,
     &                                DENSITY_FL,VEL_FL)
C
C=======================================================================
C     Calculate the momentum source and source term coefficient
C=======================================================================
C
C ----------------------------
C     Preprocessor includes
C ----------------------------
C
#include "cfd_sysdep.h"
#include "cfd_constants.h"
C
C ----------------------------
C        Argument list
C ----------------------------
C
         REAL SOURCE(3), COEF_PT, COEF_FL,
     &    DIAM_PT, VEL_PT(3), RE_PT, SLPVEL_PT,
     &    DENSITY_FL, VEL_FL(3)
C
C ----------------------------
C        Local variables
C ----------------------------
C
         REAL AREA, CD, FACT
C
C ----------------------------
C    Executable statements
C ----------------------------
C
C---- Calculate the particle reference area
C
         AREA = PI*DIAM_PT**2*QUARTER
C
C---- Calculate the Schiller-Naumann drag coefficient
C
         IF (RE_PT.LT.1000.) THEN
            CD = 24./RE_PT*(1+0.15*RE_PT**0.687)
         ELSE
            CD = 0.44
         ENDIF
C
C---- Calculate the momentum source term and linear coefficients
C
         FACT = HALF*DENSITY_FL*AREA*CD*SLPVEL_PT
C
         SOURCE(1) = FACT*(VEL_FL(1)-VEL_PT(1))
         SOURCE(2) = FACT*(VEL_FL(2)-VEL_PT(2))
         SOURCE(3) = FACT*(VEL_FL(3)-VEL_PT(3))
C
         COEF_PT = -FACT
         COEF_FL =  FACT
C
      END
```

Outlined instructions for the Fortran file is available in Creating the Shared Libraries (p. 640).

The specification of the input and output variables occurs in CFX-Pre. The definition for the user routine is created, setting the **Option** to `Particle User Routine`, setting the calling name to `pt_momsource` and specifying the **Library Name** and **Library Path** as appropriate.

On the **Fluid Pairs** form in the **Mass and Momentum** frame, `Particle User Source` is created and chosen from the drop-down list.

The Argument Variables List contains the following variables:

- Red Sand.Mean Particle Diameter
- Red Sand.Velocity
- Red Sand.Particle Reynolds Number
- Red Sand.Particle Slip Velocity
- AirSTP.Density
- AirSTP.Velocity

The Return Variables List contains all of the available options:

- Source
- Source Coefficient with respect to Particle Variable
- Source Coefficient with respect to Fluid Variable

8.5.5. Particle Injection Regions

For details, see Particle Injection Regions (p. 393).

8.6. Particle Boundary Options and Behavior

The following topics will be discussed:

- Particle-Rough Wall Model (Virtual Wall Model) (p. 390)
- Inlet/Opening Boundaries (p. 378)
- Outlet Boundaries (p. 382)
- Wall Boundaries (p. 383)
- Symmetry Plane Boundaries (p. 391)
- Interface Boundaries (p. 391)
- Domain Interfaces (p. 392)

8.6.1. Inlet/Opening Boundaries

You can introduce particles into the domain by enabling the **Define Particle Behavior** toggle on the **Fluid Values** form and specifying their properties on an inflow or opening boundary. This is used when you have more than one inlet, but particles only enter through some of them. The particle velocity, injection position, diameter distribution, and mass flow rate need to be specified. For details, see Fluid Values for Inlets and Openings in the *CFX-Pre User's Guide*.

8.6.1.1. Mass and Momentum

The `Normal Speed`, `Cartesian Velocity Components`, and `Cylindrical Velocity Components` are described at Mass and Momentum (p. 119).

The `Zero Slip Velocity` option causes the particles to be injected at the local velocity of the continuous phase. It can be used at an inflow boundary that is specified using a mass flow or pressure option for the continuous phase.

8.6.1.2. Particle Position

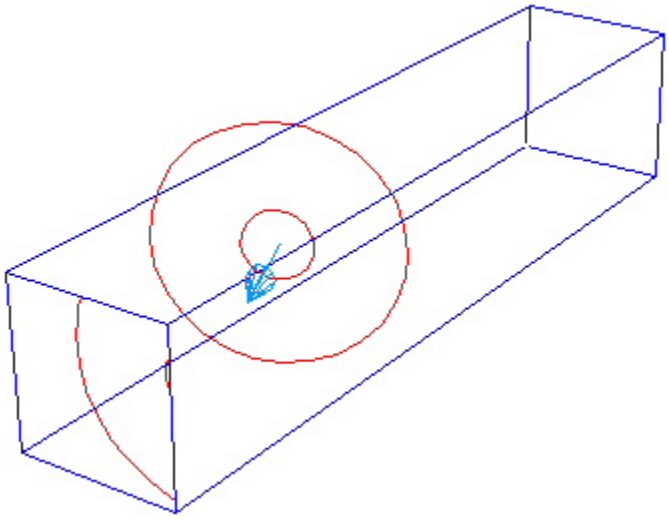
8.6.1.2.1. Uniform Injection

This option produces a random injection over the entire inlet. For details, see Number of Positions (p. 381).

8.6.1.2.2. Uniform Injection within Annulus

This option restricts the injection location to an annular region on the boundary condition. The annulus is defined by an axis, as well as a distance from the axis for both the inner and outer radii of the annulus.

Figure 8.2: Annulus projected onto a boundary region



The annulus is assumed to be infinitely long in both the positive and negative directions of its axis, which is defined by the two specified points. Particles will be injected through any portion of the boundary region intersecting the infinite annulus. If there is no overlapping region between the boundary region and the projection of the annulus onto the boundary region, then no particles will be injected.

A description of the format used to enter the First and Second Point of Axis is available in:

- Point Data Format (p. 381)
- Number of Positions (p. 381).

8.6.1.2.3. Injection With Line Weighting

The **First** and **Second Line Point for Normal Distribution** define the end points of the line. A description of the format used to enter the first and second point of axis is available in Point Data Format (p. 381). It is expected that the line lies in the plane of the injection boundary. The injection weighting is a normal distribution around the line, with the **Standard Deviation for Normal Distribution** determining the shape of the distribution. For details, see Number of Positions (p. 381).

8.6.1.2.4. Injection With Point Weighting

This option weights the injection location towards a point. It is expected that the point lies in the plane of the injection boundary. A normal distribution is used to determine the probability of injection as you move away from the point, using the **Standard Deviation for Normal Distribution**. A description of the format used to enter the **Center Point for Normal Distribution** is available in:

- Point Data Format (p. 381)
- Number of Positions (p. 381).

8.6.1.2.5. Injection With Circular Weighting

The **Center Point for Normal Distribution** and **Distance From Center Point** define the center and radius of the circle, which is expected to lie in the plane of the injection boundary. A description of the format used to enter the center point is available in Point Data Format (p. 381).

The injection weighting follows the circumference of the circle (that is, there is a lower probability of injecting particles at the center of the circle). The **Standard Deviation for Normal Distribution** determines the shape of the normal distribution as you move away from the circumference of the circle. For details, see Number of Positions (p. 381).

8.6.1.2.6. Injection With User Defined Weighting

The **Face Weighting Factor** is a dimensionless number that sets a weighting factor between 0 and 1 for each location on the injection boundary. It is expected to be a defined by a function of x, y, and/or z using either a CEL expression or a `User CEL Function`. For details, see Number of Positions (p. 381).

8.6.1.2.7. Injection at Face Centers

Particles are injected at the center of each element face. An illustration of the location of face centers is available in Discretization of the Governing Equations in the *CFX-Solver Theory Guide*.

8.6.1.2.8. Injection at IP Face Centers

Particles are injected at the integration points for each face. An illustration of the location of integration points is available in Discretization of the Governing Equations in the *CFX-Solver Theory Guide*.

8.6.1.2.9. Number of Positions

For each particle type, you must specify the number of representative particles injected at a boundary condition or particle injection region.

For steady-state simulations, the **Number of Positions** parameter sets the total number of particles to be injected. You can specify the number directly or proportional to the injected mass flow rate. In the latter case, you must specify the number of particles per unit mass flow.

For transient simulations, you must specify the number of injected particles per unit time. You can specify the number directly or proportional to the injected mass flow rate. In the latter case, you must specify the number of particles per unit time and the mass flow rate. In a transient simulation you can specify the number of particles or mass flow rate as an expression in time.

Note:

For user defined injections with the number of particles being a function of the injected mass flow rate, you must specify the mass flow rate in the CCL and not in a user routine. This applies to steady-state simulations, as well as to transient simulations.

When choosing the number of particles, you should remember that the modeled particles are a representative sample of the actual particles and the number of particles modeled will be much smaller than the real number of particles (for details, see Particle Number Rate (p. 354)). The specified **Particle Mass Flow** rate accounts for the fact that a smaller number of particles are modeled than really exist. The size and location on the boundary of the modeled particles is determined randomly (but can be weighted using a particular distribution).

The appropriate number of injected particles to be used in a simulation cannot be easily determined. This number strongly depends on several parameters such as the fluid grid size or the distribution of the particles in the domain. The area of influence of the particles in the fluid phase must be covered by enough particles per element. The number of injected particles must be varied in order to see if the number is sufficiently large, as done in grid refinement studies.

8.6.1.2.10. Point Data Format

Some of the options described above require point data to be entered. The format used for this in CFX-Pre should be a comma separated list corresponding to the x, y, and z coordinates, for example, `0,0,-0.1`, with the units selected from the drop-down list. If an expression is used, then the format will be similar to `0[m], 0[m], -0.1[m]`; that is, with a unit specified for each coordinate.

8.6.1.3. Particle Locations

By default, particles are injected randomly within the location constraints which are set under the **Particle Position** option. To enforce an equal spatial distribution, you can set the **Particle Locations** option to `Equally Spaced`.

8.6.1.4. Particle Diameter Distribution

This option applies to domains and boundaries. When specified for boundaries, the settings applied override the domain values. For details, see Particle Diameter Distribution (p. 357).

8.6.1.5. Particle Mass Flow Rate

Specify the mass flow rate that is shared among all particles. This should be the total mass flow rate of particles through this boundary.

This need not be specified for one-way coupled particles if you are only interested in the particle tracks and not in derived quantities, such as particle volume fraction and forces on walls. It is always required for fully coupled particles.

The overall mass flow rate of the particles is shared among the representative particles being tracked. By dividing by the (initial) mass of the particle, this means that each representative particle has a **Particle Number Rate**. This quantity is used internally in the code for calculating overall sources to the continuous phase, and is also used in postprocessing for calculating mass flows of particles through boundaries, forces on walls, and so on.

Note:

When expert parameter `pt force export massflow` is set to `'t'`, particle mass flows are always written to the results files. Otherwise they will only be written if they are used in CEL expressions or monitors. For details, see Particle Tracking Parameters (p. 622).

8.6.1.6. Heat Transfer

The static temperature is required for particle phases when a heat transfer model has been selected for the particles.

8.6.1.7. Component Details

This section of the form becomes available when a variable composition mixture has been used to define the particles. Mass fractions are required for each of the species in the particle phase, and must sum to unity on all boundaries.

8.6.1.8. Particle Actions at Inlets and Openings

When existing particles in the domain reach an inlet or an opening, they escape from the domain.

8.6.2. Outlet Boundaries

8.6.2.1. Particle Actions at Outlets

When existing particles in the domain reach an outlet, they escape from the domain.

8.6.3. Wall Boundaries

8.6.3.1. Wall Interaction

The wall interaction option is available in CFX-Pre for simulations involving Lagrangian particles or when a particle user routine has been created. The wall-interaction options available in Ansys CFX are outlined below:

- Standard Particle-Wall Interaction (p. 383)
- Wall Film Modeling (p. 384)
- User Defined Particle-Wall Interaction (p. 390)

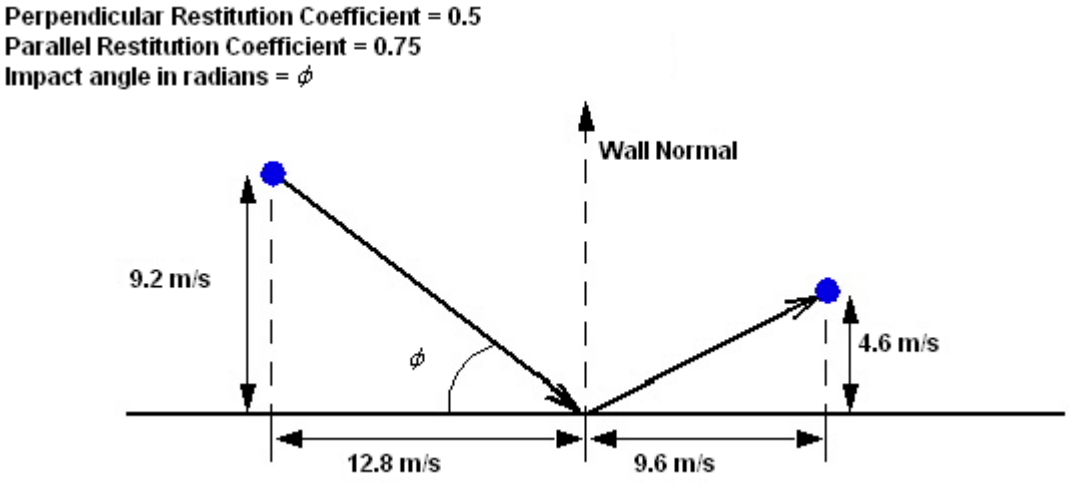
8.6.3.1.1. Standard Particle-Wall Interaction

This is the default particle-wall interaction model in Ansys CFX, typically controlled by setting the wall interaction mode to `Equation Dependent` in CFX-Pre. With this model, the droplet is reflected off a wall and the momentum change across the collision is described using the perpendicular and parallel coefficients of restitution. For details, see Fluid Values for Walls in the *CFX-Pre User's Guide*.

8.6.3.1.1.1. Restitution Coefficients for Particles

The parallel and perpendicular restitution coefficients describe the action of particles when they hit a wall. Enter a numerical quantity or CEL based time-dependent expression to specify the value of restitution coefficients. Coefficient values of 1 described an elastic collision, while values less than 1 describe an inelastic collision. Figure 8.3: Particle Track Behavior at a Wall Boundary (p. 383) shows an example of a particle velocity before and after impacting a wall when the perpendicular restitution coefficient is 0.5 and the parallel restitution coefficient is 0.75.

Figure 8.3: Particle Track Behavior at a Wall Boundary



The parallel coefficient will almost always be 1. The perpendicular coefficient will depend on the particle material. Particles that bounce off walls will have a perpendicular coefficient close

to 1, while particles that stick to walls (for example, water droplets) will have a perpendicular coefficient of 0.

If you want to terminate tracking of particles when they hit a wall boundary, then you can set both coefficients to zero. Although it should be noted that when a particle hits a wall and the perpendicular coefficient of restitution is set to zero then the particle comes to a stop regardless of what is set for the parallel coefficient of restitution

Important:

In case of a moving wall, the coefficients need to consider the velocity of the wall. This means the relevant coordinate system must always be the wall and the relative particle velocity (with respect to the wall) has to be taken into account if either of the coefficients are smaller than 1.

Note:

Setting the perpendicular coefficient of restitution to a small positive number is *not* a valid way to simulate particle sliding along a wall. Currently there is no method for accurately simulating this type of particle sliding.

The change in momentum of particles due to the action at a wall results in a force on the wall.

8.6.3.1.2. Wall Film Modeling

The following models available in Ansys CFX enable the simulation of wall film formation by particles being deposited at a wall, as well as the resulting interaction of particles with the film covered wall.

- Elsaesser particle-wall interaction model
- Stick-to-wall model
- User Defined Wall Film Modeling

For details, see Particle-Wall Interaction in the *CFX-Solver Theory Guide*.

8.6.3.1.2.1. User Defined Wall Film Modeling

Particle-wall interactions involve complex physics, especially in the presence of a wall film. The available models may not suit certain cases. Therefore a User Fortran interface has been made available to enable you to use your own models in order to define the creation of a wall film. The following example illustrates how this can be done.

This is an example of a basic routine that can be used to calculate the particle-wall interaction, enabling the generation of a particle wall film. The model uses a "child generation" model (i.e. a droplet can split into several "child" droplets). In this example, a particle impacting the wall may either be fully deposited at the wall or it will break up ("splash") into two parts ("children"): one being reflected and the other one being deposited at the wall.

In this example, the splashing criteria is solely based on the impact details of the particle, such as the impact angle and momentum.

The input arguments for the routine are:

- Wall impact angle
- Particle velocity components
- Particle diameter
- Particle number rate
- Fluid density at particle position

The return variables of the user routines are:

- "Breakup indicator"
- Particle diameter after wall interaction
- Particle number rate after wall interaction
- Normal coefficient of restitution of particle
- Parallel coefficient of restitution of particle
- Particle mode

The User Defined Wall Film Interaction model supports the following combinations of return variables:

- `Child Droplet Mode`

  This is a required return variable. It currently holds two values:

  – `__regular_particle__` (integer value of `1`)

  – `__wall_particle__` (integer value of `20`)

- `Child Droplet Breakup Indicator`

  This is a required return variable that is used by the particle tracker to determine the number of child droplets created. It can be set to a real array with a value of either `1` or `0`. If set to `0`, then any child data returned to the tracker is ignored.

- `Child Droplet Diameter`

  This is a real array that holds the child droplet diameter values.

Or

- `Child Droplet Diameter Ratio`

  This is a real array that holds the child droplet diameters' ratios (`diameter_child` divided by `diameter_parent`).

- `Child Droplet Diameter Ratio`

This is a real array that holds the child droplet number rates.

- `Child Droplet Velocity`

  This is a real array that holds the values of the child droplet velocity components.

Or

- `Child Droplet Perpendicular Coefficient of Restitution`

  Real array that holds the child droplet normal coefficient of restitution (used to determine the velocity components of child droplets).

- `Child Droplet Parallel Coefficient of Restitution`

  This is a real array that holds child droplet parallel coefficients of restitution (which are used to determine the velocity components of child droplets)

Note that a maximum of 4 child droplets may be generated with the "child generation" model.

You can copy the routine `pt_wall_splash.F` from the `examples\UserFortran` directory, which is under the installation directory.

The Fortran code for this example is:

```
#include "cfx5ext.h"
dllexport(pt_wall_splash)
      SUBROUTINE PT_WALL_SPLASH (NLOC,NRET,NARG,RET,ARG,CRESLT,
     &                           CZ,DZ,IZ,LZ,RZ)
CC
CD User routine: template for particle user routine
CC
CC --------------------
CC        Input
CC --------------------
CC
CC  NLOC   - number of entities
CC  NRET   - length of return stack
CC  NARG   - length of argument stack
CC  ARG    - argument values
CC
CC --------------------
CC      Modified
CC --------------------
CC
CC --------------------
CC        Output
CC --------------------
CC
CC  RET    - return values
CC
CC --------------------
CC       Details
CC --------------------
CC
CC=====================================================================
C
C ------------------------------
C     Preprocessor includes
C ------------------------------
C
#include "cfd_sysdep.h"
#include "cfd_constants.h"
C
```

```
C ------------------------------
C        Argument list
C ------------------------------
C
      INTEGER NARG, NRET, NLOC
C
      CHARACTER*(4) CRESLT
C
      REAL ARG(NLOC,NARG), RET(NLOC,NRET)
C
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C=======================================================================
C
C -------------------------
C    Executable Statements
C -------------------------
C
C=======================================================================
C
C     Return variables:
C     -----------------
C
C     Child droplet breakup indicator                      : RET(1,1)
C     Child droplet diameter                               : RET(1,5)
C     Child droplet number rate                            : RET(1,9)
C     Child droplet normal coef of restitution             : RET(1,13)
C     Child droplet parallel coef of restitution           : RET(1,17)
C     Child droplet mode                                   : RET(1,21)
C
C     Argument variables
C     ------------------
C
C     Particle impact angle                               : ARG(1,1)
C     Particle velocity                                   : ARG(1,2)
C     Particle diameter                                   : ARG(1,5)
C     Particle number rate                                : ARG(1,6)
C     Fluid density                                       : ARG(1,7)
C
C     We know that NLOC is 1 for the particle user source routines!!!!
C=======================================================================
C
C-----------------------------------------------------------------------
C     Calculate the return variables
C-----------------------------------------------------------------------
C
      CALL SPLASH (RET(1,1),RET(1,5),RET(1,9),RET(1,13),RET(1,17),
     &             RET(1,21),ARG(1,1),ARG(1,2),ARG(1,5),ARG(1,6),
     &             ARG(1,7))
C
      END
C
      SUBROUTINE SPLASH (BRKUP_IND,CHILD_DIAM,CHILD_NRATE,
     &                   CHILD_COEF_N,CHILD_COEF_P,CHILD_MODE,
     &                   ANGLE,VEL_PT,DIAM_PT,NRATE_PT,DENS_FL)
C
C -------------------------
C    Preprocessor includes
C -------------------------
C
#include "cfd_sysdep.h"
#include "cfd_constants.h"
C
C---- Do not change these defines, they have to be consistent with what
C     the solver uses
C
#define   __regular_particle__ 1
```

```
#define      __wall_particle__ 20
#define     __pt_uf_get_data__ 1
#define    __pt_uf_save_data__ 2
#define        __pt_uf_iseed__ 1
#define        __pt_uf_pmode__ 2
C
C ------------------------------
C        Argument list
C ------------------------------
C
      REAL   FACT, ANGLE, VEL_PT(3), DIAM_PT, NRATE_PT,
     &       BRKUP_IND(3), CHILD_DIAM(3), CHILD_NRATE(3),
     &       CHILD_COEF_N(3), CHILD_COEF_P(3), CHILD_MODE(3), DENS_FL
C
C ------------------------------
C        Local variables
C ------------------------------
C
      REAL    SIGMA, VISC_PT, R, SPEED_PT, ANGLE_DEG, ALFA, BETA,
     &        GAMMA, VEL_NORM, OH, RE, K, M1M0
      INTEGER ICHILD, ISEED, ACTION
C
C ------------------------------
C    Executable statements
C ------------------------------
C
C=======================================================================
C     Prologue
C=======================================================================
C
C---- Initialization
C
      ICHILD = 0
C
C---- Get particle seed info (needed for random number generator)
C
      ACTION = __pt_uf_save_data__
      CALL USER_PARTICLE_INFO(ACTION,__pt_uf_iseed__,ISEED)
C
C---- Some material properties (water)
C     --> Surface tension coefficient
C     --> Particle viscosity
C
      SIGMA   = 0.0201
      VISC_PT = 0.001
C
C---- Random number (required for deviation angle)
C
      CALL GET_RANDOM(R,1,ISEED)
C
C=======================================================================
C     Computation Section
C=======================================================================
C
C---- Particle velocity magnitude and impact angle
C
      SPEED_PT  = SQRT(VEL_PT(1)**2 + VEL_PT(2)**2 + VEL_PT(3)**2)
      ANGLE_DEG = ANGLE*180./PI
C
C---- Ohnesorge and Reynolds number
C     --> Re, based on wall normal impact velocity!
C
      VEL_NORM = SIN(ANGLE)*SPEED_PT
      OH = VISC_PT/SQRT(DENS_FL*DIAM_PT*SIGMA)
      RE = DENS_FL*DIAM_PT*VEL_NORM/VISC_PT
C
C-----------------------------------------------------------------------
C     Precompute reflection angle
C-----------------------------------------------------------------------
C
C---- Impact angle, ALFA, average reflection angle, BETA. Add random
```

```
C     deviation angle, GAMMA, if ALFA < 15 deg.
C
      ALFA  = 90.   - ANGLE_DEG
      BETA  = 61.88 + 0.326*ALFA
      GAMMA = 17.6  - 0.18*ALFA
C
      IF (ALFA .LT. 15.) BETA = BETA + (R-0.5)*GAMMA
C
C---- Compute K-value
C     --> Clip K to avoid overflow during m1/m0 calculation
C         (221^9.2133 = 3.9773E+21)
C
      K = MIN(OH*RE**1.25,221.)
C
C-----------------------------------------------------------------------
C     K <= 57 -> Particle deposited at wall
C-----------------------------------------------------------------------
C
      IF (K .LE. 57) THEN
C
        ICHILD = ICHILD + 1
        BRKUP_IND(ICHILD)    = 1.
        CHILD_DIAM(ICHILD)   = DIAM_PT
        CHILD_NRATE(ICHILD)  = NRATE_PT
        CHILD_COEF_N(ICHILD) = 0.0
        CHILD_COEF_P(ICHILD) = 0.0
        CHILD_MODE(ICHILD)   = __wall_particle__
C
C-----------------------------------------------------------------------
C     ... else partially reflected
C-----------------------------------------------------------------------
C
      ELSE
C
C---- Mass fraction across reflection
C
        M1M0 = 3.9869E-21*K**9.2133
C
C---- Reflected portion of particle
C
        ICHILD = ICHILD + 1
        BRKUP_IND(ICHILD)    = 1.
        CHILD_DIAM(ICHILD)   = DIAM_PT*M1M0
        CHILD_NRATE(ICHILD)  = NRATE_PT
        CHILD_COEF_N(ICHILD) = COS(BETA*PI/180.)*SPEED_PT
        CHILD_COEF_P(ICHILD) = SIN(BETA*PI/180.)*SPEED_PT
        CHILD_MODE(ICHILD)   = __regular_particle__
C
C---- Deposited portion of particle
C
        ICHILD = ICHILD + 1
        BRKUP_IND(ICHILD)    = 1.
        CHILD_DIAM(ICHILD)   = DIAM_PT*(1. - M1M0)
        CHILD_NRATE(ICHILD)  = NRATE_PT
        CHILD_COEF_N(ICHILD) = 0.0
        CHILD_COEF_P(ICHILD) = 0.0
        CHILD_MODE(ICHILD)   = __wall_particle__
C
      ENDIF
C
C=======================================================================
C     Epilogue
C=======================================================================
C
C---- Update ISEED in particle database with value computed in
C     this routine
C
      ACTION = __pt_uf_get_data__
      CALL USER_PARTICLE_INFO(ACTION,__pt_uf_iseed__,ISEED)
C
      END
```

8.6.3.1.3. User Defined Particle-Wall Interaction

The user-defined particle-wall interaction model can be used to model such quantities as restitution coefficient, mass flow absorption, erosion and particle breakup using a Fortran routine. The Fortran routine should first be written, and a shared library created. For details, see Creating the Shared Libraries (p. 640). After this step, you should create a User Routine of type `Particle User Routine`. For details, see Particle User Routines in the *CFX-Pre User's Guide*.

Input arguments to the subroutine can be any valid variables (selected from the Arguments drop-down list in CFX-Pre).

The returned quantities must be made up of one or more of the following:

- Perpendicular / Parallel Coefficient of Restitution
- Mass Flow Absorption Coefficient
- Particle Erosion (the returned erosive wear is assumed to be non-dimensional, e.g. grams of eroded material per gram of colliding particles)
- Particle Breakup Factor (the number rate of a particle is increased by the breakup factor, and the diameter is reduced by the cube root of the breakup factor)

In the `<install_dir>\examples\UserFortran` directory, you can find a wall-interaction particle routine, `pt_breakup_wall.F`, and corresponding CCL template, `pt_breakup_wall.ccl`.

8.6.3.2. Erosion Model

The erosion model can be set on a per-boundary or per-domain basis. When selected for the domain, the domain settings will apply for all boundaries that do not explicitly have erosion model settings applied to them. To override the domain setting, turn-on the **Erosion Model**. Additional information on erosion is available. For details, see:

- Erosion (p. 362)
- Basic Erosion Models in the *CFX-Solver Theory Guide*.

8.6.3.3. Particle-Rough Wall Model (Virtual Wall Model)

The rough wall model can be set on a per-boundary or per-domain basis. When selected for the domain, the domain settings will apply for all boundaries that do not explicitly have rough wall model settings applied to them. To override the domain setting, turn on the Rough Wall model on the corresponding patches. Additional information on rough wall treatment is available in Particle-Rough Wall Model (Virtual Wall Model) (p. 364) and The Sommerfeld-Frank Rough Wall Model (Particle Rough Wall Model) in the *CFX-Solver Theory Guide*.

8.6.3.4. Particle Breakup

To simulate the breakup of particles due to aerodynamic forces, five different interior breakup models are available: Reitz & Diwakar, TAB, ETAB, CAB and Schmehl model. See Spray Breakup Models in the *CFX-Solver Theory Guide* for information on these models.

8.6.3.5. Mass Flow Absorption

The mass flow absorption is available to enable a fraction of particles to leave the domain when they collide with a wall. For each particle tracked, the mass flow (number) rate is reduced whenever it strikes a wall.

8.6.3.6. Mass and Momentum

Particles can be injected through walls in the same way as they are injected through inlets. For details, see Mass and Momentum (p. 379).

8.6.3.7. Particle Impact Angle

The impact angle φ is the angle between the approaching particle track and the wall. It is measured in radians. See Figure 8.3: Particle Track Behavior at a Wall Boundary (p. 383).

8.6.3.8. Particle Position

The particle injection positions on a wall are defined in the same way as at inlets. For details, see Particle Position (p. 379).

8.6.3.9. Particle Diameter Distribution

The diameter distribution on a wall is described in the same way as for a domain. For details, see Particle Diameter Distribution (p. 357).

8.6.3.10. Particle Mass Flow Rate

The particle mass flow rate through a wall is defined in the same way as through inlets. For details, see Particle Mass Flow Rate (p. 382).

8.6.4. Symmetry Plane Boundaries

When a particle reaches a symmetry plane boundary condition, it is reflected.

8.6.5. Interface Boundaries

If your simulation includes any solid domains, then you may want to set particle options at the fluid-solid interface. To do this, you should first create the required fluid-solid domain interfaces, and then edit the automatically created interface boundary condition for the fluid side of the interface. The options on the fluid side of fluid-solid interfaces are specified in the same way as for wall boundaries. For details, see:

- Wall Boundaries (p. 383)
- Interface Boundary Conditions in the *CFX-Pre User's Guide*.

8.6.6. Domain Interfaces

8.6.6.1. Fluid-Fluid

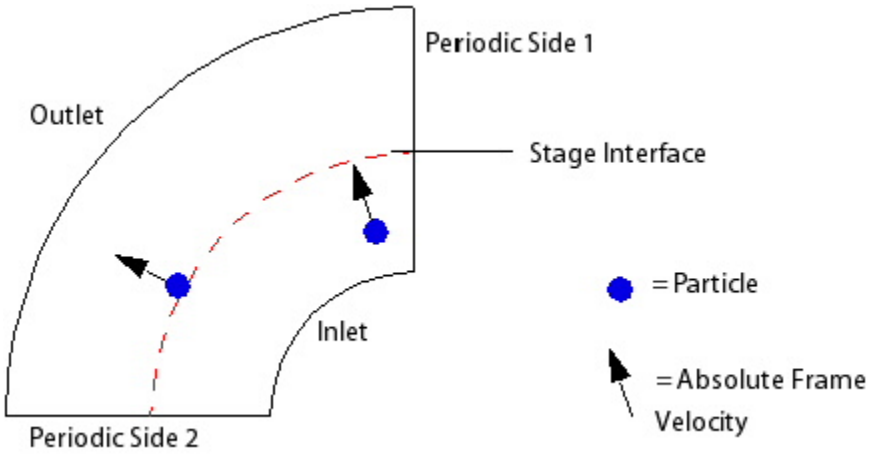
8.6.6.1.1. Frame Change Option = None

When there is no frame change across a fluid-fluid domain interface, then particles simply cross the interface and continue in the next domain.

8.6.6.1.2. Frame Change Option = Frozen Rotor

At a frozen rotor domain interface, particles simply cross the interface and continue in the next domain. You should not use a frozen rotor interface unless particles cross the interface approximately uniformly. For example, if most of the particles cross the interface at one location, then the transient behavior due to the relative motion of the two components will be completely lost. This is the same limitation that applies to local fluid features at frozen rotor domain interfaces. For details, see Frozen Rotor (p. 237).

8.6.6.1.3. Frame Change Option = Stage (Mixing-Plane)



When a particle crosses a stage interface, it enters the new domain at a random location and its velocity is rotated accordingly. This is in addition to change in the relative velocity that occurs due to the particle changing to a new frame of reference. This can be thought of as equivalent to the circumferential "averaging" that is performed for the fluid flow variables at stage interfaces. For details, see Stage (Mixing Plane) (p. 238).

If a pitch change occurs across the interface, the mass flow rate is updated appropriately. The balance sections of the CFX-Solver Output file that is written by the CFX-Solver as the solution proceeds reflect these changes.

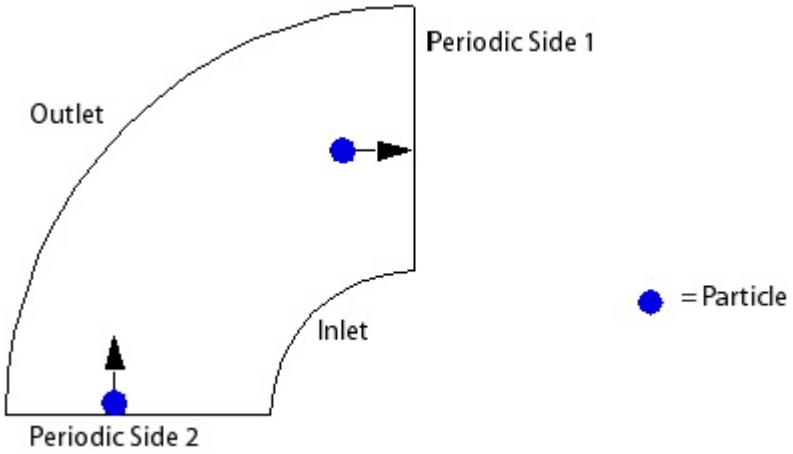
8.6.6.1.4. Frame Change Option = Transient Rotor Stator

In a transient simulation, interfaces between rotating and non-rotating domains are modeled using the Transient Rotor Stator frame change option.

When particles cross such an interface, they continue at the corresponding position across the interface. This is similar to the approach used for the Frozen Rotor interface option but the correct,

time dependent relative position of the two domains is taken into account. Due to the fact that rotating domains are rotated in CFD-Post, particle tracks will be contiguous across transient rotor interfaces.

8.6.6.2. Periodic Connections



When a particle reaches a periodic domain interface, it emerges at the new periodic location and the particle's velocity is rotated, as expected.

8.7. Subdomains

For simulations with particle tracking, a subdomain has a **Fluids** tab that enables the subdomain to act as a porous medium that can absorb particles. The **Absorption Diameter** setting specifies the minimum size of particles that are absorbed (removed from the calculation) upon entering the porous region (that is, the subdomain).

Wherever multiple subdomains occupy the same volume, the smallest specified **Absorption Diameter** value is used.

8.8. Particle Injection Regions

Injection regions can optionally be created to inject particles on sphere or cone locators, or on a custom locator defined by a User Fortran subroutine. The user subroutine should be of type `Particle User Routine` to make it available for selection in the drop-down list.

For each method of injection, apart from the location specification itself, other quantities required are the same as those that would be required at a boundary (for example, temperature would be required if heat transfer was being modeled). For details, see Particle Boundary Options and Behavior (p. 378). If a User Fortran routine is being used, all physical quantities (apart from mass fractions) can be set either in the Fortran or from the user interface.

The center and the axis (if applicable) of the injector, as well as the injection velocity (for Cartesian or cylindrical velocity components), are interpreted in the local coordinate system. The exception is for user defined injections where the particle position and the injection velocity – if returned from the user

routine – refer to the global coordinate frame `Coord 0` rather than to the one specified under **Coordinate Frame**.

The local coordinate system cannot be specified with a rotating frame motion. When a domain with a rotating frame is included in the injected region, the position from which particles are injected is assumed to rotate along with the domain. For additional information, see Rotating Frames of Reference (RFR) (p. 77) and GGI and MFR Theory in the *CFX-Solver Theory Guide*.

The following sections outline how to set up the locations using the following options:

8.8.1. Sphere

8.8.2. Cone

8.8.3. Cone with Primary Breakup

8.8.4. User Defined Injection Regions

8.8.1. Sphere

When creating a sphere, the injection center, injection velocity magnitude, and number of positions are required. If the radius of the injection sphere is not given, the radius defaults to zero (that is, all particles will be injected from a point).

8.8.2. Cone

When creating a cone, a cone may describe the shape of the particle tracks. The particles actually enter from a 2D region, which may be a point, a circle, or an annular region according to the **Cone Definition** parameters. The orientation of this plane is determined by the injection direction. Note that the direction in which the particles enter may or may not be independent of the orientation of the 2D region.

The following cone injection types are supported in CFX-Pre:

- **Point Cone**
- **Hollow Cone**
- **Ring Cone**
- **Full Cone**

The injection center, injection velocity, injection direction, number of positions, and cone definition are required for setting up various cone injection types. An optional **Dispersion Angle** of the spray can be specified for point and hollow cone types by setting one of the following parameters:

- `Dispersion Angle`: The specified value is considered to be a 'half angle' similar to the spray cone angle definition.
- `Standard Deviation of Dispersion Angle`: The specified value is considered as a normal distribution of the spray within the dispersion angle.

8.8.2.1. Injection Velocity

The injection velocity can be set via the specification of the *Velocity Magnitude*, *Cartesian Velocity Components*, *Cylindrical Velocity Components*, or *Zero Slip Velocity*. The *Zero Slip Velocity* option causes the particles to be injected at the local fluid velocity of the coupled continuous phase.

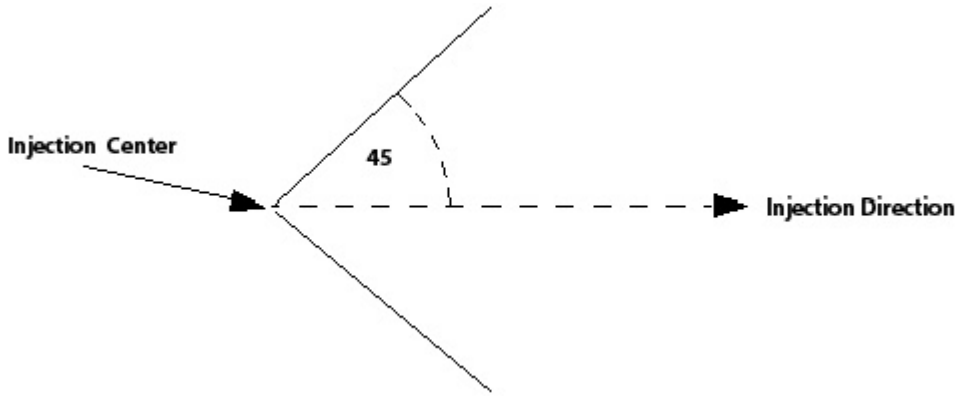
Note that the specification of *Cylindrical Velocity Components* is not supported for *Point Cones*.

To simplify the specification of swirl for the injection velocity a new particle variable <*Particle Type*>.*Particle Position* was introduced. The components of this variable hold the x, y, and z distance from the local injector center and can be used in CEL to define swirl components for the particle injection velocity vector.

Velocities for particles are always assumed to be specified in the absolute frame.

8.8.2.2. An Example of a Point Cone

The following illustration shows a point cone with cone angle of 45 degrees. The injection direction is specified normal to the injection boundary. As the **Cone Definition** parameter for point cone does not require the specification of inner and outer radii, the particles are injected from a point at the injection center.



8.8.2.3. An Example of a Point Cone Using the Dispersion Angle

This following illustration shows how the optionally specified dispersion angle, γ, changes the injection type from point cone (Figure 8.4: Point cone with specified cone angle (p. 396)) to point hollow cone (Figure 8.5: Point cone with specified cone angle and dispersion angle (p. 396)). The cone angle is shown as Θ.

Figure 8.4: Point cone with specified cone angle

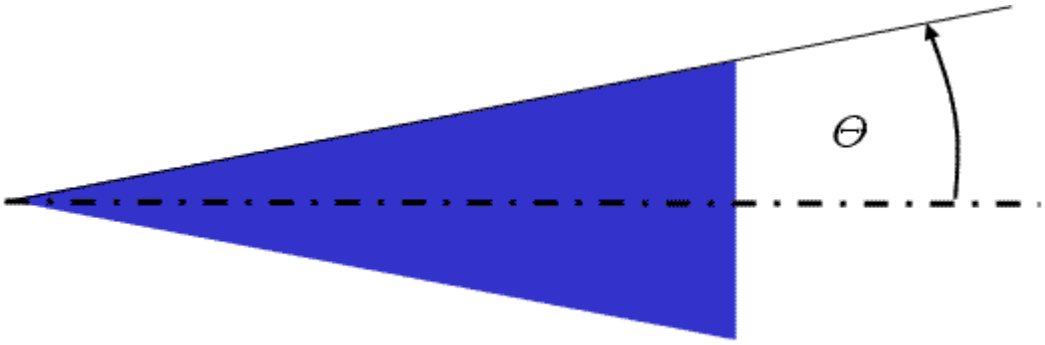
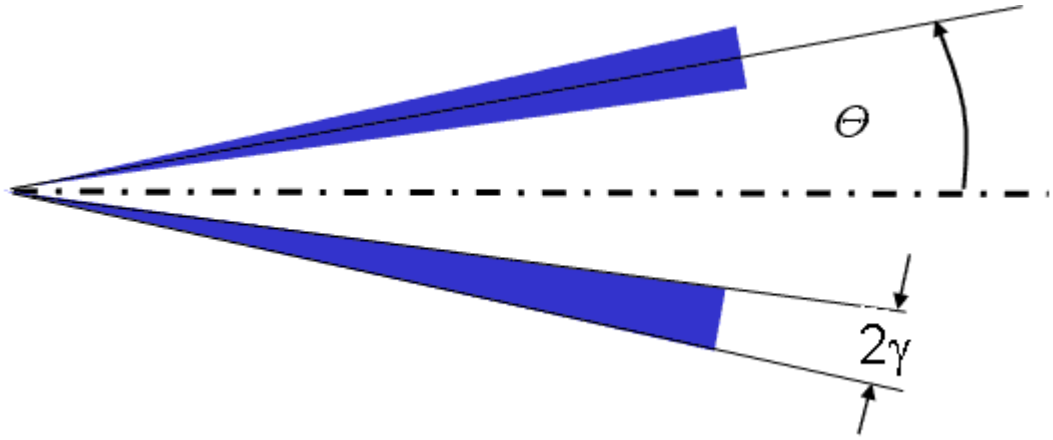


Figure 8.5: Point cone with specified cone angle and dispersion angle



8.8.2.4. An Example of a Hollow Cone using the Dispersion Angle

This following illustration shows how the optionally specified dispersion angle, γ, changes the spray pattern in the case of a hollow cone injection. The cone angle is shown as Θ.

Figure 8.6: Hollow cone with specified cone angle

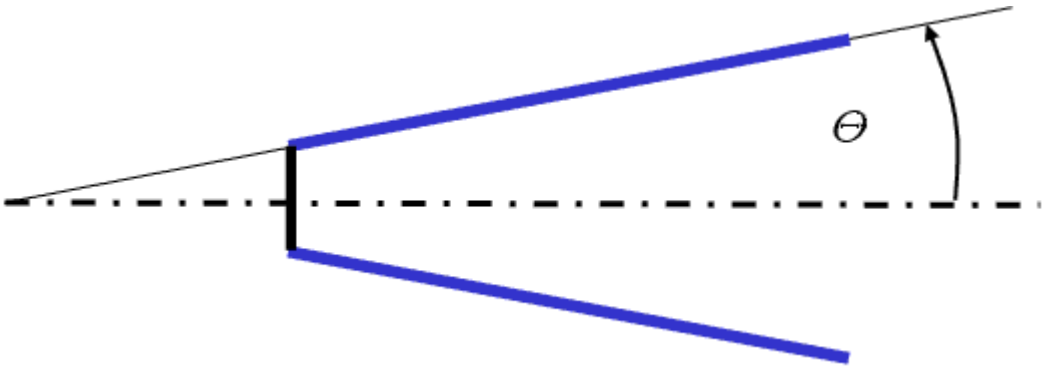
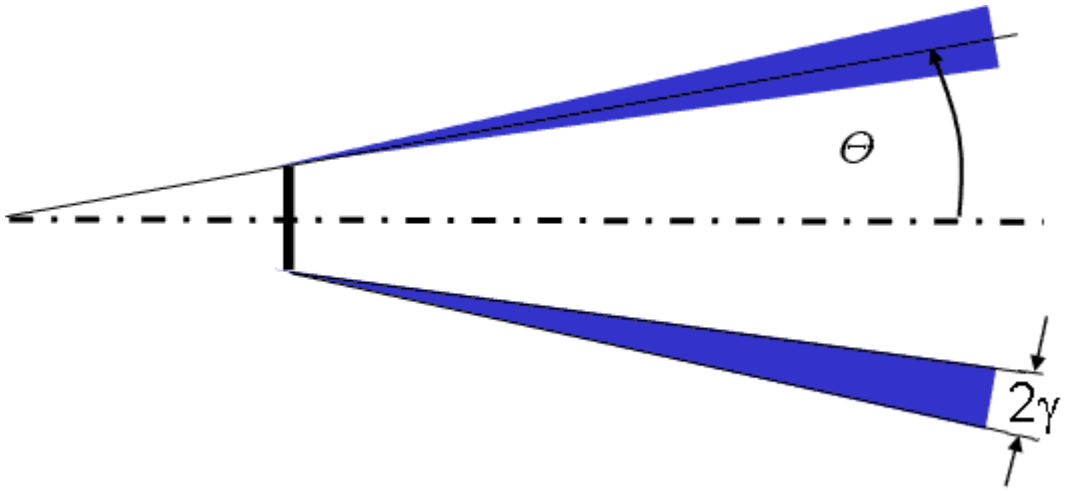


Figure 8.7: Hollow cone with specified cone angle and dispersion angle



8.8.3. Cone with Primary Breakup

This option enables the use of primary breakup models to determine starting conditions for the droplets that leave the injection nozzle. In Ansys CFX 2021 R2, various primary breakup models are available. For details about these models and their required input data, refer to the following sections in CFX-Solver Theory Guide:

- Blob Method
- Enhanced Blob Method
- LISA Model
- Turbulence Induced Atomization

8.8.4. User Defined Injection Regions

User defined injection regions enable you to use Fortran to specify a custom injection region. The routine may optionally return a range of values (such as temperature, particle mass flow rate, and so on) or they can be specified by enabling the appropriate toggles in CFX-Pre. Information on setting up a particle routine is available in Particle User Routines in the *CFX-Pre User's Guide*.

Note:

In the case of user defined injection, the particle position and the injection velocity – if returned from the user routine – refer to the global coordinate frame `Coord 0` rather than to the one specified under **Coordinate Frame**.

An example of a Fortran routine is given below. This routine returns the following variables:

- Particle Position (X, Y, Z)
- Particle Mass Flow Rate

- Mean Particle Diameter
- Velocity (u, v, w)
- Temperature

As a result, the respective options under **Define Particle Behavior** are not entered. Similar to other routines, after the Fortran routine is written, a shared library should be created. For details, see Creating the Shared Libraries (p. 640). The routine should then be defined as a User Particle Routine in CFX-Pre. For details, see Particle User Routines in the *CFX-Pre User's Guide*.

In the `<install_dir>\examples\UserFortran` directory, you can find an injection routine, `pt_injection.F`, and corresponding CCL template, `pt_injection.ccl`.

8.9. Particle Output Control

The following topics are discussed in this section:

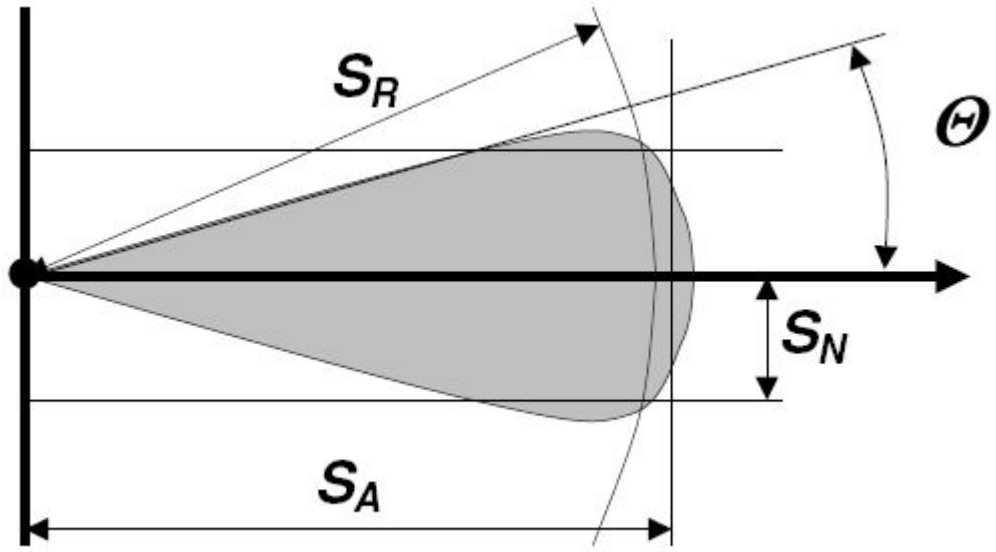
- Transient Particle Diagnostics (p. 398)
- List of Particle Variables (p. 405)

8.9.1. Transient Particle Diagnostics

For transient simulations with Lagrangian particles, it is very helpful to define integrated quantities:

- The axial penetration of a spray measured along the user-defined spray axis (S_A in Figure 8.8: Spray Penetration (p. 399))
- The radial penetration of a spray measured normal to the user-defined spray axis (S_R in Figure 8.8: Spray Penetration (p. 399))
- The penetration of a spray normal to the spray axis (S_N in Figure 8.8: Spray Penetration (p. 399))
- The penetration angle of a spray (Θ in Figure 8.8: Spray Penetration (p. 399))
- The total mass of particles

Figure 8.8: Spray Penetration

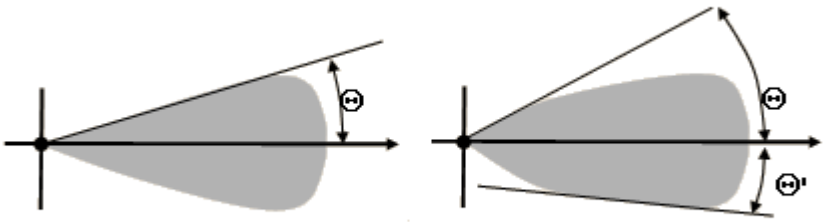


The penetration is normally evaluated by looking at a specified fraction of the spray, that is, the axial penetration of a spray is given by a distance from the spray nozzle that contains 99 percent of the particle mass. The calculation of the axial, radial, and normal penetration depth is straightforward.

The spray angle as reported by the transient particle diagnostics is calculated as follows: An imaginary cone is created with its tip at the point of injection and the cone axis parallel to the specified injection direction. The cone angle is then gradually increased up to the point where the imaginary cone contains a certain percentage of the total spray mass (by default: 99%, this can be changed via the `Contained Spray Mass Fraction` CCL parameter). The cone half angle is then reported as the spray angle, Θ.

Figure 8.9: Spray Angle Calculation for Different Spray Shapes (p. 399) shows two typical spray shapes and the reported spray angle, Θ.

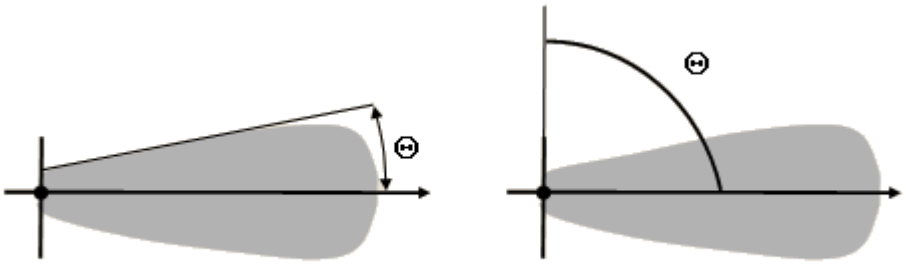
Figure 8.9: Spray Angle Calculation for Different Spray Shapes



When the spray is injected with a finite injection radius as shown in Figure 8.10: Spray Angle Calculation with Finite Injection Radius with and without Spray Radius Specified (p. 400), you will need to set the

`Spray Radius at Penetration Origin` parameter to the size of the injection radius, otherwise the solver will return a spray angle of 90°.

Figure 8.10: Spray Angle Calculation with Finite Injection Radius with and without Spray Radius Specified



There are various different ways to calculate transient particle diagnostics. A flexible User Fortran interface is available that enables you to calculate any information from all particles (given by their position, velocity, …) at any time step during a transient particle run.

8.9.1.1. User Diagnostics Routine

Besides the hard coded penetration depth and angle it is necessary to provide a method that enables you to evaluate any transient particle diagnostics. You cannot use CEL expressions for particles. As an alternative, it is possible to call a user routine that evaluates the required diagnostics information based on a specified list of particle variables. The following particle variables can be chosen:

- `Mean Particle Number`
- `Particle Number Rate`
- `Particle Position`
- `Particle Time`
- `Particle Traveling Distance`
- `Temperature`
- `Total Particle Mass`
- `Velocity`

Following this approach, the solver provides the values of all particle variables specified for user-defined diagnostics in a local working directory, together with global information about the total number of particles (NPART) at the current time step, as well as the particle type (CPT) and particle type alias name (ALIAS). This is necessary because user routines do not support subroutine arguments besides the five Fortran stacks. The user routine can now pick up the required information from the MMS by converting the variable names specified to internal solver names for which standard LOCDAT calls can be used. The obtained pointers can be passed down to another subroutine layer, which does the final calculation.

In order to monitor values calculated within the user routine, you can store a defined list of REAL variables back to the MMS (predefined place is TPD_VALUE in the local directory). Those variables are picked up by the solver and are written to the CFX-Solver Output file similar to the pre-defined transient diagnostics values. The values are also added to the list of monitored values for the Ansys CFX-Solver Manager. You must specify a list of strings (CCL-parameter: Monitored Values List) which contains names for the values to be monitored. The number of monitored values is determined from this list and the names are used in the CFX-Solver Output file and in the CFX-Solver Manager.

As an example for this strategy, a user routine has been created that simply calculates the particle mass within three spheres with different user-specified radii. The next subsections show the CCL and the listings of the required user routine, as well as the diagnostics output.

8.9.1.1.1. Example User Routine: CCL

Sphere radius and center is specified in the USER section of the CCL:

```
USER:
   CENTRE = 0.5, 0.5, 0.5
   RADIUS1 = 0.25
   RADIUS2 = 0.35
   RADIUS3 = 0.45
END
```

The CCL definition for the corresponding junction box is as follows:

```
TRANSIENT PARTICLE DIAGNOSTICS: User Routine Sphere
   Option = User Defined
   Transient Particle Diagnostics Routine = Sphere
   Particle Variables List = \
      Particle Position, \
      Total Particle Mass, \
      Particle Number Rate
   Monitored Values List = \
      First sphere, \
      Second sphere, \
      Third sphere
   Particles List = Water
END
```

As can be seen, the user routine depends on the position, mass, as well as the number rate of the particles and provides three values that are monitored during the simulation. The new diagnostics section for this CCL in the CFX-Solver Output file looks like:

```
+-----------------------------------------------------------------------+
|               Transient Particle Diagnostics                          |
+-----------------------------------------------------------------------+
Water
   User Routine Sphere
      First sphere                          3.8847E-02
      Second sphere                         5.9132E-02
      Third sphere                          7.2068E-02
```

8.9.1.1.2. Example User Routine: Mainline Routine

In the `<install_dir>\examples\UserFortran` directory, you can find an example mainline routine, `pt_tpd1.F`, and corresponding CCL template, `pt_tpd1.ccl`.

8.9.1.1.3. Example User Routine: Subroutine

```
       SUBROUTINE CALC_TPD1(TOTAL_MASS,SPHERE_CENTRE,
      &                     SPHERE_RADIUS,NPART,CRD,MASST,RATE)
C
C=======================================================================
C      Subroutine for CALC_TPD1 which does the real calculation
C=======================================================================
C
       INTEGER   NPART
       REAL      TOTAL_MASS(3), SPHERE_CENTRE(3), SPHERE_RADIUS(3),
      &          CRD(3,NPART), MASST(NPART), RATE(NPART)
C
       INTEGER IPART, I
       REAL    RADIUS
C
       DO I=1,3
         TOTAL_MASS(I) = 0.0
       ENDDO
C
       DO IPART=1,NPART
         RADIUS = SQRT( (CRD(1,IPART)-SPHERE_CENTRE(1))**2
      &               + (CRD(2,IPART)-SPHERE_CENTRE(2))**2
      &               + (CRD(3,IPART)-SPHERE_CENTRE(3))**2 )
         DO I=1,3
           IF (RADIUS.LE.SPHERE_RADIUS(I)) THEN
             TOTAL_MASS(I) = TOTAL_MASS(I) + MASST(IPART)*RATE(IPART)
           ENDIF
         ENDDO
       ENDDO
C
       END
```

8.9.1.1.4. Example: Complete CCL

The following CCL for

- Two penetration objects
  - Specified location
  - Specified particle injection region (PIR)
- One total particle mass object
- Two user defined diagnostics objects
  - User routine with three monitored values
  - User routine without any monitored values

looks as follows:

```
#=======================================================================
#    TRANSIENT PARTICLE DIAGNOSTICS
#=======================================================================
#
#-----------------------------------------------------------------------
#    Penetration
#-----------------------------------------------------------------------
#
   TRANSIENT PARTICLE DIAGNOSTICS: Penetration from Location
      Option = Particle Penetration
      PENETRATION ORIGIN AND DIRECTION:
```

```
         Option = Specified Origin and Direction
         Injection Centre = 0.01 [m], 0.5 [m], 0.01 [m]
         INJECTION DIRECTION:
            Injection Direction X Component = 1.0
            Injection Direction Y Component = 0.0
            Injection Direction Z Component = 1.0
            Option = Cartesian Components
         END
      END
      Contained Spray Mass Fraction = 0.98
      Particles List = Water 1
      AXIAL PENETRATION:
         Option = Axial Penetration
      END
      RADIAL PENETRATION:
         Option = Radial Penetration
      END
      NORMAL PENETRATION:
         Option = Normal Penetration
      END
      SPRAY ANGLE:
         Option = Spray Angle
      END
   END
   TRANSIENT PARTICLE DIAGNOSTICS: Penetration from PIR
      Option = Particle Penetration
      PENETRATION ORIGIN AND DIRECTION:
         Option = Particle Injection Region
         Particle Injection Region = Cone
      END
      Contained Spray Mass Fraction = 0.98
      Particles List = Water 1, Water 2
      AXIAL PENETRATION:
         Option = Axial Penetration
      END
      RADIAL PENETRATION:
         Option = Radial Penetration
      END
      NORMAL PENETRATION:
         Option = Normal Penetration
      END
      SPRAY ANGLE:
         Option = Spray Angle
         Spray Radius at Penetration Origin = 0.005 [m]
      END
   END
#
#------------------------------------------------------------------------
#   Total Particle Mass
#------------------------------------------------------------------------
#
   TRANSIENT PARTICLE DIAGNOSTICS: Total Particle Mass
      Option = Total Particle Mass
      Particles List = Water 1, Water 2
   END
#
#------------------------------------------------------------------------
# User Defined
#------------------------------------------------------------------------
#
   TRANSIENT PARTICLE DIAGNOSTICS: User Routine Sphere
      Option = User Defined
      Transient Particle Diagnostics Routine = Sphere
      Particle Variables List = \
         Particle Position, \
         Total Particle Mass, \
         Particle Number Rate
      Monitored Values List = \
         First sphere, \
         Second sphere, \
         Third sphere
```

```
      Particles List = Water 1, Water 2
   END
   TRANSIENT PARTICLE DIAGNOSTICS: User Routine Histo
      Option = User Defined
      Transient Particle Diagnostics Routine = Histogram
      Particle Variables List = \
         Total Particle Mass, \
         Particle Number Rate, \
         Mean Particle Diameter
      Particles List = Water 1, Water 2
   END
```

Leading to the following lines in the CFX-Solver Output file and in the CFX-Solver Manager:

```
+--------------------------------------------------------------------+
|                    Transient Particle Diagnostics                  |
+--------------------------------------------------------------------+
Water 1

   User Routine Histo
   User Routine Sphere
      First sphere                          3.8847E-02
      Second sphere                         5.9132E-02
      Third sphere                          7.2068E-02
   Total Particle Mass
      Total Particle Mass                   1.0000E-01
   Penetration from PIR
      Axial Penetration                     7.7220E-01
      Radial Penetration                    8.0765E-01
      Normal Penetration                    2.4617E-01
      Spray Angle                           3.9586E+01
   Penetration from Location
      Axial Penetration                     7.7220E-01
      Radial Penetration                    8.0765E-01
      Normal Penetration                    2.4617E-01
      Spray Angle                           4.6757E+01

Water 2

   User Routine Histo
   User Routine Sphere
      First sphere                          4.5829E-02
      Second sphere                         6.1053E-02
      Third sphere                          7.3941E-02
   Total Particle Mass
      Total Particle Mass                   1.0000E-01
   Penetration from PIR
      Axial Penetration                     7.8456E-01
      Radial Penetration                    9.6701E-01
      Normal Penetration                    6.7314E-01
      Spray Angle                           8.7652E+01
```

8.9.1.2. Particle Track Output

You can use the **Particle Histogram** option under the **Particles** tab of `Output Control` to define particle histogram data of track variables on user-specified boundary patches and/or particle injection regions. For details, see Particle Histogram in the *CFX-Pre User's Guide*.

You can also use the **Particle Track Data** option under the **Export Results** tab in `Output Control` to export a specified list of particle data on specified boundaries or particle injection regions. For details, see Particles Tab in the *CFX-Pre User's Guide*.

8.9.2. List of Particle Variables

When simulating particle-laden flows, several variables referring to the particles are written to the results file and can be used for postprocessing. For details, see Particle Variables Generated by the Solver in the *CFX Reference Guide*.

8.10. Particle Solver Control

The following topics are discussed in this section:

8.10.1. Particle Coupling Control
8.10.2. Particle Under-Relaxation Factors
8.10.3. Particle Integration
8.10.4. Particle Termination Control
8.10.5. Particle Ignition
8.10.6. Particle Source Smoothing
8.10.7. Vertex Variable Smoothing
8.10.8. Particle Source Control

The particle integrator is completely distinct from the normal CFD solver, so separate control parameters are required. The most important control is setting the coupling between the flow calculation and the particle integration—how frequently in terms of flow iterations the particle integration is carried out.

The `Forward Euler` method is the only integration method available. The time step is implicitly determined by the **Number of Integration Steps per Element** setting.

8.10.1. Particle Coupling Control

8.10.1.1. First Iteration for Particle Calculation

This parameter sets the iteration number when particles are first tracked. A few iterations should be completed to let the continuous phase settle down from the initial guess before particles are introduced. If the convergence behavior is not steady by this iteration number, then you can increase the value of the **First Iteration for Particle Calculation** parameter above default, which is 10 in a steady-state simulation and 1 in a transient simulation.

When restarting a run, the iteration number refers to the total iteration number and not that of the current run.

This parameter applies to fully coupled particle phases and transient runs where the specified **First Iteration for Particle Calculation** refers to coefficient loop iterations.

8.10.1.2. Iteration Frequency

Particles are injected during execution of the CFX-Solver at regular iteration intervals. The particles are then tracked using the fluid solution field from the previous iteration. Once the particle paths have been calculated, the particle sources to mass and momentum equations are calculated. The

source terms are applied to the fluid equations at each subsequent iteration until they are recalculated at the next injection.

The **Iteration Frequency** parameter is the frequency at which particles are injected into the flow after the **First Iteration for Particle Calculation** iteration number. This introduces particle source terms that will often cause an immediate rise in the convergence residual levels. The iteration frequency allows the continuous phase to settle down between injections.

The default value for **Iteration Frequency** is 5 in a steady-state simulation and 1 in a transient simulation.

If the continuous phase reaches its convergence criterion before the particle calculation reaches its own convergence criterion, then the code automatically overrides the value specified for the iteration frequency, and starts to track the particles on every iteration.

The parameter applies to fully coupled particle phases and transient runs where the specified iteration interval refers to coefficient loop iterations.

8.10.1.3. Particle Source Change Target

This parameter applies only to fully-coupled particle phases and is used to determine when the particle part of the calculation has converged, as follows:

The absolute value of the particle source for a given momentum equation is summed over all vertices; let this value be SUM. When the inequality Equation 8.3 (p. 406) is satisfied for all momentum equations, the particle part of the calculation is deemed to have converged.

$$\frac{\mathrm{SUM}_{\mathrm{new}}-\mathrm{SUM}_{\mathrm{old}}}{\max\left(\mathrm{SUM}_{\mathrm{new}},\mathrm{SUM}_{\mathrm{old}}\right)}<\text{Particle Source Change Target} \tag{8.3}$$

The default value is 0.01 and it is unlikely that you will need to change this.

8.10.2. Particle Under-Relaxation Factors

This parameter applies only to fully-coupled particle phases. Depending on the flow being solved, particles may introduce very large source terms in the hydrodynamic equations. In reacting flows, large source terms may be generated in the mass, scalar, and energy equations. In heavily laden flows, viscous drag may introduce large source terms in the momentum equations.

In some cases, these source terms may have a destabilizing influence on the convergence of the hydrodynamic equations, resulting in oscillations, or in severe cases, divergence. A simple example of the oscillatory behavior is in a reacting flow field containing burning solid particles. Consider the flow field contains an oxidant into which reacting particles are being injected. If there is sufficient particle mass to consume all of the oxidant, then the oxidant mass fraction approaches zero. The next time the particles are injected, there would be no oxidant, and therefore, no source terms would be generated. The oxidant concentration would return to its original value. Therefore, the oxidant concentration may oscillate between zero and its maximum.

It is possible to minimize the oscillations by under-relaxing the particle source terms. This is done as follows:

$$\boldsymbol{S}_F=\left(1.0-U_F\right)\boldsymbol{S}_F+U_F\,\boldsymbol{S}_P \tag{8.4}$$

where:

$\boldsymbol{S}_F$=Source To Flow
$\boldsymbol{S}_P$=Source From Particles
U_F=Under Relaxation Factor

8.10.2.1. Under-Relaxation Factor for Velocity, Energy, and Mass

In Ansys CFX, particle source terms are generated for the momentum, heat and mass transfer equations. The Under-Relaxation factors for velocity, energy, and mass provide damping for the above equation set. In a steady-state calculation, the default value of `0.75` has been found to be sufficiently small to dampen solutions demonstrating oscillating convergence behavior due to particle sources. In a transient calculation, no under-relaxation is imposed, that is the default values of the under-relaxation factors are `1.0`. If convergence problems are found due to the particle transport coupling, this relaxation factor can be reduced. For details, see Convergence Control for Particle Transport (p. 422).

8.10.2.2. Under-Relaxation Factor for First Particle Integration

This relaxation factor is applied to particles sources into the fluid phase, if no old particle sources exist. By default, this factor is set to `0.75` for steady-state runs and `1.0` for transient runs.

8.10.2.3. Under-Relaxation at Time Step Start

By default, no relaxation is applied to particle sources between time steps in a transient simulation. If convergence problems are found due to the particle transport coupling, this relaxation factor can be reduced.

8.10.3. Particle Integration

8.10.3.1. Number of Integration Steps Per Element

This setting controls the accuracy and time step of the particle tracking integration. A time step is chosen locally as the element length scale divided by the particle speed divided by the number of integration steps per element. The integration accuracy will be increased by using a higher number of integration steps at the expense of computational time. The default value of 10 is usually suitable.

8.10.3.2. Maximum Particle Integration Time Step

This parameter can override the automatic time step obtained from the element size and the particle velocity. You might need to use it in bubbly flows with the virtual mass force being used.

8.10.3.3. Chemistry Time Step Multiplier

When doing a calculation with multiphase reactions, the values of reactants decrease with time. If too big a time step is used, the values of reactants can become negative. To prevent this, a bound is placed on the particle integration time step. The Chemistry Time Step Multiplier is a factor (default of 1.0) that is bounded by the time it takes for a mass fraction to reach zero. If a smaller value is

used, then the mass fraction will go towards zero. The need for this parameter is uncommon for most applications.

8.10.4. Particle Termination Control

8.10.4.1. Maximum Tracking Time

This is the real time during which the particles are integrated. This should be set to a time long enough for a particle to be tracked through the geometry but not too large or the computational cost of tracking particles that may become trapped in recirculation zones could become exceedingly large. The default value is 10 [s].

8.10.4.2. Maximum Tracking Distance

This is the distance over which particles are integrated. It should be large enough to enable a particle to be tracked through the geometry, but not too large or the computational cost of tracking particles which may become trapped in recirculation zones could become exceedingly large. The default value is 10 [m].

8.10.4.3. Maximum Number of Integration Steps

This is another control that can be used to terminate tracking of particles that may become trapped in recirculation zones. The number of integration steps is calculated as the number of integration steps per element multiplied by the number of elements crossed by a particle. The default value is 10,000. You can increase this number if you believe it will be exceeded by a particle following a normal route through the simulation.

8.10.4.4. Minimum Diameter

This parameter can be used for specifying the minimum allowed particle diameter. If the diameter falls below this value, then the tracking of particle is stopped. This parameter can be used for excluding particles below the specified minimum diameter, which no longer play any significant role in the simulation, and therefore improves the computational time. The default value is 10^-8 [m].

8.10.4.5. Minimum Total Mass

This parameter can be used for specifying the minimum allowed particle mass. If the particle mass drops below this value, then the tracking of particle is stopped. This parameter can be used for excluding particles below the specified minimum total mass, which no longer play any significant role in the simulation, and therefore improves the computational time. The default value is 0 [kg].

8.10.4.6. Mass Fraction Limits

For multi-component particles, parameters **Minimum Mass Fraction** and **Maximum Mass Fraction** can be used for specifying upper and/or lower limits for a particle component mass fraction together with the **Component Name** of the particle. If the mass fraction gets below or above the specified value, then tracking of the particle is stopped.

In the CFX-Pre user interface, you can specify the reference component name in the form of `<particle type>.<component name>`, for example, `Coal1.Ash`. Using this form restricts the use of mass fraction limits to the specified particle type only.

8.10.5. Particle Ignition

Particle Ignition is used to start the initial combustion process of reacting particles, which typically depends on the temperature of the Eulerian fluid phase around the particle. However, before the reactions are calculated, the fluid temperature will often be low. Therefore, this temperature can be temporarily increased to the specified **Ignition Temperature** the first time a particle is solved. Afterwards, the particle combustion itself should deliver enough energy in order to increase the fluid temperature above the required ignition temperature. The **Particle Ignition** capability is only available for steady-state simulations.

8.10.6. Particle Source Smoothing

A smoothing procedure similar to **Vertex Variable Smoothing** can be applied to the particle source terms, which are accumulated along the path of a particle through a control volume and stored at the corresponding vertex. This may help with convergence or grid independence.

Once **Particle Source Smoothing** is selected, choose a smoothing **Option**: `Raw` (no smoothing) or `Smooth`.

8.10.7. Vertex Variable Smoothing

Quantities of particles crossing a control volume are averaged and stored at the corresponding vertex. Due to the distribution of particle tracks in the domain this can result in non-contiguous vertex fields. The use of those vertex fields, for example in a CEL expression, could cause numerical problems. In these cases it may be helpful to smooth the particle vertex arrays. Only direct neighbors are involved in the smoothing process.

Once **Vertex Variable Smoothing** is selected, choose a smoothing **Option**: `Raw` (no smoothing) or `Smooth`.

8.10.8. Particle Source Control

For simulations with high particle loadings it might be necessary to limit the particle sources into the continuous phase to help avoid either convergence problems or solver crashes due to unphysical fluid states.

For steady-state simulations, you can limit the source terms for particles by:

- Particle Heat Source Bounding (p. 410)
- Particle Momentum Source Bounding (p. 411)
- Linearization of Particle Mass Sources (p. 412)

For transient simulations, you can limit the source terms for particles only by Linearization of Particle Mass Sources (p. 412).

Note:

For transient simulations, there is currently no option for limiting particle heat and momentum source terms.

Note:

The application of source term bounding does not guarantee that a simulation will converge.

The following topics are discussed in this section:

8.10.8.1. Particle Heat Source Bounding
8.10.8.2. Particle Momentum Source Bounding
8.10.8.3. Linearization of Particle Mass Sources
8.10.8.4. Particle Source Control Usage Notes

8.10.8.1. Particle Heat Source Bounding

For energy sources due to convective heat transfer, you can limit the source terms by setting **Option** to `Correction Factor Bounding`.

You can use the **Stop Bounding after Iteration** option to limit the number of iterations that have source term bounding. The default value for this option is 40 iterations.

The particle convective energy source in the gas phase can be written as:

$$S_{p,E}=q\left(T_p-T_f\right) \tag{8.5}$$

where

T_p is the particle temperature,

T_f is the local fluid temperature, and

q is a transfer coefficient.

Equation 8.5 (p. 410) is solved in the particle tracker at each particle iteration step and the accumulated sources are stored at a place where they are later on picked up in the energy assembly of the gas phase. By default, particles are not tracked after every fluid flow simulation, but in user defined intervals (default: every 5th fluid step in a steady-state run). Whenever the particle tracker is run, new values for the particle sources are generated and these sources persist up to the next time particles are tracked. If the particle sources are just large enough, it might not be sufficient to just limit the sources once, when they are generated, but also every time they are picked up and used by the gas phase energy equation. This somehow mimics a 'low cost' particle tracking step, where only the energy sources are updated to account for changes in the fluid temperature (enthalpy) due to energy sources applied in previous flow solver steps.

The basis of the suggested modification is the equation for the particle convective energy source, Equation 8.5 (p. 410). This equation is reformulated as follows:

$$S_{p,E}=q\left(\overline{T}_p-T_{fl}\right) \tag{8.6}$$

where

$\overline{T}_p$ is the (average) particle temperature,

T_{fl} the fluid temperature, and

q is a transfer coefficient.

Building the ratio of S_E^n/S_E^o allows approximating the new value of the particle source term for the current iteration, taking the fluid temperature change from the 'old' (o) to the current time step (n) into account:

$$\frac{S_{p,E}^n}{S_{p,E}^o}=\frac{\overline{T}_p-T_{fl}^n}{\overline{T}_p-T_{fl}^o} \tag{8.7}$$

Note that in the above formulation the average particle temperature, $\overline{T}_p$ from the last tracking step is used. This quantity is not changing in between successive tracking steps. $S_{p,E}^o$ and T_{fl}^o are updated every time step.

The correction factor, $\overline{T}_p-T_{fl}^n/\overline{T}_p-T_{fl}^o$, is limited to be in the range of [0,1]. This correction is applied before the particle source term is used by the gas phase energy equation.

Note:

The energy source due to mass transfer is not taken into account.

8.10.8.2. Particle Momentum Source Bounding

For momentum sources due to fluid drag, you can limit the source terms by setting **Option** to `Correction Factor Bounding`.

You can use the **Stop Bounding after Iteration** option to limit the number of iterations that have source term bounding. The default value for this option is 40 iterations.

Similar considerations as outlined in Particle Heat Source Bounding (p. 410), can be made to limit the particle momentum sources to the gas phase. Under the assumption that the particle drag is the major contributor to the particle momentum source term, it is possible to write the particle momentum source as:

$$S_{p,M}=qC_d\left|\vec{U}_S\right|\left(\vec{U}_p-\vec{U}_{Fl}\right) \tag{8.8}$$

where

C_d is the particle drag coefficient,

$\left|\vec{U}_S\right|$ is the particle slip velocity,

$\vec{U}_F$ is the fluid velocity vector,

$\vec{U}_p$ is the particle velocity vector, and

q is a coefficient, dependent on the fluid density, ρ_{Fl}.

Forming the ratio of $S^n_{p,M}/S^o_{p,M}$ and rearranging the resulting equation for $S^n_{p,M}$ gives:

$$S^n_{p,M}=S^o_{p,M}\frac{\rho^n_{Fl}}{\rho^o_{Fl}}\frac{C^n_d}{C^o_d}\frac{\left|\vec{U}^n_S\right|}{\left|\vec{U}^o_S\right|}\frac{\left(\vec{U}_P-\vec{U}_{Fl}\right)^n}{\left(\vec{U}_p-\vec{U}_{Fl}\right)^o} \tag{8.9}$$

Assuming that the drag coefficient, C_d, is determined by the Schiller-Naumann drag law

$$C_d=\frac{24}{Re_p}\left(1+0.15Re^{0.687}\right) \tag{8.10}$$

with

$$Re=\frac{d_p\left|U_s\right|}{\nu_{Fl}}$$

being the particle Reynolds number computed from the slip velocity between particles and fluid, the particle diameter and the fluid kinematic viscosity. Inserting the particle Reynolds number into Equation 8.10 (p. 412) it is possible to simplify Equation 8.9 (p. 412) to give:

$$S^n_{p,M}=S^o_{p,M}\frac{\rho^n_{Fl}}{\rho^o_{Fl}}\frac{1+0.15\left|\vec{U}^n_S\right|^{0.687}}{1+0.15\left|\vec{U}^o_S\right|^{0.687}}\frac{\left(\vec{U}_P-\vec{U}_{Fl}\right)^n}{\left(\vec{U}_p-\vec{U}_{Fl}\right)^o} \tag{8.11}$$

Equation 8.11 (p. 412) is used to determine a factor that limits the particle momentum source by taking fluid velocity and density changes into account. The correction factor is limited to be in the range of [-1,1].

8.10.8.3. Linearization of Particle Mass Sources

In Ansys CFX, particle mass sources are, by default, not linearized with respect to the fluid mass fraction (mass fraction equations) or the fluid pressure (continuity equation). For cases with relatively small mass transfer rates, linearization of the particle mass sources may not play an important role. This may be different, though, for cases where the particle mass transfer is large enough to significantly affect the fluid energy and/or the fluid momentum. In these cases, the use of mass source linearization may be crucial for a successful simulation run.

Linearization of particle mass sources can be included by setting **Option** to `Source Coefficient with respect to Mass Fraction`. With this setting, the derivatives $\frac{d\dot{m}_c}{dY_{F,c}}$ in the following subsections are included in the linearization of sources in the equation for mass of the particle component (see Interphase Transfer Through Source Terms in the *CFX-Solver Theory Guide*).

Note:

Linearization is not available for multiphase reaction and combustion cases.

8.10.8.3.1. Simple Mass Transfer Model

Each component of mass being transferred between the continuous and particle phases satisfies the equation:

$$\dot{m}_c = -\pi d_p \rho_F D_F Sh \left(E \cdot Y_{p,c} - Y_{F,c} \right) \tag{8.12}$$

In Equation 8.12 (p. 413), m_c is the mass of the constituent in the particle, $Y_{p,c}$ is the mass fraction of component c in the particle, $Y_{F,c}$ is the mass fraction of component c in the surrounding fluid, E is the equilibrium mass fraction ratio, $\rho_F D_F$ is the dynamic diffusivity of the mass fraction in the continuum, and Sh is the Sherwood number.

For the mass fraction equation of species, c, the derivative of Equation 8.12 (p. 413) with respect to $Y_{F,c}$ is required. This derivative can be written as:

$$\frac{d\dot{m}_c}{dY_{F,c}} = \pi d_p \rho_F D_F Sh \tag{8.13}$$

Equation 8.13 (p. 413) is used to compute the source term coefficient in the gas phase mass fraction equation.

The derivative of the mass transfer rate with respect to the fluid pressure is zero, because the mass transfer rate does not explicitly depend on the fluid pressure.

8.10.8.3.2. Liquid Evaporation Model

The liquid evaporation model is a model for particles with heat transfer and mass transfer, and in which the continuous gas phase is at a higher temperature than the particles. The model uses two mass transfer correlations depending on whether the droplet is above or below the boiling point. This is determined through an Antoine equation.

8.10.8.3.2.1. Droplet Temperature Below Boiling Point

The mass transfer rate from the particle to the gas phase is determined by:

$$\dot{m}_c = \pi d_p \rho_F D_F Sh \frac{W_c}{W_{Fl}} log\left(\frac{1 - X^V_{S,c}}{1 - X^V_{vap,c}} \right) \tag{8.14}$$

where m_c is the mass of the constituent in the particle, $\rho_F D_F$ is the dynamic diffusivity of the mass fraction in the continuum, and Sh is the Sherwood number. W_c and W_{Fl} are the molecular weights of the vapor and the mixture in the continuous phase, $X^V_{vap,c}$ is the molar fraction in the gas phase, and $X^V_{S,c}$ is the equilibrium mole fraction at the droplet surface defined as the component vapor pressure divided by the pressure in the continuous phase.

The derivative of Equation 8.14 (p. 413) with respect to the fluid component mass fraction is:

$$\frac{d\dot{m}_c}{dY_{F,c}} = \pi d_p \rho_F D_F Sh \frac{1}{1 - \frac{X_{F,c} W_{Fl}}{W_c}} \tag{8.15}$$

As with the simple mass transfer model, there is no linearization applied with respect to the fluid continuity and volume fraction equations.

8.10.8.3.2.2. Droplet Temperature Above Boiling Point

If the particle is boiling, the mass transfer into the gas phase is driven by convective heat transfer and radiative heat transfer to the particle:

$$\dot{m}_c = -\frac{Q_{conv} + Q_{rad}}{V} \tag{8.16}$$

Q_{conv} and Q_{rad} are the rates of convective heat transfer and radiative heat transfer to the particle, respectively. V is the latent heat of vaporization. The mass transfer rate in the boiling regime does not depend on the fluid component mass fraction, Y_c, nor on the fluid pressure, p. Therefore both derivatives are zero:

$$\frac{d\dot{m}_c}{dY_c} = \frac{d\dot{m}_c}{dp} = 0 \tag{8.17}$$

8.10.8.4. Particle Source Control Usage Notes

- Cases with mass transfer (Ranz Marshall, Liquid Evaporation model) that show robustness problems should use particle mass source linearization rather than bounding heat and momentum sources only. In many cases, unphysical temperatures and/or flow fields are the result of overly large mass transfer rates and can be addressed most effectively by linearizing the particle mass sources.
- Note that **Particle Heat Source Bounding** only bounds sources that are due to convective heat transfer. In cases where convective heat transfer is small compared to other heat sources (for example, reactions, radiation), this option might not be sufficient to guarantee bounded fluid temperatures.
- Note that **Particle Momentum Source Bounding** only bounds sources that are due to particle drag. In cases where the momentum sources due to particle drag are small compared to other momentum sources (for example, pressure gradient) this option might not be sufficient to guarantee bounded fluid velocities.
- Note that particle source term bounding is applied over only a certain number of flow iterations (default: 40). Once this limit is reached, source term bounding is tuned off. In some cases, this limit might not be sufficient to let the flow solver establish a reasonable flow field; in such cases, the limit must be increased. The limiting iteration number can be set via the **Stop Bounding after Iteration** setting.
- In many cases where it is necessary to use a source term bounding option, it might also be necessary to set the expert parameter `pt fluid var interpolation option` to `0`. This setting should only be used to improve robustness and is no longer required once a stable flow field solution has been established. For details on this expert parameter, see Particle Tracking Parameters (p. 622).

8.11. Multiphase Reactions and Combustion

Multiphase Reactions refer to reactions involving components in different phases, and is a combination of simultaneous phase change and conversion of some materials into others. The capability is available for particle tracking calculations, but not `Eulerian-Eulerian` multiphase flows for this release of Ansys CFX. When setting up a multiphase reaction, the **Combustion Model** setting in the **Reaction** details view is not required because `Finite Rate Chemistry` is the only available option for

multiphase reactions (although for gas phase reactions, other reaction/combustion models may be applied).

A general multiphase reaction generates three classes of fluxes:

- Component Mass Sources
- Interphase Mass Transfer
- Interphase Energy Transfer

Multiphase reactions are specified in the **Reaction** details view by setting the **Option** to `Multiphase` on the **Basic Settings** form. For details, see Multiphase: Basic Settings in the *CFX-Pre User's Guide*.

8.11.1. Specification of a Binary Mixture

You should define a homogeneous binary mixture (see Material Details View: Homogeneous Binary Mixture in the *CFX-Pre User's Guide*) that links the primary particle and gas phase materials. Primary materials are the materials that characterize the mass transfer between the particle and the gas phase. For example, when modeling decomposition of raw coal into gaseous volatiles and char, the homogeneous binary mixture should be set up for 'Raw Coal' and 'Volatiles'. This allows the particle solver to calculate characteristic numbers for the particle (for example, Reynolds, Sherwood, and Nusselt numbers) based on properties in the particle boundary layer. These properties are calculated using the Antoine equation (vapor pressure equation) that is specified in the homogeneous binary mixture.

8.11.2. Reactants/Products

Reactants and products are specified in terms of parent and child materials. The parent material defines the phase in which species react, and the child materials represent the reacting species. For this release of Ansys CFX, reactants from different phases can only be specified for the char oxidation reactions (`Field Char Oxidation` or `Gibb Char Oxidation`). Arrhenius type multiphase reaction rates require that all reactants belong to the same phase. The stoichiometry for multiphase reactions is specified by means of mass coefficients rather than stoichiometry coefficients referring to moles.

8.11.2.1. Example

As an example, consider a simplified case of coal reacting with oxygen in the reaction $C+O_2 \rightarrow CO_2$. The **Parent Materials List** on the **Reactants** tab contains two materials, `Coal` (a pure substance containing carbon) and `Gas mixture` (a variable composition mixture containing `O2`, `CO2` and `N2`). The child material for `Coal` would be set to `Carbon`, and the child material for `Gas Mixture` would be set to `O2`. It follows that on the **Products** tab, the **Materials List** contains `Gas Mixture` as the parent material, which in turn would contain `CO2` as its child.

Note:

The above is presented only as an illustration of the use of the parent/child material structure.

8.11.3. Multiphase Reactions

The **Multiphase Reactions** tab is used to define the reaction properties for three different types of reaction rates. The `Arrhenius` option uses a very similar implementation for multiphase reactions as for single phase reactions. For details, see Arrhenius (p. 427). Because reactants using Arrhenius reaction type must all come from the same phase in this release of Ansys CFX, the temperature required to calculate the Arrhenius rate comes from the single permitted phase. Additional information on heat release is available in Heat Release/Heat Release Distribution (p. 417).

Typical coal combustion reactions also require a model for char oxidation. Char oxidation requires higher temperatures than devolatilization and therefore occurs after devolatilization. In Ansys CFX, char oxidation is modeled either as a global reaction of order unity (Field) or using a simple analytic approach to the diffusion of oxygen within the pores of the char particles (Gibb).

8.11.3.1. Mass Arrhenius

The `Mass Arrhenius` option is the same implementation as used for single-phase reactions. For details, see Arrhenius (p. 427).

8.11.3.2. Field Char Oxidation Model

Theory documentation for this model is available in Field in the *CFX-Solver Theory Guide*.

In the field model, a char particle is considered to be a spherical particle surrounded by a stagnant boundary layer through which oxygen must diffuse before it reacts with the char. The oxidation rate of the char is calculated on the assumption that the process is limited by the diffusion of oxygen to the external surface of the char particle and by the effective char reactivity. Additional information on Heat Release is available in Heat Release/Heat Release Distribution (p. 417).

8.11.3.3. Gibb Char Oxidation Model

Theory documentation for this model is available in Gibb in the *CFX-Solver Theory Guide*.

The alternative char oxidation model, the Gibb model, takes into account the diffusion of oxygen within the pores of the char particle. The parameters required for this model include the void fraction ε of the char particle, the particle volume/internal surface area ratio a, the effective internal diffusion coefficient D_p of oxygen within the pores, and the molar ratio φ of carbon atoms/oxygen molecules involved in the oxidation processes. The molar ratio is determined by the $CO \leftrightarrow CO_2$ equilibrium, and the relevant input data is provided by the values entered in the **Char Product Ratio** area.

8.11.3.4. Particle User Routine

This option allows you to use a User Fortran subroutine to calculate the reaction rate of a multiphase reaction. General information on creating user routines is available in User Fortran (p. 631).

In the `<install_dir>\examples\UserFortran` directory, you can find a reaction rate routine, `pt_reaction.F`, and corresponding CCL template, `pt_reaction.ccl`.

8.11.3.5. Heat Release/Heat Release Distribution

Heat release specifies how much heat is released during a multiphase reaction, and heat release distribution specifies how the heat is distributed over the participating phases. There are two principal methods that define the amount of heat released; it can either be calculated from the material reference enthalpies, or can be user specified.

If the **Heat Release** option is set to `From Material Properties`, the heat release is calculated from material reference enthalpies. When set to `Specific Enthalpy`, the value of heat released per unit mass of reactants must be specified. Because none of the reactants are known to Ansys CFX as the "fuel" component at this stage, you must also select the parent phase and reference material, so that the heat release is correctly related to the mass of the true fuel material. The Latent Heat option is equivalent to the `Specific Enthalpy` setting, except that the value for heat release is specified with the opposite sign.

The `From Material Properties` option for **Heat Release** is advantageous because the calculation guarantees the conservation of energy (that is, if a fuel and oxidant react to form some products by two separate competing reactions, then the amount of heat released automatically is the same for both reaction paths). The disadvantage is that correct reference specific enthalpies must be found and entered when the material properties are defined. Reference specific enthalpies can be difficult to determine, especially for materials that are not pure substances but mixtures of inconsistent definition (such as coal). In such cases, it is much more convenient to specify the heat release for the reaction. Heat releases for reactions can be measured directly and are therefore usually available for a given application. When specifying heat release directly, however, you must ensure that you define consistent values for heat release in the case of competing reactions or reaction paths.

The distribution of released heat to the participating phases is defined by specifying the fraction of total heat release that the individual phases receive. When setting the **Heat Release Distribution** parameters, the parent materials list must contain only materials that occur in reactants and/or products. For each phase selected, a value fraction of heat released must be specified (for example, for a gas phase receiving 25% of the heat released, the value would be 0.25). The total of all heat fractions should sum to unity.

8.11.4. Hydrocarbon Fuel Model Setup

The hydrocarbon fuel model offers a simple method of creating a solid particulate or liquid droplet combustion model starting from the ultimate and proximate analysis of the fuel. It uses the Lagrangian particle transport model to track representative fuel particles and uses the Eddy Dissipation model and/or the Finite Rate Chemistry model for the combustion of the volatile gases in the gas phase. Devolatilization (or pyrolysis) of the hydrocarbon fuel particles and oxidation of the resultant char are modeled by semi-empirical kinetic models with parameters that depend on the fuel composition and a standard fuel analysis. The model was mainly designed for the modeling of pulverized coal combustion but can also be applied to combustion of other solid hydrocarbon fuels in particulate form or liquid sprays.

This section describes the setup for the hydrocarbon fuel model. Three variants are covered here:

- Proximate/ultimate fuel analysis starting from the standard library template
- Proximate/ultimate fuel analysis when setting up manually

- Using generic multiphase reactions set-up

The first method, using the fuel analysis directly, is recommended. The third method is mainly included for compatibility with earlier releases of the software.

8.11.4.1. Setup using Library Template (Recommended)

The recommended way of setting up the hydrocarbon fuel model is to start by loading from the library provided. This approach uses the full functionality of the model, but avoids unnecessarily repeating the definition of materials and reactions. All fuel dependent data, like material composition and heat release, is gathered in the hydrocarbon fuel material definition. Therefore, after loading the library, most objects do not need to be changed by you. In most cases, it will be sufficient to only edit the hydrocarbon fuel material object.

The pre-defined multiphase reactions assume certain names for the hydrocarbon fuel material and for the gas mixture. The name of the hydrocarbon fuel is `HC Fuel`. The gas mixture material name depends on the fuel-nitrogen model: `Gas Mixture HCN NO` with the fuel-nitrogen model; and `Gas Mixture` without.

The hydrocarbon fuel model without the fuel-nitrogen sub-model can be set up using the provided library as follows:

1. From either the `Materials` or the `Reactions` object, select **Import Library Data** from the context menu to import all objects from the hydrocarbon fuel library `Hydrocarbon_Fuel.ccl` located in the `reactions-extra/` directory. Additional materials and reactions will be included automatically from the standard libraries. Note that when loading the `Hydrocarbon_Fuel.ccl` file using the general **File > Import CCL...** route, then some additional materials and reactions will not be loaded automatically, therefore causing corresponding physics error messages. In this case the missing materials and reactions need to be loaded manually in order to resolve the error messages.
2. Edit the imported material `HC Fuel`. The parameters on the **Proximate/Ultimate Analysis** tab need to be adjusted in order to match the actual fuel specification:
   a. Proximate analysis
      - As Received (mass fractions sum to 1)
      - Dry Ash Free (fixed carbon and volatiles sum to 1)

   b. Ultimate analysis
      - As Received (mass fractions sum to 1 – ash – moisture)
      - Dry Ash Free (mass fractions sum to 1)

   c. Heating value

   d. Volatiles yield enhancement (ratio of actual volatile yield to proximate yield) - Defaults for remaining parameters

3. Create the domain:
   a. Fluid material: `Gas Mixture`
   b. Particle material `HC Fuel`
   c. Multiphase reactions:
      - `HC Fuel Devolat`
      - `HC Fuel Char Field` or `HC Fuel Char Gibb`
   d. Heat transfer model fluid dependent:
      - Fluid: `Thermal` or `Total Energy`
      - Particle: `Particle Temperature`
   e. Combustion model fluid dependent:
      - Fluid: `Eddy Dissipation`
   f. Radiation model fluid dependent:
      - Fluid: `P1`, `Discrete Transfer`, or `Monte Carlo`
   g. Fluid component details:
      - **N2**: `Constraint`
      - Others: `Automatic`
   h. Particle diameter change during pyrolysis (Fluid: `HC Fuel`):
      - **Option** = `Swelling Model`
      - **Reference Material** = `Raw Combustible`
      - **Swelling Factor** = <real> # 0.0 for constant diameter
   i. Thermal radiation transfer (Fluid Pair: `HC Fuel | Gas Mixture HCN NO`):
      - **Option** = `Blended Particle Emissivity`
      - **Reference Material** = `Raw Combustible`
      - **Base Emissivity** = <real> # Emissivity of char and ash
      - **Blend Emissivity** = <real> # Emissivity of raw combustible

4. Set particle ignition under **Solver Control** / **Particle Control**:

   - **Particle Ignition** / **Ignition Temperature** = `1000 [K]`

No boundary conditions for particle components will be defined. The remaining setup is the same as for pure-material particles and a variable composition fluid.

The fuel-nitrogen model can be enabled by the following changes to the above procedure:

1. Edit the `HC Fuel` material. On the **Mixture Materials** tab change the option in the **Gas Mixture section**:

   a. **Gas Mixture/Option** = `Mixture with HCN NO`

   b. **Mixture Material** = `Gas Mixture HCN NO`

   c. Component materials:

      - HCN Material = HCN
      - HCO Material = HCO
      - NO Material = NO

   d. Remaining parameters unchanged

2. Change the fluid and the multiphase reactions on the **Edit Domain** form:

   a. Fluid material `Gas Mixture HCN NO`

   b. Multiphase reactions:

      - `HC Fuel Devolat HCN`
      - `HC Fuel Char Field HCN` or `HC Fuel Char Gibb HCN`

8.11.4.2. Set Up Manually (Experts)

The manual set up is usually not required and is intended for expert users only. For this procedure, the definition of material and reaction objects is completely left to you. When doing so, keep in mind that the hydrocarbon fuel model can detect only certain types of reactions for computation of mass coefficients and stoichiometric coefficients. The auto-computed coefficients are reported to the CFX-Solver Output file and should be checked for consistency.

In order to set up the coal model from scratch, the following additional steps are required to create the required materials and reactions:

1. Create component materials:

   a. Gas phase components: `Fuel Gas`, `O2`, `CO2`, `H2O`, `N2`, …, `NO`, `HCN`, `HCO`

b. Hydrocarbon fuel components: Ash, Char, Raw Combustible

2. Create gas phase reactions:

   a. Fuel gas oxidation

   b. Pollutants reactions (for example, for NO)

3. Create mixture materials:

   a. Homogeneous binary mixture: Fuel Gas, Raw Combustible

   b. Gas phase reacting mixture: components and reactions previously created

   c. Hydrocarbon fuel: Proximate/ultimate analysis data; homogeneous binary mixture material; gas mixture material

4. Create multiphase reactions:

   a. Devolatilization (with or without fuel nitrogen)

   b. Char oxidation (with or without fuel nitrogen)

From this point on the setup can be continued as described in section Setup using Library Template (Recommended) (p. 418), with the materials and reactions replaced with those created locally.

8.11.4.3. Using Generic Multiphase Reactions Setup

A third way of specifying combustion of a hydrocarbon fuel is to use the generic multiphase multicomponent and reaction machinery. In this case, it is up to you to set the appropriate particle component boundary values, stoichiometric coefficients for reactions, and heat release for the gas phase reactions (by means of volatiles reference enthalpy).

8.12. Restrictions for Particle Transport

The following limitations apply to the Lagrangian particle tracking model in Ansys CFX:

- When combined with the `Eulerian-Eulerian` multiphase model, only a single continuous phase may contain particles. There is no exchange of momentum between particles and a dispersed Eulerian phase.
- There are no particle source terms to the turbulence equations; therefore, turbulence is not modulated by the discrete phase.
- Particle interactions might be important in flows where the discrete phase volumetric concentration is greater than 1% (ref. [67]). In such cases, the solid particle-particle interaction should be used.
- You cannot create Additional Variables in particles.

- In steady-state simulations that involve mesh movement, particles are still solved at the user-specified intervals. For accuracy reasons we recommend to solve particles after each time step by setting the solution interval to '1'. Note that this will also ensure that particle mass flows through GGI interfaces will always be written to the results file.

8.13. Restrictions for Particle Materials

A particle may consist of an arbitrary number of different materials. For each material the following restrictions apply:

- Only constant value properties are supported for density, viscosity, and conductivity.
- For material specific heat values, either constant values, Zero Pressure Polynomials, or NASA polynomials are supported.
- For multicomponent particles, only ideal mixtures are supported.

8.14. Convergence Control for Particle Transport

Within the particle transport model, the total flow of the particle phase is modeled by tracking individual particles from their injection point until they escape the domain or some integration limit criterion is met. The fluid affects the particle and conversely, there is a counteracting influence of the particle on the fluid flow. This effect is termed coupling between the phases. Particle source terms are generated for each particle as they are tracked through the flow. The sources are applied in the control volume that the particle is in during the time step.

Depending on the flow being solved, particles may introduce very large source terms to the hydrodynamic equations. In reacting flows, large source terms may be generated in the mass, component mass fraction, and energy equations. In heavily laden flows, viscous drag may introduce large source terms in the momentum equations. In some cases, these source terms may have a destabilizing influence on the convergence of the hydrodynamic equations, resulting in oscillations, or in severe cases, divergence.

A possibility to minimize the oscillations is to under-relax the particle source terms more strongly (that is, to decrease the values of the following CCL parameters from their defaults of 1.0 to a smaller value):

Source Term	**CCL Parameter**
Momentum	`Velocity Under-Relaxation Factor`
Energy	`Energy Under-Relaxation Factor`
Mass	`Mass Under-Relaxation Factor`

The effects of the sources on the continuous phase are often linearizable, because drag depends upon the fluid velocity, and so on. The linear coefficient is calculated as the derivative of the source with respect to the relevant fluid variable (for example, dS/dVelF for momentum). Sometimes, it is useful for convergence to multiply this linear coefficient by a factor (<10) using the expert parameter: `pt linear src coef multiplier`.

In a rapidly changing fluids solution, you may want to increase the iteration interval from its default value of 5, updating tracks and sources less often, so that the flow has a chance to relax between successive calls to the particle solver.

The particle source terms acting on the fluid phase are proportional to the `Particle Number Rate` (that is, the number of physical particles that a computational particle represents). The convergence of a simulation can be improved if a sufficiently large number of particles are tracked.

In addition to the already mentioned steps, it can be helpful to start a particle transport simulation with less than the full particle mass flow rate, and gradually increase the mass flow rate using a ramping function.

8.15. Expert Parameters for Particle Tracking

Expert parameters can be used to control the behavior of the particle solver in Ansys CFX. For the full list of expert parameters for particle tracking, see Particle Tracking Parameters (p. 622).

8.16. Particle Diagnostics

Additional particle diagnostics output is available in the CFX-Solver Output file and monitor file of the CFX-Solver. This information helps to evaluate the solution process of the particles. It delivers integrated flows of the particle mass, momentum, and energy in and out of the domain and shows the global influence of the particles on the continuous phase. The output provides information about the convergence of the particle solver and summarizes the fate information for all injected particles

For additional information, see Particle Fate Diagnostics in the *CFX-Solver Manager User's Guide*.

8.17. Integrated Particle Sources for the Coupled Continuous Phase

The global balances for the coupled continuous phase also contain the contribution from the coupled particle types.

8.18. Transient Simulations: What is Different for Particles?

Although this topic has been discussed in many areas of the particle tracking modeling documentation, this section attempts to summarize the most important differences between steady-state and transient runs with respect to the particle solver.

In a **transient run:**

- You have to specify the number of particles injected per time step and not the total number of injected particles.
- Iteration control of the particle solver refers to the coefficient loop within a time step and not to the outer loop iteration.
- Different under-relaxation coefficients are used for the generated particle sources.

- The CFX-Solver Output file of a transient particle run contains additional information about time integrated particle boundary fields, mass, momentum, and energy flows.

Chapter 9: Combustion Modeling

CFX includes combustion models to enable the simulation of flows in which combustion reactions occur. The following models are currently available:

- The eddy dissipation model (EDM)
- A finite rate chemistry model (FRC)
- A combined FRC/EDM model
- The laminar Flamelet model for diffusion flames
- A model for premixed or partially combustion using the Flamelet model for the burned mixture.

The eddy dissipation model was developed for use in a wide range of turbulent reacting flows covering premixed and diffusion flames. Because of its simplicity and robust performance in predicting turbulent reacting flows, this model has been widely applied in the prediction of industrial flames.

The finite rate chemistry model enables the computation of reaction rates described by the molecular interaction between the components in the fluid. It can be combined with the eddy dissipation model for flames where chemical reaction rates might be slow compared with the reactant mixing rates. It also enables the use of user-customized reaction rate expressions via CFX Expression Language (CEL).

These models determine the rates at which a component is consumed or created in a single reaction step during the combustion process. For multiple step reactions, the reactions are added to obtain the total reaction rate.

The flamelet model can provide information on minor species and radicals such as CO and OH, and accounts for turbulent fluctuations in temperature and local extinction at high scalar dissipation rates, for the cost of solving only two transport equations.

The burning velocity model (BVM) and the extended coherent flame model (ECFM) model the propagation of a premixed or partially premixed flame by solving a scalar transport equation for the reaction progress. The BVM uses an algebraic correlation for modeling the turbulent burning velocity (propagation speed of the flame in turbulent flow). When using the ECFM, the turbulent burning velocity is closed by solving an additional transport equation for the flame surface density.

Both the BVM and ECFM are combined with the flamelet model in order to describe the composition and properties of the burnt mixture. For partially premixed cases the flamelet model also models diffusion flame 'tails', which may occur in the burnt mixture behind the flame front.

In the CFX-Solver, a limit has not been set for the maximum number of reaction steps or components in a fluid. However, the larger the number of components, the larger the memory requirements will be.

Note:

You should be familiar with the multicomponent flow chapter before reading this chapter; for details, see Multicomponent Flow (p. 64). Multiphase reactions are also available in CFX; for details, see Multiphase Reactions and Combustion (p. 414).

This chapter describes:

9.1. Reaction Models
9.2. Using Combustion Models
9.3. Eddy Dissipation Model
9.4. Finite Rate Chemistry Model
9.5. Combined EDM/Finite Rate Chemistry Model
9.6. Reaction-Step Specific Combustion Model Control
9.7. Laminar Flamelet with PDF Model
9.8. Burning Velocity Model (Premixed or Partially Premixed)
9.9. Extended Coherent Flame Model (ECFM)
9.10. Residual Material Model
9.11. Flamelet Libraries
9.12. Autoignition Model
9.13. NO Model
9.14. Chemistry Postprocessing
9.15. Soot Model
9.16. Phasic Combustion
9.17. General Advice for Modeling Combusting Flows in CFX

9.1. Reaction Models

The following information relates to the setting up of reactions using the **Reaction** details view. For details, see Reactions in the *CFX-Pre User's Guide*.

9.1.1. Naming Convention for Reaction Schemes

For the predefined reaction schemes provided in the REACTIONS library, the following conventions apply:

- Reactions with WD *n* in the name correspond to the appropriate Westbrook and Dryer [66] reaction scheme with *n* number of steps. Note that this publication contains misprints for certain reaction rates [220], which have been confirmed by the authors of the original paper. We are using the corrected coefficients provided by Westbrook and Dryer in their response to Coffee's comment, which is included at the end of [220].

- Reactions with WGS in the name correspond to reactions with the Water Gas Shift forward/backward reaction:

$$H_2 + CO_2 \Leftrightarrow H_2O + CO \tag{9.1}$$

- Reactions with NO PDF in the name are schemes that include the NO formation model. For details, see NO Model (p. 455). The reaction rates for NO formation account for turbulent fluctuations of temperature.

- It is recommended that you do not use reactions with just NO in the name, as these schemes do not account for temperature fluctuations when computing reaction rates of NO, and are therefore less accurate.

For the WD *n* reactions, single reaction step schemes can still be described as multi-step when the purpose of the additional step is to add an additional passive component (N2) to a single step reaction. The reaction step itself remains single step in these cases.

9.1.2. Reaction Rate Types

On the **Reaction Rate** tab of a reaction definition, the following types of reaction rate may be available:

- Arrhenius (p. 427)
- Arrhenius with Temperature PDF (p. 428)
- Expression (p. 428)
- Equilibrium (p. 428)

9.1.2.1. Arrhenius

Two forms of the Arrhenius dependency are supported in CFX. The first is given by:

$$k = A\, T^{\beta} \exp\left(\frac{-E}{R\,T}\right) \tag{9.2}$$

where k is reaction rate coefficient, A is the pre-exponential factor, E is the activation energy, R is the universal gas constant, β is the temperature exponent and T is the temperature. A modified form:

$$k = A\, T^{\beta} \exp\left(\frac{-T_A}{T}\right) \tag{9.3}$$

where $T_A = E / R$ is the activation temperature.

The pre-exponential factor may be an expression in space, time, absolute or static pressure, turbulence quantities (k, ε), molar mass or mixture composition (species mass fractions, molar fractions, mass concentrations, molar concentrations).

The units of the pre-exponential factor are:

$$time^{-1} \left(mol\ m^{-3}\right)^{(1-n)} \tag{9.4}$$

where n is the sum of the number of reaction orders, (plus 1 for a third body term, if present). As an example, for a reaction with two reactants, both with a reaction order of 1, and a third body term, n = 2+1 = 3. The physical dimensions for the pre-exponential factor in this case are:

$$s^{-1}\ mol^{-2}\ m^{6} \tag{9.5}$$

9.1.2.2. Arrhenius with Temperature PDF

Similar instructions apply here as for Arrhenius, except that a temperature limit list is required to describe the probability density function. For details, see Arrhenius (p. 427). The probability density function is a mathematical model that describes the probability of events occurring over time. This function is integrated to obtain the probability that the event time takes a value in a given time interval.

The temperature limit list requires the entry of the upper and lower bounds for temperature integration range (2 values).

9.1.2.3. Expression

To enter an expression, click the enter expression icon and type the name of the expression into the box. For details, see Expressions in the *CFX-Pre User's Guide*.

9.1.2.4. Equilibrium

For any reversible reaction, $a\left[A\right]\leftrightarrow b\left[B\right]$:

$$K=\left[B\right]^{b}/\left[A\right]^{a} \tag{9.6}$$

where a and b represent the number of moles of materials A and B in the reaction. While quantities [A] and [B] represent the molar concentrations and K, the equilibrium constant (which is dependent on temperature, enthalpy, entropy). The forward and backward reaction rates are defined as:

$$R_{f}=F_{k}\left[A\right]^{a} \tag{9.7}$$

and

$$R_{b}=B_{k}\left[B\right]^{b} \tag{9.8}$$

At equilibrium, $R_{f}=R_{b}$, and therefore:

$$B_{k}\left[B\right]^{b}/\ F_{k}\left[A\right]^{a}=1 \tag{9.9}$$

Where F_{k} is the forward rate and B_{k} is the backward rate. When applied, the Arrhenius Reaction Model calculates the rate according to the following:

$$B_{k}=\frac{F_{k}}{K} \tag{9.10}$$

The equilibrium option is only available if the forward reaction rate is set to Arrhenius.

A reaction is reversible when, for each reactant or production component, the reaction order is equal to the stoichiometric coefficient.

9.2. Using Combustion Models

To access Combustion Models in CFX, you must first set up a reacting fluid. This step is carried out in the **Material** details view. For details, see Material Details View: Reacting Mixture in the *CFX-Pre User's Guide*. There are a number of pre-defined reactions that can be selected, and the **Reaction** details view enables you to create custom reactions for use in CFX-Pre. For details, see Reactions in the *CFX-Pre User's Guide*.

If you want to specify any non-reacting components that will also be present in the domain, you can add them when creating the reacting mixture. Note that the thermodynamic properties for materials from the CFX-Pre database may not be appropriate (constant specific heat capacity) and may produce poor results if a passive component has a significant mass fraction.

The name of each material in the tree view indicates the range of validity for thermodynamical data. As an example `Carbon Monoxide at STP` has constant properties (density, specific heat capacity, and so on) for standard temperature and pressure. For reacting mixtures, materials from the `Gas Phase Combustion` material group should be used. These materials have variable properties and consistent reference enthalpies for heat release. These materials typically have the chemical formulae as their material name (for example, `CO`).

All liquids and solids in the MATERIALS library are defined with constant specific heat capacity. The variable `Cp` is included for all ideal gases except for `Air Ideal Gas`.

You may want to keep a CCL file containing any customized materials and reaction definitions and then import this file into CFX-Pre when needed.

Thermodynamic data for the material properties used in CFX-Pre was obtained from [26]

9.2.1. Which Model is the Most Appropriate?

The following table is meant as a guide in helping you choose which combustion model is most suitable for your simulation. You are also encouraged to read the following sections, which describe how to set up each model to obtain the most accurate results.

Eddy Dissipation Model	Turbulent Fast reaction compared to turbulent time scale (high Damköhler number) Reaction rate dominated by turbulent mixing of reactants or fresh and burned gases (premixed)
Finite Rate Chemistry Model	Laminar or turbulent If turbulent, reaction rate slow compared to turbulent time scale (low Damköhler number) Reaction rate dominated by kinetics (chemistry) Kinetic data for reaction rates required May need special initialization for flame ignition (temperature dependence of reaction rates)

Combined EDM/FRC Model	Turbulent Whole range of Damköhler numbers Kinetic data for reaction rates required May need special initialization for flame ignition (temperature dependence of reaction rates).
Laminar Flamelet Model	Fast chemistry (high Damköhler number) Turbulent Non-premixed "Fuel" and "Oxidizer" well defined, Chemistry library required
Burning Velocity Model, BVM (Partially Remixed), (Turbulent Flame Closure)	Turbulent Premixed or partially-premixed "Fuel" and "Oxidizer" can be mixed (specify inlet mixture fraction) Chemistry library required

The Damköhler number is the ratio of flow-time scale to chemical-time scale:

$$Da = \frac{t_{\text{flow}}}{t_{\text{chem}}} \tag{9.11}$$

9.3. Eddy Dissipation Model

The Eddy Dissipation model is best applied to turbulent flows when the chemical reaction rate is fast relative to the transport processes in the flow. There is no kinetic control of the reaction process. Thus, ignition and processes where chemical kinetics may limit reaction rate may be poorly predicted.

By default, for the Eddy Dissipation model it is sufficient that fuel and oxidant be available in the control volume for combustion to occur. If the product limiter is enabled, by setting the **Eddy Dissipation Coefficient B** parameter to positive, then products must also be available. For details, see Eddy Dissipation Model Coefficient A/B (p. 432). However, combustion products may not always be specified as an input. In this case, products could not form unless they are introduced into the domain. Assuming the problem is one in which a stable flame may be established, initial specification of products within the domain should be sufficient to start and maintain combustion. However, if the combustion is difficult to maintain, it may be necessary to introduce a small fraction of products at an inlet.

The following topics will be discussed:

- Fluid Models (p. 431)
- Initialization (p. 432)

• Solver Parameters (p. 432)

9.3.1. Fluid Models

The models settings described in this section are set when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

9.3.1.1. Chemical Time Scale

The reaction rate is set to zero if the turbulent time scale is smaller than this value. When the model for flame extinction at high turbulence is activated, local extinction occurs when the turbulence time scale is smaller than a chemical time scale (quenching time scale) provided by you. As this is a very simple model for predicting local extinction, the specified chemical time scale may need to be adjusted in order to achieve best results for a specific problem. For methane-air combustion, good starting points are `1.37e-4 [s]` when applying the Kolmogorov time scale, or `5e-4 [s]` when comparing to the mixing time scale. Using the Kolmogorov time scale tends to be more aggressive and may lead to global extinction of the flame, even in situations where this is not physical. It is for this reason that the mixing time scale is recommended. By default, the mixing time scale is applied. This may be changed by setting the expert parameter `use kolmogorov ts for extinction` to `T`.

9.3.1.2. Extinction Temperature

This is a simple model that disables the reaction wherever the temperature is less than the specified extinction temperature.

9.3.1.3. Maximum Flame Temperature

Because of the assumption of complete combustion, the Eddy Dissipation model may over-predict temperature under certain conditions (for example, for hydrocarbon fuels in regions with fuel-rich mixture). When the expected maximum temperature (or an estimated bound for it) is known prior to the simulation, it can be provided to the solver as a constraint. The exact limit is the adiabatic flame temperature for the local mixture. Depending on the application it may be sufficient to use a simple correlation or even a constant limit. CEL expressions may be used in the maximum flame temperature parameter. When the specified maximum flame temperature limit is reached, the reaction rate will be locally stopped, leaving behind fuel and oxidizer. The reaction may continue in other regions of the domain where the fluid temperature is lowered (for example, by mixing with cold fluid or by heat transfer). Note that the maximum flame temperature only bounds temperature increase due to reaction heat release, but it is possible that a different process such as compression or heat transfer can exceed the specified limit.

9.3.1.4. Mixing Rate Limit

The maximum allowed value of the local turbulent mixing rate. Only applicable if the **Eddy Dissipation Model** is used. The **Eddy Dissipation Model** sometimes predicts unphysical behavior, such as the flame creeping across walls. This is because the ratio of the turbulence quantities ε / k becomes large close to wall boundaries, but the turbulence in these regions is low. This value is used to enforce an upper limit on the value ε / k used for computing the reaction rate. For methane/air, a value of approximately 2500[1/s] is reasonable.

9.3.1.5. Eddy Dissipation Model Coefficient A/B

The coefficient A is a factor that appears in both the reactants and the products limiter. For details, see:

- Reactants Limiter in the *CFX-Solver Theory Guide*
- Products Limiter in the *CFX-Solver Theory Guide*.

A must always be positive. There is special treatment in the solver for negative B. If this is the case, the product limiter is not applied and the reaction rate is determined by reactants concentrations and turbulence. When B is negative, the magnitude of B is not significant.

The following default values apply:

`A = 4.0, B = -1`

Variable expressions (CEL) are supported for both parameters.

9.3.1.6. Component Details

This is used to define how the mass fraction (the ratio of the mass of a component to the total mass of the fluid) of the component is to be computed. You must set exactly one component to `Constraint`. The mass fraction of this component will be calculated to be 1 minus the mass fractions of all other components. The constraint could be any of the components, either passive or taking part in the reaction(s). For reasons of accuracy, however, this should be a major component (large mass fraction). For combustion of a fuel in air, the best choice is N2.

For the **Eddy Dissipation Model**, choose `Automatic` or `Transport Equation` for all other components.

9.3.2. Initialization

The initial guess for scalars can have a large effect on the convergence history of a combustion problem. For example, initial specification of a large concentration of fuel and oxidant in the domain can produce high, unrealistic temperatures that slow convergence. It is best to specify the concentrations in such a way that the reaction proceeds starting from the inlet and proceeds through the domain. This will help ensure reasonable flame temperatures during convergence.

In cases where the domain initially contains air, you should specify oxygen with a mass fraction of 0.232 and set each of the other species to `Automatic`. For temperature and other component mass fractions, you can safely choose `Automatic` as the option in most cases.

9.3.3. Solver Parameters

Timestep choice may have a significant effect on the convergence of a flow calculation. Combustion introduces large density gradients in the flow field that can increase the sensitivity of the continuity equation to the timestep. If you are having difficulty converging a combusting flow, particularly during an early stage of convergence, try reducing the timestep to improve solution of the continuity equation. For details, see Eddy Dissipation Model (p. 430).

If the solution of a combusting flow appears to stall with mass not well conserved, use a timestep appropriate to the fluid flow for a few iterations to push fluid through the system.

9.4. Finite Rate Chemistry Model

The Finite Rate Chemistry model is best applied to situations where the chemical time scale is rate-limiting. This model can be used in conjunction with both laminar and turbulent flow.

The following topics will be discussed:

- Fluid Models (p. 433)
- Initialization (p. 434)
- Solver Parameters (p. 434)

9.4.1. Fluid Models

The models settings described in this section are set when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

9.4.1.1. Chemical Time Scale

The reaction rate is set to zero if the turbulent time scale is smaller than this value. When the model for flame extinction at high turbulence is activated, local extinction occurs when the turbulence time scale is smaller than a chemical time scale (quenching time scale) provided by you. Because this is a very simple model for predicting local extinction, the specified chemical time scale may need to be adjusted in order to achieve best results for a specific problem. For methane-air combustion, good starting points are `1.37e-4 [s]` when applying the Kolmogorov time scale, or `5e-4 [s]` when comparing to the mixing time scale. Using the Kolmogorov time scale tends to be more aggressive and may lead to global extinction of the flame, even in situations where this is not physical. It is for this reason that the mixing time scale is recommended. By default the mixing time scale is applied. This may be changed by setting the expert parameter `use kolmogorov ts for extinction` to `T`.

9.4.1.2. Extinction Temperature

This is a simple model that disables the reaction wherever the temperature is less than the specified extinction temperature.

9.4.1.3. Component Details

This is used to define how the mass fraction of the component is to be computed. You must set exactly one component to `Constraint`. The mass fraction of this component will be calculated to be 1 minus the mass fractions of all other components. The constraint could be any of the components, either passive or taking part in the reaction(s). For reasons of accuracy, however, this should be a major component (large mass fraction). For combustion of a fuel in air, the best choice is N2.

For the **Eddy Dissipation Model**, choose `Automatic` or `Transport Equation` for all other components.

9.4.2. Initialization

The Finite Rate Chemistry model initial temperature depends on the temperature dependency of the reactions. If no sensible value is known, restarting from a converged run (with residual values of the order 10^{-4}) that used the EDM model can be useful.

In cases where the domain initially contains air, you should specify oxygen with a mass fraction of 0.232 and set each of the other species to `Automatic`.

For **Temperature**, you should specify a value that is as close to the correct value as you can determine. This value should typically be above 900 K, and 1200 K is recommended as a starting point. If the mixture fails to ignite, running to a converged solution first with the Eddy Dissipation model and then restarting with the Finite Rate Chemistry model may fix the problem.

9.4.3. Solver Parameters

When using the Finite Rate Chemistry model, using a timestep that is too large at the beginning may cause the solver to fail due to reaction rates not converging. Under these circumstances, the timestep must be reduced.

It may help to increase time scales for energy and mass fraction equation classes after running the simulation for a while. You should always use same time scale for energy and mass fraction equation classes.

Overshooting temperatures during the first few iterations may cause convergence difficulties. Try running first few iterations with the Eddy Dissipation Model, then restart with Finite Rate Chemistry.

Enabling temperature damping may also help, but may slow down convergence (this is set using the Expert Parameter `temperature damping=t`).

9.5. Combined EDM/Finite Rate Chemistry Model

For the combined Finite Rate Chemistry/Eddy Dissipation Model, the reaction rates are first computed for each model separately and then the minimum of the two is used. This procedure is applied for each reaction step separately, so while the rate for one step may be limited by the chemical kinetics, some other step might be limited by turbulent mixing at the same time and physical location.

It is also possible to apply different combustion models to each of the steps in a multi-step scheme. Some of the predefined schemes make use of this feature, regardless of the global model selection.

The combined model is valid for a wide range of configurations, provided the flow is turbulent. In particular, the model is valid for many reactions that range from low to high Damköhler numbers (chemistry slow/fast compared to turbulent time scale). Use of this model is recommended if reaction rates are limited by turbulent mixing in one area of the domain and limited by kinetics somewhere else. The Eddy Dissipation model can, however, be more robust than Finite Rate Chemistry or the combined model.

The following topics will discussed:

- Fluid Models (p. 435)
- Initialization (p. 435)
- Solver Parameters (p. 436)

9.5.1. Fluid Models

The models settings described in this section are set when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

9.5.1.1. Chemical Time Scale

The reaction rate is set to zero if the turbulent time scale is smaller than this value. When the model for flame extinction at high turbulence is activated, local extinction occurs when the turbulence time scale is smaller than a chemical time scale (quenching time scale) provided by you. As this is a very simple model for predicting local extinction, the specified chemical time scale may need to be adjusted in order to achieve best results for a specific problem. For methane-air combustion, good starting points are `1.37e-4 [s]` when applying the Kolmogorov time scale, or `5e-4 [s]` when comparing to the mixing time scale. Using the Kolmogorov time scale tends to be more aggressive and may lead to global extinction of the flame, even in situations where this is not physical. It is for this reason that the mixing time scale is recommended. By default the mixing time scale is applied. This may be changed by setting the expert parameter `use kolmogorov ts for extinction` to `T`.

9.5.1.2. Extinction Temperature

This is a simple model that disables the reaction wherever the temperature is less than the specified extinction temperature.

9.5.1.3. Component Details

This is used to define how the mass fraction of the component is to be computed. You must set exactly one component to `Constraint`. The mass fraction of this component will be calculated to be 1 minus the mass fractions of all other components. The constraint could be any of the components, either passive or taking part in the reaction(s). For reasons of accuracy, however, this should be a major component (large mass fraction). For combustion of a fuel in air, the best choice is N2.

For the Eddy Dissipation Model, choose Automatic of Transport Equation for all other components.

9.5.2. Initialization

The Finite Rate Chemistry model initial temperature depends on the temperature dependency of the reactions. If no sensible value is known, restarting from a run that used the EDM model can be useful.

As the Finite Rate Chemistry model is more sensitive to initial guess values, you should follow the advice given in the initialization section for this model. For details, see Initialization (p. 434).

9.5.3. Solver Parameters

The advice given for the Finite Rate Chemistry model is also applicable to the combined model. For details, see Solver Parameters (p. 434).

9.6. Reaction-Step Specific Combustion Model Control

For multi-step reactions, there is the option of specifying a different combustion model for each reaction step. The choice of combustion model for each step is optional, and is made when the **Reaction or Combustion Model** check box is selected for a single-step reaction. Specification is carried out in the **Reaction** details view in CFX-Pre. For details, see Single Step: Basic Settings in the *CFX-Pre User's Guide*.

The available choices for a reaction-step specific reaction are Eddy Dissipation Model, Finite Rate Chemistry Model, and EDM/Finite Rate Chemistry Model. For details, see:

- Eddy Dissipation Model (p. 430)
- Finite Rate Chemistry Model (p. 433)
- Combined EDM/Finite Rate Chemistry Model (p. 434).

When a reaction contains a reaction-step specific combustion model, the model overrides the choice of combustion model set on the **Fluid Domains** form in CFX-Pre, except if the option on the **Fluid Domains** form is set to None.

Because of the behavior described in the paragraph above, it is important that you select a combustion model on the **Fluid Domains** form, even if you have selected models for each and every reaction step in a multi-step reaction.

9.7. Laminar Flamelet with PDF Model

The laminar flamelet model solves only two transport equations for a large number of species (low computational cost). It provides information on minor species and radicals (such as CO and OH). As well as accounting for turbulent fluctuations in composition (presumed PDF), it models local extinction at high scalar dissipation rates or shear strain.

The model is only applicable for two-feed systems (fuel and oxidizer), and requires a chemistry library as input. The same pressure level must apply to the whole domain. The model is only for non-premixed systems.

The following topics will be discussed:

- Fluid Models (p. 437)
- Boundary Settings (p. 437)
- Initialization (p. 438)

- Solver Parameters (p. 438)

Note:

If you are using the NO model with the Flamelet model, you will need to set up your simulation differently. For details, see NO Model with Flamelet Model (p. 456).

9.7.1. Fluid Models

The models settings described in this section are set when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

9.7.1.1. Component Details

Because transport equations are not solved for species involved in the flamelet library reactions, you should select either `Library` or `Automatic` for all such components.

Always set N2 to be a constraint. Components other than N2 and components covered by the flamelet library must be set to Automatic or Library.

Components not covered by the flamelet library should be set to `Transport Equation`. The transport equation components must be minor components, as they are either postprocessed or their added mass fractions will be subtracted from the mass fraction of the constraint material. For details, see NO Model (p. 455).

9.7.2. Boundary Settings

When specifying an inlet or opening boundary condition, you will set a **Mixture** option based on the properties of the incoming fluid. The options available are:

9.7.2.1. Fuel

This option is used when the mass fraction of the fuel component is 1.

9.7.2.2. Oxidizer

This is used when the mass fraction of the oxidizer (usually air) is 1.

9.7.2.3. Mixture Fraction

Use this parameter when you need to specify the reacted mixture of fuel and oxidizer (for example, in a reacted pilot flame). A value of zero corresponds to 100% oxidizer, and a value of 1 corresponds to 100% fuel.

9.7.2.4. Mixture Fraction Mean and Variance

Used in conjunction with the **Mixture Fraction** parameter, but additionally you can specify the variance for the "unmixedness" of fuel and oxidizer. By default, the three options above imply zero variance.

9.7.3. Initialization

When using the Flamelet model, no initialization data for the library species is required because no transport equations are solved for them. You can set values for the mixture fraction mean and variance, but `Automatic` is fine if you do not know of sensible initial values (a value of zero corresponds to 100% oxidizer, and a value of 1 corresponds to 100% fuel).

If you are modeling the formation of NO, you will need to set an initialization option for the NO component. For details, see NO Model (p. 455).

9.7.4. Solver Parameters

The advice given for timestep selection in the EDM model is applicable here. For details, see Solver Parameters (p. 432).

9.8. Burning Velocity Model (Premixed or Partially Premixed)

This model is a combined model using:

- Model for the progress of the global reaction: Burning Velocity Model (BVM), also called Turbulent Flame Closure (TFC)
- Model for the composition of the reacted and non-reacted fractions of the fluid: Laminar Flamelet with PDF

The following topics will be discussed:

- Fluid Models (p. 438)
- Turbulent Burning Velocity (p. 439)
- Spark Ignition Model (p. 439)
- Boundary Settings (p. 440)
- Initialization (p. 441)
- Solver Parameters (p. 441)
- Other Parameters (p. 441)

Information on the theory behind this model is available in Burning Velocity Model (Premixed or Partially Premixed) in the *CFX-Solver Theory Guide*.

9.8.1. Fluid Models

The models settings described in this section are set when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

9.8.1.1. Component Details

Because transport equations are not solved for species involved in the flamelet library reactions, you should select either **Library** or **Automatic** for all such components.

Always set N2 to be a constraint. Components other than N2 and components covered by the flamelet library must be set to `Automatic` or `Library`.

Components not covered by the flamelet library should be set to Transport Equation. The transport equation components must be minor components, as either they are postprocessed or their added mass fractions will be subtracted from the mass fraction of the constraint material. For details, see NO Model (p. 455).

9.8.2. Turbulent Burning Velocity

Select the model for the turbulent burning velocity. For many gas turbine applications, the Zimont model, which is the default, is a good choice.

9.8.2.1. Value

Specify the value for the turbulent burning velocity. Typically, this will be an expression in terms of the laminar burning velocity.

9.8.2.2. Zimont Correlation

Set the critical velocity gradient parameter to control the flame stretch and flame extinction under high turbulence. The default value of g_{cr}=10000 [1/s] is appropriate for methane/air combustion at standard conditions. At higher pressures, better results may be obtained by increasing the value. For details, see Adjusting Model Coefficients in the *CFX-Solver Theory Guide*.

Variable expressions (CEL) may be specified for **Turbulent Burning Velocity Factor** A, **Critical Velocity Gradient** g_{cr} and **Variance Coefficient** μ_{str}.

9.8.2.3. Peters Correlation

For details on the model coefficients, see Adjusting Model Coefficients in the *CFX-Solver Theory Guide*.

9.8.2.4. Mueller Correlation

Set **Critical Scalar Dissipation** to control the amount of quenching occurring under rapid mixing. The default value is χ_{q}=1.0 [1/s]. For details on the model coefficients, see Adjusting Model Coefficients in the *CFX-Solver Theory Guide*.

A variable expression (CEL) may be specified for **Critical Scalar Dissipation**.

9.8.3. Spark Ignition Model

The spark ignition model is available only for transient simulations.

9.8.3.1. Spark Kernel

Set **Kernel Center** and **Initial Volume** to the location and the initial size of the spark. Set **Transition Radius** to control when the transition from the spark model to the regular combustion model to occur. The transition radius should be sufficiently large such that the transition region can be resolved by the mesh.

9.8.3.2. Ignition Time

Specify the time for spark ignition to occur.

9.8.3.3. Spark Energy

Optionally, the electrical energy of the spark may be included into the model. The energy of the spark can be defined by one of the following:

- Spark energy (specifies the total energy of the spark)
- Spark power (energy flux) and duration of the spark

The **Electrode Efficiency** parameter controls the fraction of the spark energy introduced into the fluid. The complementary fraction is modeled to be lost to the external world by heat transfer in the spark electrode.

9.8.4. Boundary Settings

When specifying an inlet or opening boundary condition, you will set a mixture option based on the properties of the incoming fluid. The options available are:

9.8.4.1. Fuel

This option is used when the mass fraction of the fuel component is 1.

9.8.4.2. Oxidizer

This is used when the mass fraction of the oxidizer (usually air) is 1.

9.8.4.3. Mixture Fraction

Use this parameter when you need to specify the reacted mixture of fuel and oxidizer (for example, in a reacted pilot flame). A value of zero corresponds to 100% oxidizer, and a value of 1 corresponds to 100% fuel.

9.8.4.4. Mixture Fraction Mean and Variance

Used in conjunction with the Mixture Fraction parameter, but additionally you can specify the variance for the "unmixedness" of fuel and oxidizer. By default, the three options above imply zero variance.

9.8.4.5. Reaction Progress

This parameter is used to specify the reaction progress variable c. c=0 corresponds to fresh (unburned) gases, whereas c=1 corresponds to completely burned gases. You can select Fresh Gases (c=0), Burned Gases (c=1), or enter a value or expression for the value of c.

9.8.5. Initialization

When using the Burning Velocity Model (BVM), no initialization data for the library species is required since no transport equations are solved for them. You can set values for the mixture fraction mean and variance, but Automatic is fine if you do not know of sensible initial values (a value of zero corresponds to 100% oxidizer, and a value of 1 corresponds to 100% fuel).

If you are modeling the formation of NO, you will need to set an initialization option for the NO component. For details, see NO Model (p. 455).

9.8.5.1. Reaction Progress

This parameter is used to specify the reaction progress variable c within the domain. c=0 corresponds to fresh (unburned) gases, whereas c=1 corresponds to completely burned gases. You can select Fresh Gases (c=0), Burned Gases (c=1), or enter a value or expression for the value of c.

9.8.6. Solver Parameters

The advice given for timestep selection in the EDM model is applicable here. For details, see Solver Parameters (p. 432).

9.8.7. Other Parameters

You can turn off the calculation of the reaction progress variable c (discussed in Initialization (p. 441)) by setting the Expert Parameter `solve reaction progress` to `f`.

9.9. Extended Coherent Flame Model (ECFM)

The Extended Coherent Flame Model (ECFM) is a combined model employing:

- A model for the progress of the global reaction: Extended Coherent Flame Model (ECFM), which is a member of the class of Flame Surface Density Models
- A model for the composition of the reacted and non-reacted fractions of the fluid: Laminar Flamelet with PDF. For partially premixed configurations this model does also implement post-flame fuel oxidation (diffusion flame tail)

The coupling of ECFM with the flamelet model for effects of non-premixedness extends the range of applicability of the model similar to the three-zones extension of the model found in the literature (ECFM-3Z).

The following topics will be discussed:

- Fluid Models (p. 442)

- Spark Ignition Model (p. 442)
- Boundary Settings (p. 443)
- Initialization (p. 444)
- Solver Parameters (p. 444)
- Other Parameters (p. 445)

Information on the theory behind the Extended Coherent Flame Model is available in Extended Coherent Flame Model (ECFM) in the *CFX-Solver Theory Guide*.

9.9.1. Fluid Models

The model settings described in this section are set when creating a domain in Ansys CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*

9.9.1.1. Laminar Flame Thickness

The laminar flame thickness is an important input quantity to the Extended Coherent Flame Model. The following options are available:

- Blint: Applies Blint's correlation for laminar flame thickness. See Laminar Flame Thickness in the *CFX-Solver Theory Guide* for the definition of the correlation.
- Value: Specify a CEL expression for the laminar flame thickness.

9.9.1.2. Component Details

It is recommended that you choose N_2 to be the constraint, which is the default. The remaining components in the flamelet library must be set to `Automatic` or `Library`. Components not covered by the flamelet library should be set to `Transport Equation`. The transport equation components should be minor species, as they are either postprocessed or their added mass fractions will be subtracted from the mass fraction of the constraint material. For details, see NO Model (p. 455).

9.9.2. Spark Ignition Model

The spark ignition model is available only for transient simulations.

9.9.2.1. Spark Kernel

Set **Kernel Center** and **Initial Volume** to the location and the initial size of the spark. Set **Transition Radius** to control when the transition from the spark model to the regular combustion model to occur. The transition radius should be sufficiently large such that the transition region can be resolved by the mesh.

9.9.2.2. Ignition Time

Specify the time for spark ignition to occur.

9.9.2.3. Spark Energy

Optionally, the electrical energy of the spark may be included into the model. The energy of the spark can be defined by one of the following:

- Spark energy (specifies the total energy of the spark)
- Spark power (energy flux) and duration of the spark

The **Electrode Efficiency** parameter controls the fraction of the spark energy introduced into the fluid. The complementary fraction is modeled to be lost to the external world by heat transfer in the spark electrode.

9.9.3. Boundary Settings

When specifying an inlet or opening boundary condition, you will set options for mixture, for reaction progress and for flame surface density. When specifying a wall boundary condition an option will be set only for flame surface density.

9.9.3.1. Mixture

The options available are:

- **Fuel**: This option is used when the mass fraction of the fuel component is 1.
- **Oxidizer**: This option is used when the mass fraction of the oxidizer (usually air) is 1.
- **Mixture Fraction**: Use this parameter when you need to specify the mixture of fuel and oxidizer or the equivalent reacted mixture (for example, in a reacted pilot flame). A value of zero corresponds to 100% oxidizer, and a value of 1 corresponds to 100% fuel.
- **Mixture Fraction Mean and Variance**: Used in conjunction with the **Mixture Fraction** parameter, but additionally you can specify the variance for the "unmixedness" of fuel and oxidizer. By default, the three options above imply zero variance.
- Equivalence Ratio: Use this parameter when you need to specify the mixture of fuel and oxidizer, or the equivalent reacted mixture. The difference to the **Mixture Fraction** option is that the value is given for the equivalence ratio: 1 = stoichiometric, <1 for lean mixtures and >1 for fuel rich mixtures.

9.9.3.2. Reaction Progress

This parameter is used to specify the reaction progress variable c. `c=0` corresponds to fresh (unburned) gases, whereas `c=1` corresponds to completely burned gases. You can select `Fresh Gases (c=0)`, `Burned Gases (c=1)`, or enter a value or expression for the value of c.

9.9.3.3. Flame Surface Density

For inlet or opening boundary conditions a value will be specified for the flame surface density, which is the area of flame surface per volume. A value of `0 [m^-1]` corresponds to no flame existing on the boundary and is appropriate for most inlets or openings.

At wall boundaries an option will be specified controlling how the flame surface interacts with the wall. The following options are available:

- **Wall Quenching**:Enables the wall quenching model
- **Zero Flux**: Disables the wall quenching model
- **Wall Flux in**: Use this parameter in order to implement custom models for flame-wall interaction. Negative values for the **Flame Surface Density Flux** parameter correspond to destruction of the flame at the wall (quenching).

9.9.4. Initialization

When using the Extended Coherent Flame Model (ECFM), no initialization data for the library species is required because no transport equations are solved for them. If you are modeling the formation of NO, you will need to set an initialization option for the NO component. For details, see NO Model (p. 455).

9.9.4.1. Mixture

You can set values for the mixture fraction mean and variance, but **Automatic** is fine if you do not know of sensible initial values (a value of zero corresponds to 100% oxidizer, and a value of 1 corresponds to 100% fuel).

9.9.4.2. Reaction Progress

This parameter is used to specify the reaction progress variable within the domain. $c=0$ corresponds to fresh (unburned) gases, whereas $c=1$ corresponds to completely burned gases.

9.9.4.3. Flame Surface Density

This parameter is used to specify the flame surface density Σ within the domain. Σ=0 [m^-1] corresponds to no reaction present. Unless the reaction is ignited by other means, for example, the Spark Ignition Model (p. 442), it may be necessary to specify non-zero values at least in part of the region in order to start the combustion. When the **Automatic** option is set the default initial values are derived from the gradient of reaction progress,

$$\Sigma = \frac{s_T^{Zimont}}{s_L} \cdot \left| \nabla \tilde{c} \right|$$

where s_L denotes the laminar burning velocity and s_T^{Zimont} the turbulent burning velocity according to the Zimont correlation.

9.9.5. Solver Parameters

The advice given for timestep selection in the EDM model is applicable here. For details, see Solver Parameters (p. 432).

9.9.6. Other Parameters

You can turn off the calculation of the reaction progress variable c by setting the Expert Parameter `solve reaction progress` to `f`. You can turn off the calculation of the flame surface density variable Σ by setting the Expert Parameter `solve fsd` to `f`.

9.10. Residual Material Model

The following topics will be discussed:

- Fluid Models (p. 445)
- Laminar Burning Velocity (p. 446)
- Boundary Settings (p. 446)
- Initialization (p. 446)
- Initialization (p. 444)
- Other Parameters (p. 446)

9.10.1. Fluid Models

The single-mixture fraction approach describes mixing as a two-stream system consisting of fuel and oxidizer. The residual material model extends the mixing model to a three-stream system consisting of fuel, oxidizer and residual material (inert, ballast). The model is available in combination with the partially-premixed combustion models 'BVM (Partially Premixed)' and 'Extended Coherent Flame Model'.

The residual material model accounts for the following effects caused by the presence of ballast material in the mixture:

- Additional degree of freedom for mixture composition (1 extra transport equation)
- Lower peak temperatures because the fuel/air mixture is diluted
- Reduced velocity for flame propagation.

For the **Exhaust Gas** option the residual material is modeled to consist of products originating from the same fuel and oxidizer as for the combustion. This assumption enables calculating the whole mixture composition from a single-mixture fraction flamelet library, that is, the same library that is also used without a residual material model. The details of the component mass fraction calculation are described in the solver theory documentation, see Mixture Composition in the *CFX-Solver Theory Guide*.

9.10.1.1. Reinitialization

For transient simulations the residual material can be reinitialized at specified times. Under reinitialization all products currently present in the mixture are marked for being treated as residual material from now on. The **Time List** may be specified for applying reinitialization once or several times.

A typical example for use of reinitializing the residual material is the simulation of subsequent cycles of an internal combustion engine. By incomplete load exchange a part of the products generated in one cycle will remain in the combustion chamber, hence serving as residual material for the following engine cycle. This can be modeled by specifying the time for reinitialization at some point after the end of combustion in the first cycle and before the fresh mixture enters the chamber for the next cycle.

9.10.2. Laminar Burning Velocity

When the residual material model is selected, it is recommended that you also specify the dependency of the laminar burning velocity. For details, see Laminar Burning Velocity (p. 452).

9.10.3. Boundary Settings

For the residual material model the following additional options are available for **Mixture** boundary conditions:

- **Mixture Fraction and Fuel Tracer**: Use this parameter for direct specification of boundary values for the mean mixture fraction and fuel tracer.
- **Mixture Fraction and Residual Material**: Specification of the residual material mass fraction and the conditional mixture fractions for the fresh and residual fractions.
- **Equivalence Ratio and Residual Material**: Specification of the residual material mass fraction and the conditional equivalence ratios for the fresh and residual fractions.

For options other than the above, the fuel tracer is set equal to mixture fraction, which corresponds to no residual material present in the inlet stream.

9.10.4. Initialization

For the residual material model initial values can be specified for the fuel tracer. The default or **Automatic** option initializes the fuel tracer to be equal to mixture fraction, which corresponds to zero residual material.

9.10.5. Other Parameters

The transport equation for fuel tracer is solved in the mixture fraction equation group. You can turn off the calculation of the group by setting the Expert Parameter `solve mixture fraction` to `f`.

9.11. Flamelet Libraries

The mixture fraction based combustion models in Ansys CFX (PDF Flamelet, Burning Velocity Model and Extended Coherent Flame Model) require a flamelet library that defines the burnt mixture composition as a function of the combustion scalars solved.

The following topics will be discussed:

- Loading Flamelet Libraries (p. 447)

- Flamelet Library (FLL) File Format (p. 447)
- Stoichiometric Mixture Fraction (p. 451)
- Laminar Burning Velocity (p. 452)

It is possible to use flamelet libraries, provided that the appropriate file format is used. For details, see Flamelet Library (FLL) File Format (p. 447).

9.11.1. Loading Flamelet Libraries

To use a particular flamelet library in CFX, simply load the corresponding `<lib>.ccl` file into CFX-Pre. You can do so by selecting:

1. **File** > **Import** > **CCL ...** from the menu bar
2. *Import Library Data/File to Import*  in the **Materials** workspace, or
3. *Import Library Data/File to Import*  in the **Reactions** workspace.

Each of these methods leads to the same **Import CCL** dialog box. Loading the `<lib>.ccl` file defines the material properties for all required species and creates a reaction object for the flamelet library. When importing from the **Materials** workspace or the **Reactions** workspace, you should select all materials and reactions in the file for import. Edit the reaction to define the laminar burning velocity (laminar flame speed) and the stoichiometric mixture fraction.

The flamelet library reaction object contains the path name of the flamelet library file (`.fll` file) in the **Library File** setting. If the path name is specified by a relative path, it will be interpreted as being relative to the working directory. If the `.fll` file cannot be found there, the relative path will be interpreted as being relative to *<CFXROOT>*`/etc/reactions-extra`. The **Reaction** details view can be used to change the **Library File** setting of the flamelet library reaction object as required.

9.11.2. Flamelet Library (FLL) File Format

9.11.2.1. Flamelet Library (FLL) File Contents

The Flamelet Library (FLL) file contains a flamelet library that can be used with the PDF Flamelet model or with partially premixed combustion models such as Burning Velocity Model (BVM) and the Extended Coherent Flame Model (ECFM).

The flamelet library file defines the component mass fractions in the mixture in tabulated form. For the unburnt solution (inert mixing), the mass fractions are a function of the mean mixture fraction (Favre-average):

$$\tilde{Y}_{\alpha}^{\,u} = \tilde{Y}_{\alpha}^{\,u}\left(\tilde{Z}\right) \tag{9.12}$$

For the burnt mixture, the mass fractions depend on the mean mixture fraction, the variance of mixture fraction, and scalar dissipation rate:

$$\tilde{Y}_{\alpha}^{\,b} = \tilde{Y}_{\alpha}^{\,flamelet}\left(\tilde{Z}, \overline{Z''^2}, \chi\right) \tag{9.13}$$

In order to improve the efficiency of the tabulation, the burnt solution is not tabulated directly as a function of the variance. Instead, a normalized variance is used as the input variable:

$$\tilde{Y}_{\alpha}^{\ b} = \tilde{Y}_{\alpha}^{\ flamelet}\left(\tilde{Z}, z\,\mathrm{var}\,norm, \chi\right) \tag{9.14}$$

The variance is normalized according to the equation:

$$z\,\mathrm{var}\,norm = \frac{\sqrt{\widetilde{Z''^2}}}{\min\left(\tilde{Z}, \left(1-\tilde{Z}\right)\right)} \tag{9.15}$$

The mass fractions are stored in the FLL file as multidimensional tables over the input variables.

9.11.2.2. Flamelet Library (FLL) File Format

The FLL file is organized into the following sections:

1. Comments at the beginning of the file (for example, descriptions of library parameters (optional)).
2. A header defining the library dimensions and the sample points for tabulation.
3. Component mass fractions for the unburnt solution (that is, inert mixing).
4. Component mass fractions for the burnt flamelet solutions.

9.11.2.2.1. Detailed FLL File Format

Each line in the FLL file may be up to 80 characters long.

9.11.2.2.1.1. Comment Section

The comment section at the beginning of the file may consist of any number of lines, whose contents is ignored. The end of the comments section is identified by the keyword `DIMENSIONS` at the beginning of the line.

9.11.2.2.1.2. Header

The relevant file contents start below a line beginning with the `DIMENSIONS` keyword. First the library dimensions are defined, followed by the table points for the input quantities. The values given under `STRAIN` are currently not used by the CFX-Solver, so the actual values do not matter (provided that the correct count of values given).

```
DIMENSIONS :
MSTZ, MSPEC, MSTRN, MSVAR (I3,I3,I3,I3)
nmean nmat nchi nvar   integer
-----------------------------------------
STRAIN - STEPS
write (iunit,*) (strain(i),i=1,nchi)          | all following real
SCALAR DISSIPATION RATE - STEPS               |
write (iunit,*) (chi(i),i=1,nchi)             |
VARIANCE - STEPS                              |
write (iunit,*) (zvarnorm(i),i=1,nvar)        |
Z                                             |
write (iunit,*) (zmean(i),i=1,nmean)          |
```

9.11.2.2.1.3. Unburnt Flamelet

For each component the material name is given on a line by its own. Each material name is followed by several lines with the mass fraction values corresponding to the mean mixture fraction tabulation points.

```
Unburnt Flamelet
do j=1,nmat
  cmat(j)   character*(*)
  write (iunit,*) (mfinert(i,j),i=1,nmean)     | all following real
```

9.11.2.2.1.4. Burnt Flamelet Solution

For each sample point of the normalized variance and the scalar dissipation rate, the component mass fractions are specified as 1D tables over mixture fraction. The order of the components must be the same as for the unburnt flamelet.

```
do l=1,nvar
do k=1,nchi
  chi(k) zvarnorm(l)   real
  do j=1,nmat
    cmat(j)   character*(*)
    write (iunit,*) (mfflm(i,j,k,l),i=1,nmean) | all following real
```

9.11.2.2.2. Example FLL File

```
C     Flamelet Library
C
C     Pressure         [bar]:       1.0000
C     Temp Oxidizer    [K]  :     300.0000
C     Temp Fuel        [K]  :     300.0000
C     Scalar Diss Rate [1/s]:   0 - 10
C     Z-Variance factor :       0 - 0.99
C
C     Composition oxidizer side:
C      N2         :   0.767000
C      O2         :   0.233000
C
C     Composition fuel side    :
C      H2         :   1.000000
C
-------------------------------------------
DIMENSIONS :
MSTZ, MSPEC, MSTRN, MSVAR (I3,I3,I3,I3)
 10  9  2  2
-------------------------------------------
STRAIN – STEPS  (unused)
 0.00000E+00   0.00000E+00
SCALAR DISSIPATION RATE - STEPS
 0.00000E+00   0.10000E+02
VARIANCE - STEPS
 0.00000E+00   0.99000E+00
Z
 0.00000E+00   0.25000E-01   0.50000E-01   0.75000E-01   0.10000E+00
 0.48571E+00   0.61429E+00   0.74286E+00   0.87143E+00   0.10000E+01
Unburnt Flamelet
H2
 0.00000E+00   0.25000E-01   0.50000E-01   0.75000E-01   0.10000E+00
 0.48571E+00   0.61429E+00   0.74286E+00   0.87143E+00   0.10000E+01
O2
 0.23300E+00   0.22718E+00   0.22135E+00   0.21553E+00   0.20970E+00
 0.11983E+00   0.89871E-01   0.59914E-01   0.29957E-01   0.10408E-15
H2O
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
```

```
OH
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
H
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
O
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
HO2
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
H2O2
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00
N2
 0.76700E+00   0.74782E+00   0.72865E+00   0.70947E+00   0.69030E+00
 0.39446E+00   0.29584E+00   0.19723E+00   0.98613E-01   0.00000E+00
   0.000000E+00   0.000000E+00
H2
 0.10000E-24   0.32167E-03   0.22257E-01   0.48043E-01   0.73785E-01
 0.47074E+00   0.60305E+00   0.73537E+00   0.86768E+00   0.10000E+01
O2
 0.23300E+00   0.26938E-01   0.13402E-05   0.30433E-08   0.12735E-10
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.10000E-24
H2O
 0.10000E-24   0.21966E+00   0.24896E+00   0.24267E+00   0.23612E+00
 0.13493E+00   0.10120E+00   0.67463E-01   0.33732E-01   0.11623E-15
OH
 0.10000E-24   0.50470E-02   0.26098E-03   0.13152E-04   0.75212E-06
 0.38811E-24   0.77449E-29   0.00000E+00   0.00000E+00   0.10000E-24
H
 0.10000E-24   0.22525E-04   0.89601E-04   0.17580E-04   0.28243E-05
 0.12612E-18   0.10053E-21   0.60974E-22   0.16296E-21   0.10000E-24
O
 0.10000E-24   0.41505E-03   0.12661E-05   0.61760E-08   0.39353E-10
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.10000E-24
HO2
 0.10000E-24   0.30968E-05   0.11313E-08   0.31681E-11   0.13934E-13
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.10000E-24
H2O2
 0.10000E-24   0.25535E-06   0.12105E-08   0.18292E-10   0.38118E-12
 0.00000E+00   0.00000E+00   0.00000E+00   0.00000E+00   0.10000E-24
N2
 0.76700E+00   0.74759E+00   0.72843E+00   0.70926E+00   0.69009E+00
 0.39433E+00   0.29575E+00   0.19717E+00   0.98588E-01   0.00000E+00
   0.100000E+02   0.000000E+00
H2
 0.10000E-24   0.25063E-02   0.20492E-01   0.46255E-01   0.72274E-01
 0.47042E+00   0.60285E+00   0.73524E+00   0.86763E+00   0.10000E+01
O2
 0.23300E+00   0.45440E-01   0.12583E-02   0.85114E-04   0.20097E-04
 0.55737E-06   0.28727E-06   0.14271E-06   0.57107E-07   0.00000E+00
H2O
 0.10000E-24   0.18711E+00   0.24295E+00   0.24161E+00   0.23582E+00
 0.13493E+00   0.10119E+00   0.67463E-01   0.33732E-01   0.21684E-18
OH
 0.10000E-24   0.10782E-01   0.40147E-02   0.87590E-03   0.25608E-03
 0.18801E-08   0.11176E-08   0.72679E-09   0.28904E-09   0.10017E-24
H
 0.10000E-24   0.91966E-03   0.20853E-02   0.16570E-02   0.13228E-02
 0.19559E-03   0.11600E-03   0.64313E-04   0.27968E-04   0.00000E+00
O
 0.10000E-24   0.53884E-02   0.55758E-03   0.43811E-04   0.69272E-05
 0.25113E-08   0.31153E-08   0.27801E-08   0.16103E-08   0.10340E-24
HO2
 0.10000E-24   0.17485E-04   0.63262E-06   0.50986E-07   0.14220E-07
 0.28498E-08   0.23780E-08   0.18064E-08   0.10560E-08   0.10340E-24
H2O2
 0.10000E-24   0.16932E-04   0.15026E-05   0.27677E-06   0.10599E-06
 0.10852E-09   0.67381E-10   0.42146E-10   0.20608E-10   0.10017E-24
```

```
N2
 0.76700E+00   0.74782E+00   0.72864E+00   0.70947E+00   0.69030E+00
 0.39445E+00   0.29584E+00   0.19723E+00   0.98610E-01   0.00000E+00
   0.000000E+00   0.990000E+00
H2
 0.10000E-24   0.83488E-02   0.29476E-01   0.53419E-01   0.78272E-01
 0.48332E+00   0.60536E+00   0.73540E+00   0.86768E+00   0.10000E+01
O2
 0.23300E+00   0.93664E-01   0.56982E-01   0.42752E-01   0.35775E-01
 0.10067E+00   0.18453E-01   0.24171E-03   0.00000E+00   0.10000E-24
H2O
 0.10000E-24   0.14918E+00   0.18411E+00   0.19377E+00   0.19520E+00
 0.21515E-01   0.80278E-01   0.67179E-01   0.33732E-01   0.11623E-15
OH
 0.10000E-24   0.10223E-02   0.86039E-03   0.68068E-03   0.55922E-03
 0.44316E-04   0.12687E-03   0.92239E-05   0.32532E-09   0.10000E-24
H
 0.10000E-24   0.30313E-04   0.33528E-04   0.28895E-04   0.27175E-04
 0.20180E-05   0.55100E-05   0.73005E-06   0.86118E-10   0.10000E-24
O
 0.10000E-24   0.38721E-04   0.31603E-04   0.24800E-04   0.20325E-04
 0.19873E-05   0.44947E-05   0.35796E-06   0.56847E-11   0.10000E-24
HO2
 0.10000E-24   0.51104E-06   0.37260E-06   0.28331E-06   0.23001E-06
 0.26906E-07   0.54387E-07   0.32524E-08   0.00000E+00   0.10000E-24
H2O2
 0.10000E-24   0.47490E-07   0.34747E-07   0.26623E-07   0.21759E-07
 0.32576E-08   0.55969E-08   0.30967E-09   0.00000E+00   0.10000E-24
N2
 0.76700E+00   0.74772E+00   0.72851E+00   0.70932E+00   0.69015E+00
 0.39445E+00   0.29577E+00   0.19717E+00   0.98588E-01   0.00000E+00
   0.100000E+02   0.990000E+00
H2
 0.10000E-24   0.85312E-02   0.29186E-01   0.52917E-01   0.77691E-01
 0.48328E+00   0.60515E+00   0.73521E+00   0.86762E+00   0.10000E+01
O2
 0.23300E+00   0.98319E-01   0.60926E-01   0.45884E-01   0.38370E-01
 0.10092E+00   0.19024E-01   0.28950E-03   0.82501E-07   0.00000E+00
H2O
 0.10000E-24   0.13837E+00   0.17493E+00   0.18644E+00   0.18914E+00
 0.20926E-01   0.78935E-01   0.67061E-01   0.33732E-01   0.21684E-18
OH
 0.10000E-24   0.42102E-02   0.37067E-02   0.29900E-02   0.24882E-02
 0.23040E-03   0.54292E-03   0.54105E-04   0.84910E-08   0.10017E-24
H
 0.10000E-24   0.73407E-03   0.10170E-02   0.10398E-02   0.99514E-03
 0.78870E-04   0.27180E-03   0.13554E-03   0.34183E-04   0.00000E+00
O
 0.10000E-24   0.20086E-02   0.15792E-02   0.12333E-02   0.10115E-02
 0.10873E-03   0.23342E-03   0.17181E-04   0.16711E-08   0.10340E-24
HO2
 0.10000E-24   0.27321E-04   0.17375E-04   0.12985E-04   0.10659E-04
 0.25423E-05   0.34829E-05   0.10183E-06   0.93590E-09   0.10340E-24
H2O2
 0.10000E-24   0.25316E-04   0.16404E-04   0.12302E-04   0.10084E-04
 0.20839E-05   0.30995E-05   0.10509E-06   0.28831E-10   0.10017E-24
N2
 0.76700E+00   0.74777E+00   0.72862E+00   0.70947E+00   0.69028E+00
 0.39445E+00   0.29584E+00   0.19723E+00   0.98614E-01   0.00000E+00
```

9.11.3. Stoichiometric Mixture Fraction

The stoichiometric mixture fraction for the flamelet library is required for computing the local equivalence ratio of the mixture, which is an important input parameter for the laminar burning velocity.

9.11.3.1. Value

Specify the value for stoichiometric mixture fraction, $0<Z_{st}<1$.

9.11.3.2. Reactants

Specify the reactants and their stoichiometric coefficients for the global reaction. The setup is the same as if it were for the reactants of an equivalent global single-step reaction. For example, if the flamelet library is a single component fuel burnt with air, $\nu_F \, \mathrm{Fuel} + \nu_{Ox} O_2 \rightarrow (\mathrm{products})$, then the stoichiometric coefficient for Fuel would be set to ν_F, and for component O_2, the value would be set to ν_{Ox}.

9.11.3.3. Automatic

Select the automatic option to let the solver determine the stoichiometric mixture fraction from the flamelet library. This is an approximate method; when used, you should check the calculated value reported to the CFX-Solver Output file. For details, see CFX-Solver Output File (Combustion Runs) in the *CFX-Solver Manager User's Guide*.

9.11.4. Laminar Burning Velocity

The laminar burning velocity is an input parameter for the burning velocity combustion model (partially premixed model).

9.11.4.1. Value

Specify the laminar burning velocity. It is recommended that you use an expression that accounts for dependency of the burning velocity on equivalence ratio or mixture fraction. Properties of the unburnt mixture, such as temperature, density, specific heat capacity, and thermal conductivity may be used in order to account for preheating and mixture dependency.

9.11.4.2. Metghalchi and Keck

Built-in correlation fitted for the laminar burning velocities of simple hydrocarbon fuels. The type of fuel is indicated by its carbon index, that is, the number of carbon elements in the fuel molecule:

Molecule	Carbon Index
Methane CH_4	1
Propane C_3H_8	3
Iso-octane C_8H_{18} (gasoline)	8

The correlation accounts for preheat temperature and pressure. However, the flammability limits are assumed to be the same as at (25 [C], 1 [atm]) for all temperatures and pressure. Therefore, the correlation is expected to show its best accuracy at near-standard conditions.

9.11.4.3. Equivalence Ratio Correlation

Laminar burning velocity specified as the product of the burning velocity at reference conditions multiplied by corrections for preheat and pressure dependency. For details, see Equivalence Ratio Correlation in the *CFX-Solver Theory Guide*.

9.11.4.3.1. Reference Burning Velocity

Define the correlation for the laminar burning velocity at reference conditions as a function of the local equivalence ratio of the mixture.

- Beta Function in the *CFX-Solver Theory Guide*
- Quadratic Decay in the *CFX-Solver Theory Guide*
- Fifth Order Polynomial in the *CFX-Solver Theory Guide*

For the quadratic decay and fifth order polynomial options, the fit range defines the range of equivalence ratio for which the polynomials are valid. For equivalence ratios outside the fit range, the reference burning velocity is assumed to decay linearly until it becomes zero at the flammability limit.

9.11.4.3.2. Flammability Limits

Specify the range for equivalence ratio in which the mixture is flammable. The laminar burning velocity becomes zero if the local equivalence ratio is less than the lean flammability limit or larger than the rich flammability limit.

9.11.4.3.3. Preheat Temperature Dependency

Specify the reference temperature and the coefficients b_i for the temperature exponent. When all coefficients b_i are set to zero, there will be no dependency on preheat temperature.

9.11.4.3.4. Pressure Dependency

Specify the reference pressure and the coefficients b_i for the pressure exponent. When all coefficients b_i are set to zero, there will be no dependency on pressure.

9.11.4.3.5. Residual Material Dependency

Specify the coefficients c_i for the residual material dependency of the laminar burning velocity. The residual material dependency is optional. If unspecified or when all coefficients c_i are set to zero, there will be no dependency on the residual material concentration.

9.12. Autoignition Model

A flammable mixture residing at sufficiently high temperature will ignite without further interaction after some temporal delay. This phenomenon is called 'autoignition' or 'selfignition'. Depending on the application this may be either desired or adverse behavior. For example, it is an essential feature for compression-ignition (CI, Diesel, HCCI) engines. For spark-ignition (SI) engines autoignition is usually

unwanted, and its occurrence is known as engine knock. A third example where autoignition may need to be considered is designing the length of a premixing section, for example, for a gas turbine combustor.

Classical combustion models like Eddy Dissipation, Flamelets, Burning Velocity Model or Extended Coherent Flame Model have been developed for high temperature combustion. The purpose of the autoignition model is to extend the primary combustion model with a special treatment for the lower temperature regime, thereby providing a combined model that is applicable over the combined range.

The low temperature regime is characterized by initial breakdown of fuel molecules and buildup of radical species. When the concentration of radicals is sufficiently high, they begin to feed the oxidation and the rate of heat release will increase. The combustion then transitions to the high temperature regime. The net effect of autoignition (or selfignition) is that significant heat release due to combustion occurs if, and only if, a certain delay time has expired. This is modeled by applying an 'Ignition Delay Time' to the combustion model.

Depending on the behavior of the principal combustion model, the reaction must either be suppressed for the delay time or enforced after the delay time has expired. This establishes the two kinds of autoignition models:

1. **Ignition Delay**: suppress combustion until delay time has expired; this is available for non-premixed models (Eddy Dissipation, Flamelet)

2. **Knock**: enforce combustion after delay time has expired; this is available for premixed or partially premixed models (BVM, ECFM).

9.12.1. Ignition Delay Time

The **Ignition Delay Time** defines the time to expire before autoignition occurs. The following options are available:

- **Douaud and Eyzat**: correlation for Gasoline

- **Hardenberg and Hase**: correlation for Diesel

- **Value**: user-defined correlation for ignition delay time (CEL)

9.12.2. Customize Knock Reaction Rate

When autoignition is detected for the **Knock** model, the fuel burns locally according to the **Knock Reaction Rate**. If unspecified the solver assumes a first-order Arrhenius rate with default coefficients. You may customize the coefficients by enabling the **Customize Knock Reaction Rate** option. The following options are available:

- **Arrhenius**: first-order reaction rate proportional to fuel available

- **Value**: specify custom reaction rate (to be applied locally when knock occurs).

The **Solution Change Factor** may be specified in order to limit the fraction of fuel consumed per time step (default=0.6 or 60%). Using this setting ensures that the fuel combustion is resolved by at least few timesteps; this can improve robustness significantly.

9.13. NO Model

The following topics will be discussed:

- Introduction to the NO Model (p. 455)
- Fluid Models (p. 456)
- Initialization (p. 457)
- Solver Parameters (p. 457)

9.13.1. Introduction to the NO Model

The NO model calculates mass fractions of NO (with user extensions) formed in the combustion process. It solves additional transport equations for these variables but does not affect the main combustion calculation. NO is treated as a regular component, but because NO concentrations are typically very low, the effect on the global flow and combustion is negligible.

The NO is created or destroyed through four mechanisms: thermal NO, prompt NO, fuel nitrogen, and NO reburn. The fuel nitrogen mechanism only affects coal and oil combustion. In CFX, the temperature variance and NO are solved together with the other equations. NO is treated as a regular component.

In CFX, the NO model is implemented by means of generic reactions with Arrhenius Temperature PDF reaction rates. Control for these reactions is the same as for Arrhenius rates, but with the following extensions:

- Temperature Limit List: The Upper and Lower Bounds should be specified for temperature integration range (2 values).
- Control for temperature variance transport equation currently is for expert users (editing the CCL in CFX-Solver input files)

The NO formation model consists of several parts:

- Predefined reaction schemes for NO formation that are provided by the `REACTIONS` database (available in CFX-Pre) and that are user-adjustable and extensible.
- Integration of the reaction rates for NO formation over a presumed Probability Density Function (PDF) in order to account for turbulent fluctuations of temperature.
- Solving a transport equation for temperature variance.

9.13.1.1. NO Model with Eddy Dissipation / Finite Rate Chemistry / Combined Model

To model NO, select the reaction scheme with the suffix `NO PDF` (for example, `Methane Air WD1 NO PDF`) when creating your reacting mixture material. This will introduce NO as an additional component and add reactions for thermal and prompt NO.

The fuel nitrogen mechanism can be added by additionally selecting the HCN NO PDF multi step reaction (further adding HCN and HCO to the components).

When the fuel is methane, NO reburn can be enabled by adding the Reburn NO Methane PDF reaction. For other fuels the corresponding reaction is not predefined in the REACTIONS database. However, for a given fuel the reaction may be created by copying from the Reburn NO Methane PDF reaction, and changing the fuel and stoichiometry as appropriate.

9.13.1.2. NO Model with Flamelet Model

When using the Flamelet model, you will need to define a new multi-step reaction scheme before you create your variable composition mixture, as the reaction is a multi-step reaction.

In the **Reaction** details view, create a new multistep reaction scheme and select the following reactions:

- The flamelet model library (for example, Methane Air Flamelet 298K 1 bar)
- Thermal NO O Radical PDF implements NO formation by the thermal NO mechanism using O radical information provided by the Flamelet library. This is in contrast to the standard Thermal NO PDF reaction that approximates O radical concentration from O2 concentration and temperature. There is no need to do this when running the Flamelet model since O is one of the components.
- Prompt NO <fuel> PDF (where <fuel> is the name of the fuel; for example, Prompt NO Methane PDF). This reaction accounts for NO formation by the prompt NO mechanism.

  The fuel nitrogen mechanism and the reburn mechanism can be added in the same way as for the Eddy Dissipation / Finite Rate Chemistry / Combined Model as described above.

The fuel nitrogen mechanism or the NO reburn mechanism can be added in the same way as for the Eddy Dissipation / Finite Rate Chemistry / Combined Model described above. For details, see NO Model with Eddy Dissipation / Finite Rate Chemistry / Combined Model (p. 455).

9.13.1.3. NO Model for Coal Combustion / Hydrocarbon Fuel Model

See Hydrocarbon Fuel Model Setup (p. 417) for a description of how to set up the combustion of a solid hydrocarbon fuel with NO formation.

9.13.2. Fluid Models

The models settings described in this section are set when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

9.13.2.1. Component Details

You should select Transport Equation for the NO component. A transport equation is solved for the NO component when NO is modeled, regardless of the combustion model used.

When running the EDM or Finite Rate Chemistry model, setting **Transport Equation** has the same effect as setting Automatic.

When running the Flamelet model, the flamelet library could also contain NO. In this case, specifying **Transport Equation** would activate the NO model discussed here. Specifying `Automatic` or `Library` would use NO from the library instead.

When the fuel nitrogen mechanism is added, the components HCN and HCO need to be specified with the same option as for NO. For example, when used in combination with the flamelet model, HCN and HCO should be set to `Transport Equation`.

9.13.2.2. Boundary Conditions

For NO, the default value 0 is appropriate in many configurations. When a fuel nitrogen mechanism is used, the HCN component must be set at the boundary according to the nitrogen content in the fuel.

9.13.3. Initialization

The same initialization settings as for regular components can be set (that is, the default initial condition for NO of 0 mass fraction is appropriate for many configurations).

Tip:

The NOx model can have problems if it is enabled for the first few iterations of a simulation. For improved robustness, obtain an established flow field first and then turn the NOx model on.

9.13.4. Solver Parameters

The solver parameters depend on the type of combustion model you are running. The presence of NO does not affect the timestep or equation class settings.

9.14. Chemistry Postprocessing

The Chemistry Postprocessing model calculates material components and reactions one-way coupled to the main simulation. This is appropriate when the component mass fractions and the reaction rates are sufficiently small so that their effect on mixture properties and flow field may be neglected. This is the case for tracer materials or pollutants and their corresponding formation and destruction reactions (for example, the NO model).

Solving materials with postprocessing has the advantage that the corresponding transport equations are solved under the final flow and temperature fields from the beginning and, therefore, fewer iterations are required before convergence is reached. Further, postprocessing allows for running different variations of the pollutant formation models based on the same (frozen) main solution.

The following topics will be discussed:

- Fluid Models (p. 458)
- Solver Parameters (p. 458)

9.14.1. Fluid Models

The model settings described in this section are set under the **Combustion Model** section when creating a domain in CFX-Pre. For details, see Reaction or Combustion Model.

9.14.1.1. Chemistry Postprocessing

Set the check box for chemistry postprocessing to enable the model.

9.14.1.2. Materials List

Specify the materials for postprocessing. When a material is defined simultaneously as a component of the reacting mixture and is under chemistry postprocessing, then the component will be post processed.

9.14.1.3. Reactions List

Set the reactions for postprocessing. It is allowed to specify reactions for postprocessing that are also defined in the reacting mixture material, in which case the reactions will be postprocessed. The result will be identical as if the reactions were only specified under chemistry postprocessing.

9.14.2. Boundary and Initial Conditions

For postprocessing materials, the same boundary conditions and initial conditions apply as for regular components.

9.14.3. Solver Parameters

The solver parameters depend on the type of combustion model you are running. The presence of chemistry postprocessing does not affect the timestep or equation class settings.

9.15. Soot Model

The formation of soot in gaseous flames can result both in significantly enhanced radiative heat transfer, and in particulate pollution. Soot is important in flames in which the carbon to oxygen mole ratio is approaching unity, and its formation can be calculated using the model of Magnussen. For details, see Soot Model in the *CFX-Solver Theory Guide*.

The following topics will be discussed in this section:

- Fluid Models (p. 459)
- Boundary Settings (p. 460)
- Initialization (p. 461)

A material describing soot must be created in the **Material** details view. The effect of soot on radiation can then be specified in the **Radiation Properties** section, as well as its density. For details, see Material Properties Tab in the *CFX-Pre User's Guide*.

9.15.1. Fluid Models

The models settings described in this section are set under the **Combustion Model** section when creating a domain in CFX-Pre. For details, see Domains in the *CFX-Pre User's Guide*.

9.15.1.1. Soot Model

Set the **Soot Model** to `Magnussen` to enable the soot model.

9.15.1.2. Fuel Material

The **Fuel Material** is a required parameter and should be set to the name of the fuel. In this release of CFX, only a single fuel is assumed to generate soot. If there are several fuel components, choose the one with the largest mass fraction and carbon content and increase model coefficients accordingly. For example, set the **Fuel Carbon Mass Fraction** and **Nuclei Formation Pre Exponential Factor** such that the correct value for the following product is obtained:

*Fuel Carbon Mass Fraction * Nuclei Formation Pre Exponential Factor * Fuel Mass Fraction*.

9.15.1.3. Soot Material

The **Soot Material** is a required parameter indicating which material in the Library contains its properties description for Density, Absorption Coefficient, and so on.

9.15.1.4. Fuel Consumption Reaction

This is an optional parameter, but you are strongly recommended to set an appropriate value for it. When using the EDM or Finite Rate Chemistry model, select the fuel break-up reaction. This reaction must be a single-step reaction.

The component specified for **Fuel Material** must be among the reactants of The Fuel Consumption Reaction and only those reactions are offered in the user interface.

If no Fuel Consumption Reaction is specified, a simplified version of the Magnussen model will be applied that is lacking some physical aspects (turbulence effects, soot combustion).

For the Flamelet or related combustion models (Burning Velocity Model, Extended Coherent Flame Model), the `Fuel Consumption Reaction` parameter is not available in the user interface. This is because these models define the fuel consumption by means of a Flamelet library, in contrast to a single reaction step. However, in this case the `Fuel Consumption Reaction` may be specified using CCL functionality (expert use), thereby enabling all features of the soot model in combination with models that do not use a single step for the primary combustion:

1. From the CFX-Pre user interface, create a separate Single Step reaction as follows:

   a. Reactants and products corresponding to the global reaction that is represented by the Flamelet library.

   b. All components of the new reaction should already exist in the Flamelet library.

c. Set the combustion model under the reaction definition to **Eddy Dissipation** and/or **Finite Rate Chemistry**. Additionally, **Finite Rate Chemistry** requires specification of the **Forward Reaction Rate**.

2. From the CFX-Pre user interface, add the newly created reaction to the list of reactions that define the reacting mixture.

3. From the Command Editor in CFX-Pre, use CCL to set the parameter `Fuel Consumption Reaction` under `Soot Model` so that it references the new reaction. The following is an example of the CCL for a single phase case:

```
FLOW:
  DOMAIN: <domain name>
    FLUID MODELS:
      COMBUSTION MODEL:
        SOOT MODEL:
          Option = Magnussen
          Fuel Material = CH4
          Soot Material = Soot
          Fuel Carbon Mass Fraction = 0.75
          Fuel Consumption Reaction = <reaction name>
        END
      END
    END
  END
END
```

where <*domain name*> and <*reaction name*> are the names of the domain and new reaction, respectively.

9.15.1.5. Fuel Carbon Mass Fraction

The mass fraction of carbon in the fuel should be entered (for example, for methane enter `0.75` or `12/16`).

9.15.1.6. Soot Particle Mean Diameter

This is also an optional parameter. The default value is 178.5 angstrom (where 1 angstrom = $1.0*10^{-10}$ m).

9.15.2. Boundary Settings

You can set the amount of soot at inlet and opening boundaries using one of the following two methods.

9.15.2.1. Mass Concentration and Nuclei Concentration

Specify the boundary condition data in volumetric concentrations (for example, soot in kg/m^3 and soot nuclei in mol/m^3).

9.15.2.2. Mass Fraction and Specific Nuclei Specific Concentration

Specify the boundary condition data in specific concentrations (for example, soot in kg/kg and soot nuclei in mol/kg).

9.15.3. Initialization

Default initial conditions are zero for soot and soot nuclei. This is appropriate for most situations. You can optionally specify the volumetric or specific concentration.

9.16. Phasic Combustion

Combustion within different phases can be set up using different models with different phases. Refer to the relevant modeling doc for each type of combustion model for advice.

There is no mass transfer between phases for this release of CFX when setting up the default combustion models. You may set this up using interphase mass transfer, but the rates should be sufficiently small that there is no significant mass source.

9.17. General Advice for Modeling Combusting Flows in CFX

The following topics will be discussed:

- General Procedure for Running Simulations (p. 461)
- Advantages and Disadvantages of Multistep Reaction Mechanisms (p. 462)
- Tips for Improving Convergence (p. 462)
- Advanced Combustion Controls (p. 462)

9.17.1. General Procedure for Running Simulations

For many cases, such as the Combustion Tutorial, the run can be started using automatic initial guesses for most fields and the choice of model in the first part of the tutorial, (EDM) is the most robust of the combustion models. If you are having trouble getting a complex problem to converge, the following steps may aid convergence:

- Begin by specifying a single step reaction using the Eddy Dissipation Model.
- For initial conditions, specify **Oxygen** with a mass fraction of 0.232 and set each of the other species to `Automatic`.
- Allow the problem to converge to a reasonable level.
- Restart the problem with Finite Rate Chemistry or Finite Rate Chemistry/Eddy Dissipation Model using the results from the previous problem as an initial guess.
- Restarting the flamelet model from EDM does not work well because of the different model approach. This is because a previous EDM run would not provide mixture fraction mean and variance fields. Instead, it would be better to reduce the timestep for a few iterations.

Combustion introduces a strong coupling between scalars, energy and momentum. In difficult problems, you may find it helps to calculate a cold flow solution before turning on combustion.

In general for the Finite Rate Chemistry Model, the time scale of the combustion process is smaller, so you may have to continue with a smaller timestep to force the mass fraction and energy equations to converge. The time scale for the Eddy Dissipation Model is equal to the turbulence time scale (k / ε).

In some cases, it may be appropriate to use a larger time scale for the mass fraction equations. This is particularly true when there are regions of small flow velocity in the domain. In these cases, the time needed to convect the components into all areas of the domain may be relatively large.

9.17.2. Advantages and Disadvantages of Multistep Reaction Mechanisms

Multi-step schemes enable predicting a larger number of species and intermediate species like CO. They also apply if the fuel is a mixture of several species (for example, when modeling natural gas as a mixture of CH_4, CO, and H2).

For multi-step schemes, the products limiter in the Eddy Dissipation model should not be turned on, because global extinction of the flame would happen with the commonly used value B = 0.5. Because of this, when applying multi-step schemes to premixed or partially-premixed systems it is recommended that you use the combined FRC/EDM model.

It is important to keep in mind that Finite Rate Chemistry does not account for turbulent fluctuations of temperature when computing temperature dependent reaction rates, unless the **Arrhenius with Temperature PDF** is used for the reaction rates.

Another consideration is that even though several reactions steps can be modeled, this usually remains a strong simplification from reality where dozens of species may be involved in hundreds of reactions simultaneously.

9.17.3. Tips for Improving Convergence

It is often useful to apply a larger timestep to temperature and species than the remaining equations. Try starting with a relatively large timestep to let the material sweep through the domain. You can reduce the timestep later to achieve better resolution of the simulation conditions and, therefore, convergence.

When using the Finite Rate Chemistry model, using a timestep that is too large at the beginning may cause the solver to fail due to reaction rates not converging. Under these circumstances, the timestep must be reduced.

Selecting the expert parameter `monitor ranges` enables you to monitor species mass fractions and temperature. This is useful to check that the reaction has started, in particular when using Finite Rate Chemistry of the combined EDM/Finite Rate Chemistry model.

Additional information on expert parameters is available. For details, see CFX-Solver Expert Control Parameters (p. 603).

9.17.4. Advanced Combustion Controls

Global Dynamic Model Control for combustion takes effect when the EDM product limiter is enabled by setting **Eddy Dissipation Model Coefficient B** to positive. Global Dynamic Model Control for combustion disables the product limiter for the EDM model until after the Transition Iteration. This

allows the specification of no products in the domain as a valid initial condition. Global Dynamic Model Control is always selected for combustion, regardless of the choice selected in CFX-Pre, as it provides the most appropriate solver behavior.

You can disable this feature by setting the **Transition Iteration** to `0` on the **Advanced Options** of the **Solver Control** form in CFX-Pre. Select **Combustion Control** and enter a number for the transition iteration.

Chapter 10: Radiation Modeling

This chapter describes:

10.1. Comparison of the Radiation Models

10.2. Terminology

10.3. Material Properties for Radiation

10.4. Rosseland Model

10.5. The P1 Model

10.6. The Discrete Transfer Model

10.7. The Monte Carlo Model

10.8. General Radiation Considerations

CFX includes several radiation modeling options: The Rosseland model (or Diffusion Approximation model), the P-1 model (also known as the Gibb's model or Spherical Harmonics model), the Discrete Transfer model and the Monte Carlo model.

Many fluid flows of practical interest occur in situations where the fluid and/or the enclosing boundaries are hot. In such situations, the effect of radiant heat transfer may become important. A typical environment where radiation plays a significant role is a furnace or other such combustion chamber.

Two limits can be identified in the way that radiation interacts with a fluid or solid medium. One extreme is the situation where the medium is transparent to radiation at wavelengths in which the majority of the heat transfer occurs. In this case, the radiation only affects the medium by heating or cooling the surfaces of the domain, with no radiant energy transfer directly to the medium. Only the Monte Carlo model should be used for this limiting case. The Discrete Transfer model has also been used in this limit, but with limited success.

The opposite extreme is the situation in which the medium is optically dense, and radiation interacts with the medium throughout the interior of the domain, as well as at surfaces. If the medium is optically dense, radiant energy is either scattered, or absorbed and re-emitted in all directions with a small length scale compared to the size of the domain. This situation is known as the "diffusion limit," because radiant intensity is independent of direction. (Note that there is no assumption that the radiation is "diffuse," in the sense of "rarefied.") In this limit, the Rosseland and P1 models are an attractive alternative to the Discrete Transfer and Monte Carlo models because of their simplicity.

For general cases, ranging from optically thin (transparent) to optically thick (diffusion) regions, like combustion, the Discrete Transfer and the Monte Carlo models more accurately represent the solution of the radiative transfer equation.

The Thermal Radiation Model can be selected from the Fluids/Solids Models form on the Domains tab whenever a heat transfer model has been set to Thermal Energy or Total Energy. The default for the Thermal Radiation Model is None.

Once a thermal radiation model has been selected, two additional submodels must also be chosen: the spectral model and the scattering model. For details, see Spectral Model (p. 479) and Scattering Model (p. 481).

The radiation modeling options in CFX enable you to:

- Set up a gray/non-gray media enclosed by opaque diffuse surfaces, except at openings (inlets, outlets and openings), which are considered fully transparent. Symmetry planes and periodic boundaries are treated as specular surfaces when using the Discrete Transfer or Monte Carlo models.
- Select any of the thermal radiation models in any fluid or porous domains[1]. For solid domains, only the Monte Carlo option is available.
- Set up spectral dependent radiation quantities (material properties, radiation sources, directions, surface properties) via CEL expressions by using any of the available spectral variables: frequency, wavelength in vacuum, or wavenumber in vacuum.
- Setting of boundary conditions at walls, domain interfaces, and open boundaries: inlets, outlets and openings.

10.1. Comparison of the Radiation Models

In problems where thermal radiation is significant, the proper choice of the thermal radiation will affect not only the quality of the solution, but also the computational time it requires. Detailed thermal radiation calculations are time consuming, so proper selection must be made from physical considerations.

For problems in the diffusion or thick limit (τ>5), all the modeling options will produce nearly the same results. Then, the best alternative is a balance between Rosseland and P1 models. As the optical thickness decreases and approaches 1, the P1 model becomes the least expensive alternative. Finally, in the thin limit and for purely transparent cases only the Monte Carlo and Discrete Transfer model should be used. For details, see Optical Thickness in the *CFX-Solver Theory Guide*.

For gray models, where the radiation field is expected to be reasonably homogeneous everywhere (at least on a local basis), and high spatial resolution is required, the discrete transfer method is much more efficient and provides very accurate results if sufficient angular resolution is used.

A major advantage of the discrete transfer method is its fixed sampling in situations where the same mesh is to be used again and again, as in the case of a combined flow-radiation calculation for modeling a combustion chamber. In this case, the ray paths can be calculated once and stored giving a large improvement in efficiency. This is impossible with Monte Carlo because the photon trajectories depend on the absorption coefficient and walls emissivity.

A major problem with discrete transfer is the lack of error information. It is possible to perform angular sub-sampling for surface fluxes, for example, but this does not help with ray effects. If a source contribution has been missed by the complete ray sample, it will also be missed by the sub-sample. This can be dealt with by running a crude Monte Carlo simulation to detect any large errors.

The actual computational time needed by the discrete transfer method can also be very difficult to assess if iterations are needed to converge to the solution, as will be the case if scattering is present. The effi-

[1] As long as radiation does not travel through the solid separating two fluid domains, you may have different radiation models on each side of the solid.

ciency advantage of discrete transfer over Monte Carlo also rapidly disappears when non-gray models with a large number of spectral bands are to be computed. Discrete transfer treats each band independently and so the computational time increases in proportion to the number of bands used. Effectively N separate models are computed for an N-band model. The ray tracking is only done once, however. Because it is the radiative heat transfer that is computed, and the actual spectrum is of no interest, a Monte Carlo simulation is hardly affected by the number of bands as the spectrum is just another independent parameter to be sampled.

Unlike Monte Carlo, all the physical quantities of interest are found at fixed points (due to the fixed sampling and ray discretization), not as surface or volume averages.

10.2. Terminology

The terms in this section include:

- Absorptivity (p. 467)
- Diffuse (p. 467)
- Gray (p. 467)
- Opaque (p. 467)
- Reflectivity (p. 468)
- Spectral (p. 468)
- Transmissivity (p. 468)

10.2.1. Absorptivity

Refers to the fraction of incoming energy that is absorbed at the surface.

10.2.2. Diffuse

Refers to quantities that do not depend on incoming or outgoing direction; however, they might be functions of temperature as well as location.

10.2.3. Gray

Refers to quantities that do not have spectral dependency (that is, they are not a function of frequency, wavelength, or wavenumber).

10.2.4. Opaque

Refers to a surface through which radiation cannot travel (that is, radiation is reflected and/or absorbed at the surface).

10.2.5. Reflectivity

Refers to the fraction of incoming energy that is reflected at a surface, and is a function of direction and frequency.

10.2.6. Spectral

Refers to a quantity that is a function of any of the spectrum variables: frequency, wavelength and wavenumber. The preferred variable is frequency, but to avoid inconsistencies when dealing with non-unitary refractive index materials, the wavelength or wavenumber in a vacuum are also supported.

10.2.7. Transmissivity

Transmissivity, τ, refers to the fraction of incoming energy that travels through the surface; that is, the surface is semi-transparent when $0<\tau<1$, transparent when $\tau=1$ and opaque when $\tau=0$.

10.3. Material Properties for Radiation

There are three material properties that must be defined in the **Material** details view for radiation simulations: absorption coefficient, scattering coefficient and refractive index. For details, see Material Details View: Pure Substance in the *CFX-Pre User's Guide*. For multicomponent flows, set the properties in the **Mixture Properties** section of the **Material** details view because this can save substantial CPU time. For details, see Mixture Properties Tab in the *CFX-Pre User's Guide* and Radiation Properties (p. 95).

Absorption coefficient, scattering coefficient and refractive index may be a function of intensive thermodynamic variables such temperature and pressure, as well as composition. In some applications, the radiative properties are not uniform in the whole spectrum (that is, non-gray media). In this situation, you may specify these properties as a function of the CEL spectral variables: frequency, wavelength in vacuum, or wavenumber in vacuum. The spectral variable is evaluated at the midpoint of the spectral band in frequency space.

10.4. Rosseland Model

The Rosseland approximation assumes that the media is optically thick and that radiant energy emitted from other locations in the domain are quickly absorbed and have no influence in the local transport. This implies that the approximation is not valid near walls. In CFX, special treatment is applied to wall boundaries to over come this limitation. Other boundaries are not given any special treatment.

The Rosseland approximation is extremely convenient to use because it does not solve for an additional transport equation. It is usually valid for an optical thickness/depth greater than 5. It should not be used whenever the optical thickness is below 1 because it will affect robustness of the flow solver.

For details, see General Radiation Considerations (p. 473).

10.4.1. Fluid Models

Information on radiation modeling in multiple domains is available in Domain Considerations (p. 474).

10.4.1.1. Include Boundary Temperature Slip

Optional parameter. For simulations where thermal radiation is by far the most dominant mode of heat transfer, a temperature slip must be allowed at physical boundaries (Siegel and Howell). By default, the temperature slip is not included.

Information on the mathematical implementation of this is available. For details, see Rosseland Model (p. 468).

10.4.1.2. Spectral Model

See Spectral Model (p. 479).

10.4.1.3. Scattering Model

See Scattering Model (p. 481).

10.4.2. Initial Conditions

Because the Rosseland model does not solve for any additional transport quantity, no initial guess or condition is required.

10.4.3. Solver Control

No specific advice is required for this model.

10.5. The P1 Model

The Differential Approximation or P1 adds an additional transport equation to the simulation, consequently it is more costly. The P1 model is valid for an optical thickness greater than 1. For example, the model has proved adequate for the study of pulverized fuel (PF) flames, in regions away from the immediate vicinity of the flame. However, it has been used for lower values with varying success.

The P1 model implementation in CFX only allows for opaque diffuse walls. That is, the diffuse fraction setting would then be ignored.

Open boundaries: Inlets, outlets and openings are treated as fully transparent boundaries. That is, they absorb all outgoing energy, and the incoming energy is computed as a blackbody at either the local temperature or at a user specified external blackbody temperature. For details, see General Radiation Considerations (p. 473).

10.5.1. Fluid Models

Information on radiation modeling in multiple domains is available. For details, see Domain Considerations (p. 474).

10.5.1.1. Spectral Model

See Spectral Model (p. 479).

10.5.1.2. Scattering Model

See Scattering Model (p. 481).

10.5.2. Initial Conditions

The radiation model is an additional energy transport mechanism; it does not create any additional sources of energy (except where the model provides for increased heat flow at boundaries). However, it does not follow that switching on the radiation model results in conservation of total energy within the system. The difference between the incident radiation and emitted radiation represents a heating or cooling effect. There is, therefore, a quantifiable energy "storage" (either positive or negative).

Suppose, for example, a combusting flow is solved with the radiation model initially set to None, and then radiation is turned on. Where the fluid is already hot (for example, owing to combustion), the default initial guess introduces a large amount of radiant energy into the domain. Note that incident radiation scales with the fourth power of the local temperature. This extra energy can take a long time to diffuse out of the domain. The default initial guess can therefore be poor for combusting flows.

By its nature, the incident radiation is more uniform throughout the domain than the medium temperature T_f. Thus, a uniform value of incident radiation everywhere might be a better initial guess. It is recommended that the chosen value for the incident radiation result in as small a change as possible in the energy storage within the flow domain. A suitable level can be obtained by integrating the radiant energy equation over the entire flow domain (Ω): For details, see The P1 Model in the *CFX-Solver Theory Guide*.

$$4\,\sigma \int_{\Omega} K_a\left(T_f^4 - T_r^4\right) dV = \int_{\partial\Omega} \frac{2\,\sigma\,\varepsilon_w}{\left(2-\varepsilon_w\right)} \left[T_w^4 - T_r^4\right] dA \tag{10.1}$$

where $\partial\Omega$ is that part of the boundary where an "emissivity-specified" boundary condition is applied. Hence, an average constant value of T_r^4 may be evaluated from:

$$\overline{T_r^4}\left\{\int_{\Omega} K_a\,dV + \int_{\partial\Omega} \frac{\varepsilon_w}{2\left(2-\varepsilon_w\right)}\,dA\right\} = \int_{\Omega}\left[K_a\,T_f^4\right] dV + \int_{\partial\Omega} \frac{\varepsilon_w}{2\left(2-\varepsilon_w\right)}\,T_w^4\,dA \tag{10.2}$$

where:

$$T_r^4 = \frac{1}{V}\int_{\Omega} T_f^4\,dV \tag{10.3}$$

and V is the total volume.

10.5.3. Solver Control

No specific advice is required for this model.

10.6. The Discrete Transfer Model

This model is based on tracing the domain by multiple rays leaving from the bounding surfaces. The technique was developed by Shah (1979) and depends upon the discretization of the equation of

transfer along rays. The path along a ray is discretized by using the sections formed from breaking the path at element boundaries. The physical quantities in each element are assumed to be uniform.

These rays have to be traced through the domain in the same way that the photons would be tracked in the Monte Carlo model. Therefore, the model description for both Monte Carlo and Discrete Transfer is identical.

For the results to be accurate the elements must be chosen so that the radiation field is reasonably homogeneous inside them. This means, for example, that they must be small enough that the scattering optical depth is less than unity across each element.

Non-gray models are dealt with by treating each band as a separate calculation (possible because scattering and reflection are assumed to be coherent). Tracking is done only once, and the results for the bands are combined to give the total radiative heat transfer.

For details, see General Radiation Considerations (p. 473).

10.6.1. Fluid Models

Information on radiation modeling in multiple domains is available. For details, see Domain Considerations (p. 474).

10.6.1.1. Number of Rays

To determine the direction of the rays, the unitary hemisphere over the face of a parametric element is discretized using spherical coordinates. The span is divided into angles by the number of rays, and rays directions are computed to pass through the center of the angles. In total, the square of the number of rays is traced from an element surface. The default is set to 8.

10.6.1.2. Transfer Mode

The **Transfer Mode** setting defines which radiative transfer mode is enabled. The default value is `Participating Media`; that is, the domain material emits, absorbs, and/or scatters radiation. The `Surface to Surface` option implies that volumetric emission, absorption and scattering are ignored regardless of the specified material properties.

Note:

Solution time may be adversely affected when using the `Surface to Surface` option in a case involving a subdomain that only partially covers the whole domain. With the `Surface to Surface` option, the CFX-Solver tries to reduce the radiation mesh to a single element (because only surfaces are of interest) but the presence of the subdomain causes the number of elements to be at least two, triggering a potentially time-consuming ray trace. This problem applies to both the Discrete Transfer and Monte Carlo models.

The workaround is to use the `Participating Media` option instead of the `Surface to Surface` option and adjust the coarsening controls to produce a coarser radiation mesh than the default. Choosing a coarsening rate that produces a few hundred radiation elements should result in a solution similar to that under the `Surface to Surface` option, but with perhaps not the same resolution of the intensity field at the boundaries.

10.6.1.3. Spectral Model

See Spectral Model (p. 479).

10.6.1.4. Scattering Model

See Scattering Model (p. 481).

10.6.2. Initial Conditions

Because this model is not solving a transport equation, no initial guess or condition is required. However, if a value is needed to properly evaluate the initial radiation term in the energy equation, the advice given for the P1 model should be considered.

10.6.3. Solver Control

See Thermal Radiation Control (p. 476).

10.7. The Monte Carlo Model

The Monte Carlo method simulates the underlying processes that govern the system of interest (that is, the physical interactions between photons and their environment). A photon is selected from a photon source and tracked through the system until its weight falls below some minimum at which point it 'dies.' Each time the photon experiences an 'event,' a surface intersection, scattering or absorption for example, the physical quantities of interest are updated. This process generates a complete 'history' of that photon in the system. Many photon histories need to be generated to get good estimates of the physical quantities of interest in a system. Photon sources are selected (that is, 'sampled') on the basis of emitted radiation, each band being treated independently for non-gray models.

In CFX, the main computational overhead in generating a history is in tracking the photons across the domain. It is therefore essential to produce a balanced description of the domain to efficiently track the photons. This is done by using a coarser mesh for the radiation field than for the flow field under the assumption that the radiation field has less sharp changes than any other transport variables. When the domain material does not emit, absorb, and scatter radiation, there is no need for a mesh in the volume because the radiation transfer is between the boundary surfaces only (see Transfer Mode (p. 473)). For details on mesh coarsening controls, see Thermal Radiation Control (p. 476).

10.7.1. Fluid Models

Information on radiation modeling in solid/multiple domains is available in Domain Considerations (p. 474).

10.7.1.1. Number of Histories

Optional parameter to indicate the total number of histories to be tracked for the simulation. The default value is 10000. This can have a large effect on the accuracy of simulation results. For details, see Monte Carlo Statistics. For information on interpreting Monte Carlo results, see Monte Carlo Model.

10.7.1.2. Transfer Mode

The **Transfer Mode** setting defines which radiative transfer mode is enabled. The default value is `Participating Media` (that is, the domain material emits, absorbs, and/or scatters radiation). The `Surface to Surface` option implies that volumetric emission, absorption and scattering are ignored regardless of the specified material properties.

Note:

Solution time may be adversely affected when using the `Surface to Surface` option in a case involving a subdomain that only partially covers the whole domain. With the `Surface to Surface` option, the CFX-Solver tries to reduce the radiation mesh to a single element (because only surfaces are of interest) but the presence of the subdomain causes the number of elements to be at least two, triggering a potentially time-consuming ray trace. This problem applies to both the Discrete Transfer and Monte Carlo models.

The workaround is to use the `Participating Media` option instead of the `Surface to Surface` option and adjust the coarsening controls to produce a coarser radiation mesh than the default. Choosing a coarsening rate that produces a few hundred radiation elements should result in a solution similar to that under the `Surface to Surface` option, but with perhaps not the same resolution of the intensity field at the boundaries.

10.7.1.3. Spectral Model

See Spectral Model (p. 479).

10.7.1.4. Scattering Model

See Scattering Model (p. 481).

10.7.2. Initial Conditions

Similar to the Discrete Transfer model. For details, see Initial Conditions (p. 472).

10.7.3. Solver Control

See Thermal Radiation Control (p. 476).

10.8. General Radiation Considerations

This section contains advice on the topic of setting up radiation modeling in CFX-Pre.

10.8.1. Domain Considerations
10.8.2. Domain Interface Considerations
10.8.3. Boundary Details
10.8.4. Sources
10.8.5. Thermal Radiation Control
10.8.6. Spectral Model

10.8.7. Scattering Model
10.8.8. Radiometers

10.8.1. Domain Considerations

Whenever a domain material does not emit, absorb or scatter radiation, the `Surface to Surface` transfer mode option should be chosen. This option reduces the computational time without any changes in the results.

The following set of rules applies when modeling radiation in solid domains and in simulations with more than one domain:

- For solid domains where thermal radiation is important, the Monte Carlo model is the only available option.
- For domain interfaces using the Conservative Interface Flux option, the radiation model must be the same on both sides of the interface. That is, for Fluid-Solid, Solid-Solid, or Porous-Solid domain interfaces, the Monte Carlo model is the only available option.
- For a simulation with multiple solid domains, the radiation model can be set independently for each solid domain, subject to the previous rule.

10.8.2. Domain Interface Considerations

The following set of rules applies when modeling radiation through domain interfaces:

- For domain interfaces using the Conservative Interface Flux option, the radiation model must be the same on both sides of the interface.
- For a simulation with radiation and MFR interfaces, the two sides of the interface must be aligned and must have the same shape; no pitch change or orientation change is supported.
- For the Discrete Transfer and Monte Carlo radiation models, rotational periodic GGI interfaces are treated as symmetry conditions.

10.8.3. Boundary Details

The boundary condition options that appear will depend on the type of model you are using. Choices will come from the following:

10.8.3.1. External Blackbody Temperature

This setting represents the effective blackbody temperature of any bodies beyond that boundary. For example, in the instance that a body has a wall temperature of T_w and an emissivity of ε_w, in the absence of any attenuation, the effective blackbody temperature is given by $\left[\varepsilon_w T_w{}^4\right]^{0.25}$. This temperature is not necessarily the same as the local temperature.

10.8.3.2. Local Temperature

The local fluid temperature must be used to account for the incoming radiation energy. It is extremely useful for outlets or openings when the external blackbody temperature is either unknown or much lower than the expected outlet temperature.

10.8.3.3. Radiation Intensity

Specify the mean radiation intensity only, and it is only supported by the P1 model.

10.8.3.4. Radiative Heat Flux

You can specify the radiative heat flux directly, and it is only supported by the P1 model.

10.8.3.5. Emissivity

At opaque boundaries, the emissivity must be supplied. It can be set as a function of spectral variables using CEL expressions when using the Multiband spectral modeling option.

10.8.3.6. Diffuse Fraction

Diffuse fraction represents the ratio between the diffuse reflected energy and the total reflected energy at an opaque boundary. If the boundary is black (that is, unitary emissivity), this value has no meaning because no energy is reflected.

10.8.4. Sources

Non-thermal radiation sources can be set when using either the Discrete Transfer or Monte Carlo radiation models.

These non-thermal radiation sources (fluxes at boundaries) are divided into 2 groups: isotropic and directional. The Directional Radiation Source and Directional Radiation Flux is only supported when using the Monte Carlo model.

The strength of the source or flux can be a function of the spectral variables: frequency, wavelength in vacuum, or wavenumber in vacuum when using the Multiband spectral modeling option.

10.8.4.1. Directional Radiation Source

It allows the specification of the source strength and its direction. The direction can be specified by either Cartesian Components or Cylindrical Components using a local axis.

10.8.4.2. Isotropic Radiation Source

This allows the setting of a single source strength when the source strength is uniform in all directions.

10.8.4.3. Directional Radiation Flux

Specifies collimated non-thermal radiation flux at boundaries. The direction can be set by using Cartesian Components, Cylindrical Components using a local axis, or Normal to Boundary. The external refractive index can be set. If the value is set, the refraction at the boundary is calculated automatically in a manner similar to that for radiation through domain interfaces; see Radiation Through Domain Interfaces in the *CFX-Solver Theory Guide*. If the external refractive index is left unspecified it is assumed to have the same value as the material in the domain and the radiation will pass through the boundary without any refraction or reflection.

10.8.4.4. Combining Radiation Sources

Multiple radiation sources can, in general, be defined on any given locator that is eligible to have a radiation source. However, when using the Monte Carlo and Discrete Transfer models, the following limitations apply:

- For the `Gray` and `Weighted Sum of Gray Gases` spectral model options, multiple radiation sources cannot be defined on the same locator unless they are all isotropic.
- For the `Multiband` spectral model option, a given locator may have multiple radiation sources if the strengths and directions do not overlap within the same spectral band. *Otherwise, the results will be incorrect*.

  To help prevent overlapping spectral bands on the same source locator, you can:

  – Combine multiple isotropic sources into a single source definition (where possible).

  – Combine multiple frequency (or wavelength) dependent sources into a single source using CEL for the strength and, if relevant, direction.

10.8.4.5. Isotropic Radiation Flux

Specifies a directionally uniform, non-thermal radiation flux.

10.8.5. Thermal Radiation Control

The radiation modeling options based on ray tracing, Discrete Transfer and Monte Carlo, require additional controls. These controls are found on the **Advanced Options** tab of the **Solver Control** form.

Because these radiation models require considerable time, there are several ways to strike a balance between accuracy and computer time. This can be done by coarsening the fine mesh used for the flow field, or calculating the radiation field at a different frequency than the other transport equations.

10.8.5.1. Iteration Interval

It sets the frequency of the radiation calculation respect to the flow solver. If left unset, radiation will be calculated at each flow solver iteration (that is, 1).

10.8.5.2. Diagnostic Output Level

When performing the ray tracing calculation, either Discrete Transfer or Monte Carlo, several diagnostics can be written to the CFX-Solver Output file. The output is controlled as follows:

0 - Quiet. No output is reported to the CFX-Solver Output file even if minor problems have occurred. If the solver encounters a fatal error, it will stop automatically.

1 - Minimal. A diagnostic results file is written and warnings are reported. The diagnostic results file includes radiation quantities for each radiation element.

2 - Verbose. A diagnostic results file is written each radiation iteration and the solver will stop even for some warnings. This level is meant for debugging purposes only.

Information about the diagnostic results file is available. For details, see CFX Radiation File in the *CFX-Solver Manager User's Guide*.

10.8.5.3. Ray Reflection Control

When modeling specular boundaries, a ray may be reflected multiple times. When the energy content of the original ray has dropped below the Ray Reflection Threshold (p. 479), the tracing is halted.

Increasing the threshold may reduce accuracy. In this case, energy residuals and imbalances may not be smooth over a restart, in particular when the radiation properties have evolved during the initial run.

Decreasing the threshold will result in longer traces. Because the energy content of the original ray decays along the trace, after a certain run length its contribution becomes negligible. Reducing the threshold below a certain level, thus excessively prolonging the traces, will then have no significant influence on results.

10.8.5.4. Coarsening Control

10.8.5.4.1. Target Coarsening Rate

This represents the target ratio between elements in the fine mesh and radiation elements in the coarser mesh. The default value is 64; that is, the number of radiation elements is 64 times smaller than the number of elements in the fine mesh. The actual coarsening could be smaller than the target specified. A summary is presented in the Radiation Coarsening section of the CFX-Solver Output file.

10.8.5.4.2. Minimum Blocking Factor

This represents the minimum number of elements within a coarser element at a given coarsening level.

10.8.5.4.3. Maximum Blocking Factor

This represents the maximum number of elements within a coarser element at a given coarsening level.

10.8.5.4.4. Small Coarse Grid Size

It is the minimum number of radiation elements in the coarser mesh. The coarsening algorithm will stop when either this value or the target coarsening rate is achieved.

10.8.5.4.5. Diagnostic Output Level

The scope of radiation mesh coarsening diagnostic information written to the CFX-Solver Output file can be controlled by setting the diagnostic output level to 0, 1, or 2.

- 0 - Minimal.

  This is the default diagnostic output level. A basic summary of the radiation mesh coarsening information is written to the CFX-Solver Output file.

- 1 - Minimal.

  Same effect as output level 0.

- 2 - Verbose.

  The CFX-Solver Output file contains a detailed summary of the radiation mesh coarsening information. As well, additional radiation mesh coarsening information is written to the CFX-Solver Output file in the form of a table with the columns: `Crs Level`, `# Elems`, `Avg Nbrs`, `Avg Crs Rate` and `Tot Crs Rate`.

  – `Crs Level` is the progressive level of coarsening.

  – `# Elems` is the number of radiation elements at each progressive level.

  – `Avg Nbrs` is the average number of neighboring elements that each radiation element has.

  – `Avg Crs Rate` is the average coarsening rate achieved at each progressive level, calculated as the ratio of the number of radiation elements (found under `# Elems`) between the current and subsequent progressive levels.

  – `Tot Crs Rate` is the total coarsening rate achieved at each progressive level, calculated as the ratio of the number of radiation elements (found under `# Elems`) between the first and current progressive levels.

10.8.5.5. Ray Tracing Control

This section only applies for the Discrete Transfer model.

10.8.5.5.1. Iteration Interval

Sets the frequency for the calculation of the ray tracks. Because the ray tracing is time-consuming and disk-intensive, it is rarely used. The default value is zero; that is, the tracks are computed only once and stored permanently.

10.8.5.5.2. Maximum Buffer Size

To minimize total memory usage and maximize disk throughput, information for the tracks is stored in a memory buffer before being written to disk. The **Maximum Buffer Size** parameter sets the maximum allowed buffer storage in words. The default value is 6000 words.

10.8.5.5.3. Maximum Number of Track Segments

A single ray is made of tracks, the fraction of ray within a radiation element. In certain cases, highly specular surfaces with a small spacing, the rays may have infinite number of reflections before being totally absorbed. Whenever the number of tracks reaches this maximum, the tracing for this ray is halted, and the next ray is started. Default value is 9000.

10.8.5.5.4. Maximum Number of Iterations

Limits the inner loop when there are reflecting boundaries; otherwise, the first iteration usually satisfies the iteration convergence criterion (1%).

10.8.5.5.5. Iteration Convergence Criterion

Sets the maximum relative radiosity (emission plus reflection) change for convergence of the inner loop when there are reflecting boundaries.

10.8.5.5.6. Ray Reflection Threshold

When modeling specular boundaries, a ray may get reflected multiple times. When the energy content of the original ray has dropped below this threshold, the tracing is halted.

10.8.5.5.7. File Path

Specifies the location in which to store the track files. This is required only when the local disk space is not large enough to store the track files, or a follower in a parallel run does not have access to the run directory.

10.8.6. Spectral Model

In CFX, three different spectral models are supported: Gray, Multiband, and Multigray or Weighted Sum of Gray Gases. Each of radiation modeling option can use the following spectral models:

10.8.6.1. Gray

The Gray model assumes that all radiation quantities are uniform throughout the spectrum. This simplifies the radiation calculation considerably because fewer equations should be solved. For combustion calculations, where certain gases are absorbing in finite regions of the spectrum and transparent for rest, it will introduce errors in the total radiative heat flux.

10.8.6.2. Multiband

The Multiband model discretizes the spectrum into bands of finite width and assumes that radiation quantities are nearly uniform within the band. The total radiative heat flux is computed by adding

the results within each band. Each spectral band can be defined by different means: Frequency, Wavelength in Vacuum, or Wavenumber in Vacuum.

At least two different spectral bands must be set. The solver will check that the union of all spectral bands fully covers the thermal radiation part of spectrum and that they do not overlap. The wavenumber range should cover at least the range between 100 [cm^-1] and 100,000 [cm^-1] to cover the full thermal spectrum. A warning message will be written to the CFX-Solver Output file otherwise.

When using CEL expressions to describe the spectral variation of any radiation quantity, the solver will use the frequency for the spectral band.

10.8.6.3. Multigray/Weighted Sum of Gray Gases

This model assumes that gas absorption can be represented by a weighted sum of gray media. The weights and absorption coefficients have been correlated in the literature for a variety of gases (see Modest [8]). The current implementation does not provide a specific set of weights or coefficients, though there is a CFX-Pre template available. Therefore, you must specify the weights/amplitudes and the absorption coefficients for each gray gas.

Caution should be taken when specifying the set of coefficients:

1. There must be one clear gas.

2. Weights must add up to 1 at all times.

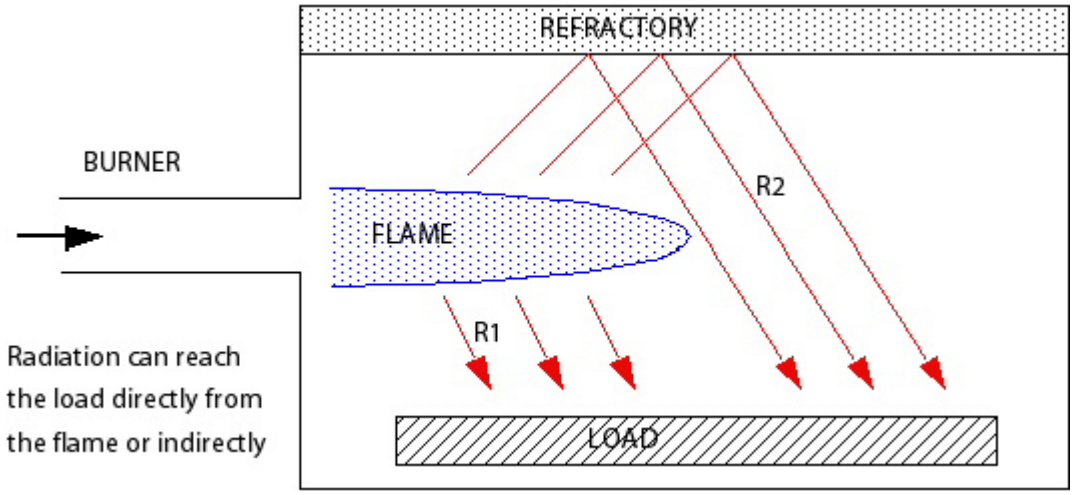
A Multigray CCL Template is available in the Library mode of CFX-Pre. For details, see Multigray Radiation in the *CFX-Pre User's Guide*.

10.8.6.4. When is a Non-Gray Spectral Model Appropriate?

Although it is convenient to average radiation properties over the whole spectrum (Gray model), real gases only absorb and emit in discrete bands. Upon diffuse reflection at walls, radiation emitted by the gases within discrete bands is re-emitted at all wavelengths as thermal radiation with the characteristic blackbody spectrum. Although much of this reflected radiation will be reabsorbed by the gases, some will now lie outside the absorption bands of any of the gases present and will therefore pass through the gas volume without absorption.

This effect can be significant when it is important to distinguish between emission from gases and wall reflection, for example, in a reheat furnace, see Figure 10.1: Example of a Reheat Furnace (p. 481), where a thermal load is being heated directly by a gaseous flame (R1) and indirectly by a refractory wall (R2). If the emissivity of the refractory wall is increased (for example, by a special coating), the proportion of radiation with wavelengths outside the gaseous absorption bands increases and the intensity of radiation reaching the load becomes higher. If the radiative heat transfer in the reheat furnace is modeled using a Gray spectral model, then this effect would not be correctly predicted. The Multiband, and Multigray/Weighted Sum of Gray Gases models include a clear band and correctly predict an increase of radiative heat transfer to the load, as wall emissivity increases in the reheat furnace example.

Figure 10.1: Example of a Reheat Furnace



A more obvious limitation of the Gray model in combustion calculations is that a single absorption coefficient is set, independent of the local gas composition. This implies that the combustion air has the same radiative properties as the combustion products, although the latter contains a high percentage of CO_2 and H_2O, which are highly efficient emitters of thermal radiation. The actual error caused by this approximation is not usually large because combustion air is usually at much lower temperatures than the products but, nevertheless, it leads to an overestimate of the absorption due to the air.

10.8.7. Scattering Model

The radiative transfer equation includes two terms due to scattering: attenuation by scattering or out-scattering and augmentation by scattering or in-scattering.

In CFX, the scattering term can be controlled independently in several ways. When using the Monte Carlo radiation model, you can optionally specify a scattering model. If you are using an option other than None, you must specify a **Scattering Coefficient** on the **Material Properties** form. If you specify a **Scattering Coefficient**, this does not automatically imply that a scattering model will be used, you must also select either the Gray or Linear Anisotropy models. For details, see Material Details View: Pure Substance in the *CFX-Pre User's Guide*.

10.8.7.1. Option = None

The scattering coefficient is ignored, even if a non-zero coefficient has been set for the medium. This option could be used, for example, for cases with clean participating gases, without particles, where the absorption coefficient is much larger than the scattering coefficient.

10.8.7.2. Option = Isotropic

It assumes that in-scattering is uniform in all directions.

10.8.7.3. Option = Linear Anisotropy

CFX includes support for the linear anisotropic phase function

$$\Phi = 1 + A\left(\boldsymbol{s}\cdot\boldsymbol{s}'\right) \tag{10.4}$$

The anisotropy coefficient A must be supplied. If the coefficient has spectral dependency, it can be evaluated using CEL expressions using any of the spectral variables: `frequency`, `wavelength`, or `wavenumber`. By default, the coefficient A is set to zero.

10.8.8. Radiometers

A radiometer is a user-defined point in space that monitors the irradiation heat flux at the required location. For a description of the radiometer settings, see Radiometer in the *CFX-Pre User's Guide*.

A radiometer reports the irradiation heat flux at the specified location using a hemisphere based on the direction provided. The irradiation heat flux is relative to the radiometer temperature (that is, the incoming radiation minus the emission from the radiometer based on its specified temperature). By default, radiometers are ideal and the efficiency factor is 1.

Refer to Variables Relevant for Radiation Calculations in the *CFX Reference Guide* for definitions of radiation variables such as `Irradiation Heat Flux` and `Incident Radiation`.

For each radiometer, the following parameters must be specified:

- **Location**

  The physical location of the radiometer in the domain, in Cartesian or cylindrical coordinates.

- **Temperature**

  The temperature of the radiometer.

- **Quadrature points**

  The number of rays used for tracing from the radiometer location.

- **Viewing direction**

  The direction in which the radiometer points, specified as direction vector components.

The following parameter may be optionally specified:

Diagnostic Output Level

The CFX-Solver will write the radiometer ray traces to a series of polylines in a `.csv` file that can be visualized in CFD-Post. This can be used to determine if the number of quadrature points is optimal. The output is controlled as follows:

- `0` - Quiet. No `.csv` file is written. This is the default.
- `1` - Minimal. At the last radiation iteration, a `.csv` file is written, named `pflux.<radiometer name>.csv`.

- 2 - Verbose. At each radiation iteration, a `.csv` file is written at named `<timestep>_pflux.<radiometer name>.csv`.

Note:

The ray traces out of a radiometer are distributed slightly differently from those out of a boundary. Consequently, for a small number of rays, the value of `Irradiation Heat Flux` at a boundary node may be different than that at a radiometer that is at the same location. However, it will converge to the same value for a sufficient number of rays.

Chapter 11: Rigid Body Modeling

This chapter describes:

11.1. Introduction to Rigid Body Modeling

11.2. Rigid Body Motion

11.3. Modeling a Rigid Body

11.4. CEL Access of the Rigid Body State Variables

11.5. Monitor Plots related to Rigid Bodies

11.6. Solver Control of Rigid Bodies

11.7. Limitations to using Rigid Bodies

11.1. Introduction to Rigid Body Modeling

A rigid body is a solid object that moves through a fluid without itself deforming. Its motion is dictated by the fluid forces and torques acting upon it, plus any external forces (such as gravity) and external torques.

Within Ansys CFX, a rigid body can be modeled in two ways:

- A rigid body can be defined by a collection of 2D regions that form its faces. When a rigid body is modeled in this way, the rigid body itself does not need to be meshed. Mesh motion is used to move the mesh on the rigid body faces in accordance with the solution of the rigid body equations of motion.
- Alternatively, an immersed solid can be defined to be a rigid body. In this case the motion of the immersed solid is dictated by the solution of the rigid body equations of motion.

11.2. Rigid Body Motion

Ansys CFX computes the position and orientation of a rigid body using equations of motion. These equations can provide up to six degrees of freedom ("6 DOF"): up to three translational and up to three rotational degrees of freedom. You have full control over which degrees of freedom are allowed in a given simulation.

In the context of rigid bodies in Ansys CFX, an orientation is represented by a collection of three *Euler angles* that follow the ZYX convention. Under this convention, a reference coordinate frame is reoriented so that it acquires the orientation of interest as follows:

- **Euler Angle Z** modifies the initial orientation by a rotation about the Z axis (using the right-hand rule to determine the direction).

- **Euler Angle Y** then further modifies the orientation by a rotation about the (modified) Y axis (using the right-hand rule to determine the direction).
- **Euler Angle X** then further modifies the orientation by a rotation about the (twice modified) X axis (using the right-hand rule to determine the direction).

Figure 11.1: Rigid Body with Initial Orientation (p. 486) shows a rigid body in an unmodified position. Figure 11.2: Rigid Body Reoriented with Euler Angles Shown (p. 487) shows the same rigid body after being reoriented. The mounting system has pairs of pins at each pivot location. The angle that develops between a pair of pins at a given pivot is the Euler angle associated with the axis on which the pivot was initially located.

Figure 11.1: Rigid Body with Initial Orientation

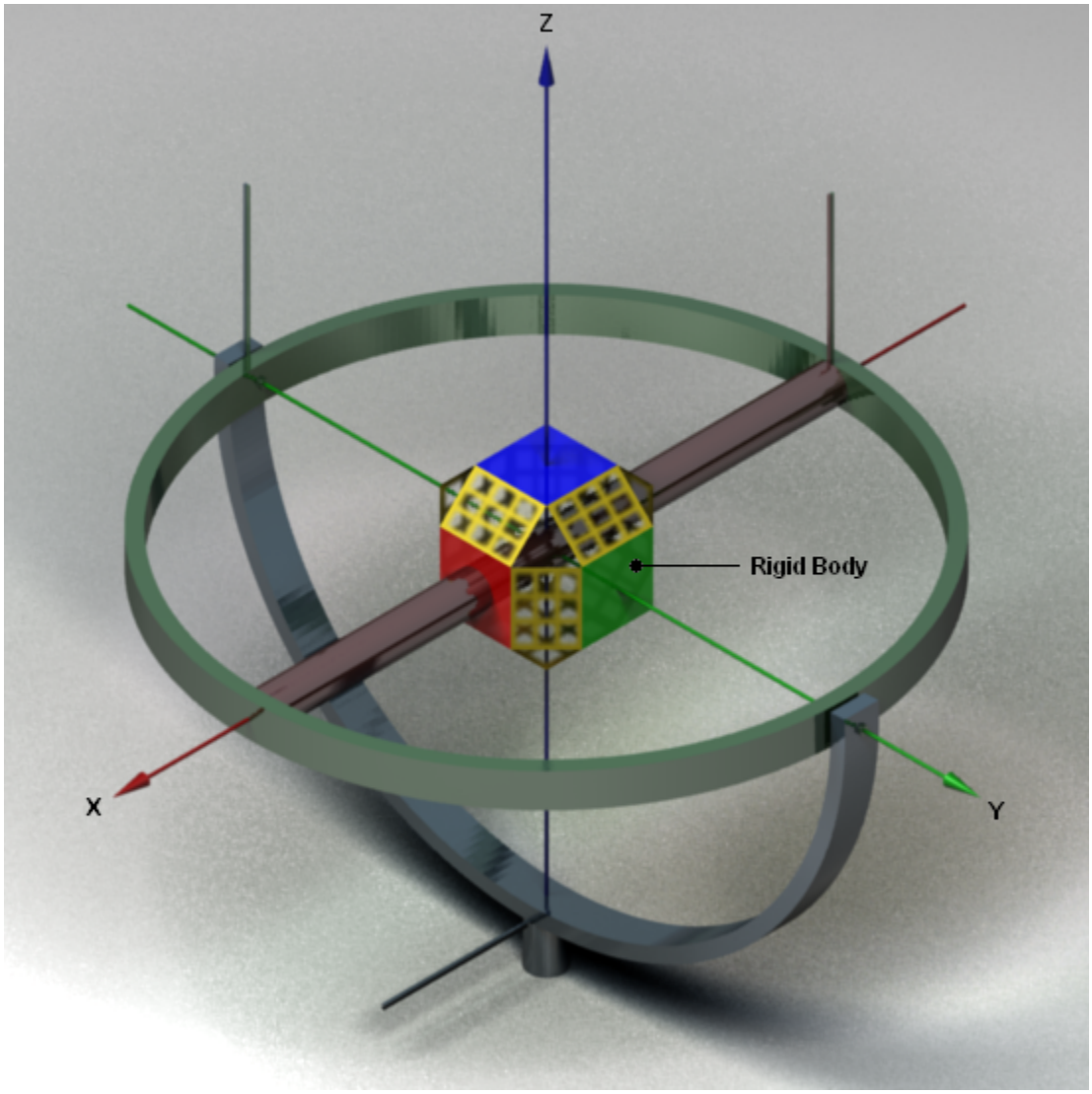
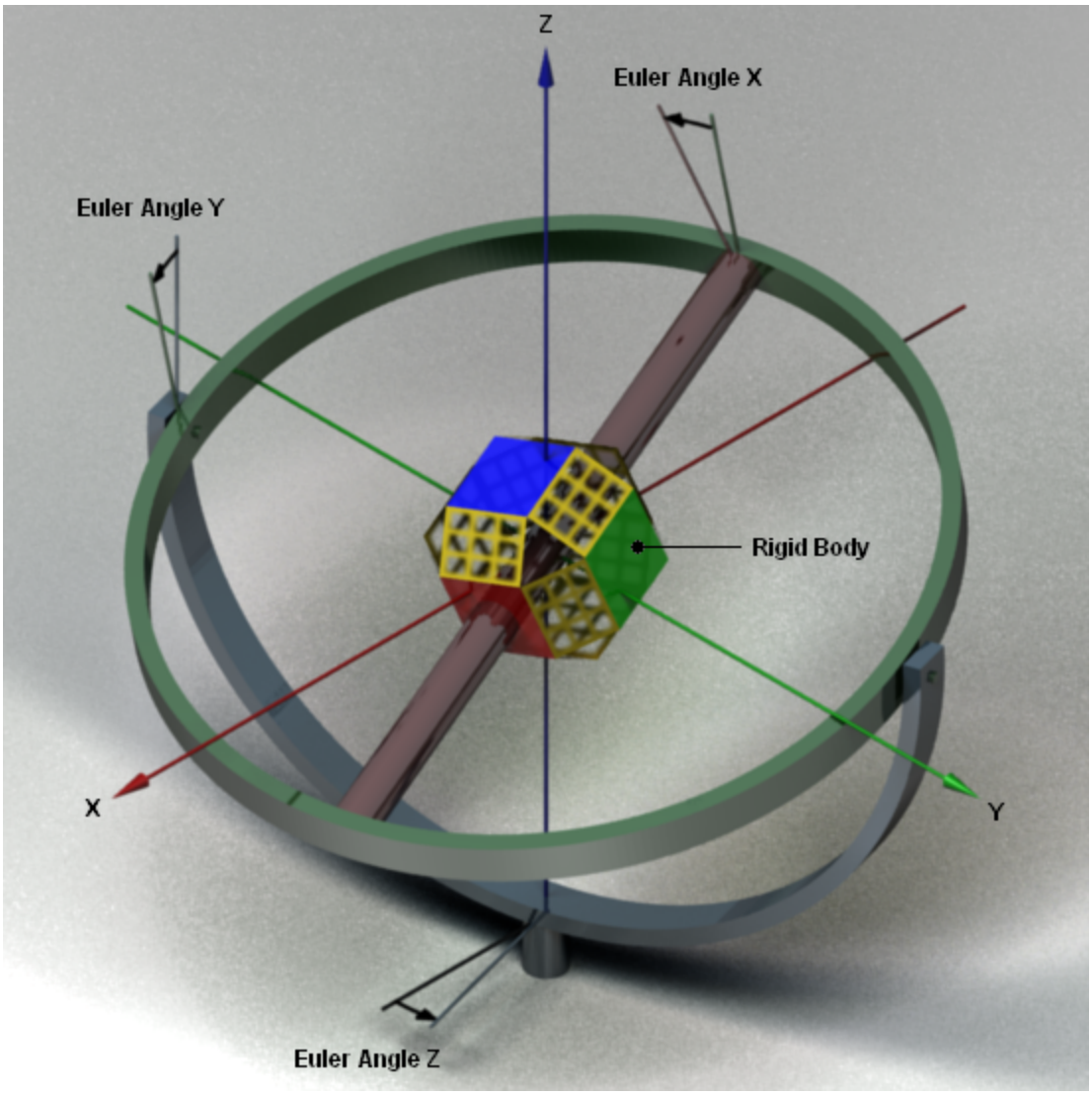


Figure 11.2: Rigid Body Reoriented with Euler Angles Shown



For theoretical details about the rigid body implementations in Ansys CFX, see Rigid Body Theory in the *CFX-Solver Theory Guide*.

11.3. Modeling a Rigid Body

As stated in the introduction, there are two kinds of rigid bodies: those implemented as a collection of 2D faces and those implemented as an immersed solid.

The following sections describe how to model each type of rigid body:

11.3.1. Modeling a Rigid Body as a Collection of 2D Regions
11.3.2. Modeling a Rigid Body using an Immersed Solid

11.3.1. Modeling a Rigid Body as a Collection of 2D Regions

To set up a rigid body as a collection of 2D regions:

1. Insert and define a rigid body object.

   The rigid body object holds settings that define: the rigid body characteristics, information about external forces and torques, which degrees of freedom are to be allowed, and initialization settings for the rigid body. Many of the settings are with respect to the *rigid body coordinate frame*, described in Coordinate Frame in the *CFX-Pre User's Guide*. For details on the parts of the user interface relevant to this step, see Rigid Body User Interface in the *CFX-Pre User's Guide*. The rigid body object, once created, is listed in the **Outline** tree view.

2. Ensure that the mesh motion in the related domain(s) reflects the motion of the rigid body by setting the **Mesh Motion** option for the boundaries that contain the rigid body faces to `Rigid Body Solution`. (Mesh motion requires, as a prerequisite, the **Mesh Deformation** setting of `Regions of Motion Specified` for the domain. For details, see Mesh Deformation in the *CFX-Pre User's Guide*.)

3. As required, apply the **Mesh Motion** option `Rigid Body Solution` to other boundaries (**Boundary Details** tab) and subdomains (**Mesh Motion** tab) in order to move (consistently with the rigid body) parts of the mesh not on the rigid body faces. Properly done, this can help to maintain the mesh quality as the rigid body moves.

4. As required, apply motion constraints to any of the boundaries/subdomains that are governed by the rigid body solution.

   The **Motion Constraints** setting is found in the **Mesh Motion** group of settings, which is found in the **Boundary Details** tab for boundaries and the **Mesh Motion** tab for subdomains.

   As an example of applying a motion constraint, you might have a circular mesh interface surrounding the rigid body, and you might want the mesh on the outside of that interface to not rotate but still translate to follow the rigid body. In this case, you would apply the `Ignore Rotations` constraint on the outer boundary of the interface.

5. As required, adjust the settings that govern the rigid body solver. For a description of these settings, see Rigid Body Control Tab in the *CFX-Pre User's Guide*.

11.3.2. Modeling a Rigid Body using an Immersed Solid

To set up a rigid body using an immersed solid:

1. Define an immersed solid using a **Domain Motion** option of `Rigid Body Solution`.

   For information about modeling immersed solids, see Immersed Solids (p. 60).

2. Configure the rigid body settings found on the **Basic Settings** tab for the immersed solid domain.

   These settings are essentially the same as those on the **Basic Settings** and **Dynamics** tabs in the details view for the rigid body object; those tabs are described in CFX-Pre User's Guide at:

   - Basic Settings Tab
   - Dynamics Tab.

11.4. CEL Access of the Rigid Body State Variables

The rigid body state variables hold the position, orientation, and the first and second derivatives of position and orientation, of a rigid body object or an immersed solid domain that is governed by a rigid body solution. The rigid body state variables can be accessed through CEL in CFX-Pre and CFX-Solver Manager (but not in CFD-Post) via the `rbstate` function. For details on the `rbstate` function, see Quantitative Function List in the *CFX Reference Guide*.

11.5. Monitor Plots related to Rigid Bodies

Using CFX-Pre, you can set up monitors that track the rigid body state by using CEL expressions that access the rigid body state variables. For details, see CEL Access of the Rigid Body State Variables (p. 489), Setting up Monitors in the *CFX-Pre User's Guide*, and Convergence History Plots and User Point Plots in the *CFX-Solver Manager User's Guide*. When you run the simulation, these monitors will be plotted on the **User Points** tab in CFX-Solver Manager.

When you run a simulation that involves a rigid body, you can see the following built-in monitor plots (in addition to any user point plots you have set up in CFX-Pre):

- Rigid Body Convergence

  This plot appears on the **Rigid Body Convergence** tab. If this tab is not available, select the following menu item in CFX-Solver Manager: **Monitors** > **Rigid Body** > **Rigid Body Convergence**.

- Rigid Body Euler Angles

  This plot shows, by default, the Euler angles describing the orientation of the rigid body with respect to the global coordinate frame.

  This plot appears on the **Rigid Body Euler Angles** tab. To see this plot, select the following menu item in CFX-Solver Manager: **Monitors** > **Rigid Body** > **Rigid Body Euler Angles**.

- Rigid Body Position

  This plot shows the position of the center of mass of the rigid body with respect to the global coordinate frame.

  This plot appears on the **Rigid Body Position** tab. To see this plot, select the following menu item in CFX-Solver Manager: **Monitors** > **Rigid Body** > **Rigid Body Position**.

In each of these plots, you can edit the monitor properties (for example, by right-clicking in the plot and selecting **Monitor Properties** from the shortcut menu) to change which plot line variables are plotted. Not all of the variables pertaining to rigid bodies appear in the built-in plots by default; some plot line variables must be manually selected in order for them to appear in a plot. For example to plot the components of angular acceleration, angular velocity, linear acceleration, or linear velocity of a rigid

body, you would have to edit the monitor properties of a plot and select the corresponding plot line variables.

Note:

All of the plot line variables pertaining to a rigid body are with respect to the global coordinate frame.

11.6. Solver Control of Rigid Bodies

The rigid body motion is controlled by the rigid body solver, which is called by the main flow solver. Each call to the rigid body solver is made immediately before the step of processing mesh motion.

You can choose when the rigid body solver is called. You can also change the convergence criteria and amount of under relaxation applied by the rigid body solver. For details, see Rigid Body Control Tab in the *CFX-Pre User's Guide*.

11.7. Limitations to using Rigid Bodies

The following is a list of limitations that apply when using rigid bodies:

- When modeling an immersed solid governed by the rigid body solver, you cannot set initial conditions (as you can for the rigid body object).
- Rigid bodies cannot be modeled in a System Coupling simulation.
- There is no collision modeling to detect when a rigid body contacts a solid wall or another rigid body.
- Rigid bodies cannot be modeled in a rotating domain.
- The built-in spring models are limited. However, you may be able to use CEL expressions to apply forces and torques that simulate a general spring. When working with CEL, you can use the `rbstate` function to get information about the rigid body position and orientation. For details, see rbstate in the *CFX Reference Guide*.
- Only simple motion constraints are possible. For example, there is no straightforward way to constrain the motion of a rigid body so that it rotates about a point other than its own center of mass.
- The rigid body coordinate frame definition should not be changed when a case is restarted.
- Although the rigid body orientation is internally tracked with quaternions, the RBSTATE callbacks report the orientation in terms of Euler angles with respect to the global coordinate frame. Euler angles suffer from the well known limitation of "gimbal lock" and can therefore be misleading. This does not indicate a problem with the rigid body orientation.

Chapter 12: Real Fluid Properties

This chapter describes:

12.1. Setting up a Dry Real Gas Simulation

12.2. Table Interpolation and Saturation Curve Clipping

12.3. Equilibrium Phase Change Model

12.4. Setting up an Equilibrium Phase Change Simulation

12.5. Important Considerations

12.6. Real Gas Property (RGP) File Contents

12.7. Real Gas Property (RGP) File Format

12.8. Parameters in the .rgp File Controlling the Real Gas Model

For many classes of problems, the thermodynamic properties for a gaseous pure substance are closely approximated by the ideal gas equation of state. The assumptions associated with an ideal gas are especially suited for compressible gas flows at low density, which in practical terms can be stated to be appropriate under the following conditions:

1. At very low pressures, ideal gas equations of state can be used, regardless of temperature, with good accuracy as long as no phase change occurs.

2. When the pressure is low relative to the critical pressure and the temperature is high relative to the critical temperature.

For flows where the pressure and temperature ranges vary, such that the density approaches a significant fraction of the critical density, molecular interactions start to become significant and the ideal gas equation of state is no longer appropriate. In this case, a real gas equation of state may be required to give a better representation of the fluid behavior. This includes regimes with flows of dense gases, or liquids at high pressures above the critical point, or, possibly flows with phase change.

Within CFX, you can use real gas equations of state that are appropriate for this type of regime. CFX has the capability to deal with flows of both superheated vapors and subcooled liquids, while at the same time using equilibrium or non-equilibrium models for phase change mass transfer rates.

This documentation describes the real fluid model capabilities within CFX. It describes material property set-up, model set-up and modeling considerations for performing calculations that do not involve phase change (vapors or liquids only) and also calculations that do involve phase change (condensing vapors or liquids with evaporation, cavitation or boiling).

The most general setup in CFX enables both equilibrium and non-equilibrium phase change models to be run, and there are specific, built-in models that apply to both kinds of phase change modeling. From a modeling standpoint, this document mainly focuses on how to set up and run the homogeneous equilibrium phase change model, however much of the advice is equally applicable to the non-equilibrium models as well.

Non-equilibrium phase change models are available when using Eulerian multiphase and Particle Transport. The physical properties of the two phases undergoing phase change are set up in a manner similar to what is described here for both of those approaches. For details, see:

- Thermal Phase Change Model (p. 324)
- Mass Transfer (p. 372).

12.1. Setting up a Dry Real Gas Simulation

There are several methods available to model real gases in CFX:

- Using a real gas cubic equation of state.
- Using the IAPWS equation of state (for water).
- Using Real Gas Property tables (in CFX-TASCflow format).
- Using a general, user supplied, equation of state.

12.1.1. Using a Real Gas Equation of State

It is recommended that one of the Real Gas equations of state (Aungier Redlich Kwong or Peng Robinson) is used because this is the most flexible option in terms of enabling different temperature and pressure ranges. There is no need to deal with `.rgp` files (discussed below) because the flow solver automatically calculates properties. The implementation of real gas cubic equations of state are discussed in the flow solver theory documentation. For details, see Real Gas Properties in the *CFX-Solver Theory Guide*. To use cubic equations of state, the following substance dependent quantities are required:

- Critical Temperature
- Critical Pressure
- Critical Volume
- Boiling Temperature
- Acentric Factor
- Zero pressure (Ideal Gas) specific heat capacity

The default pressure range for this option is 0.01 bar to 10 bar and the default temperature range is 100 K to 1000 K.

Many presupplied materials that use the Real Gas equation of state are contained in the `MATERIALS-redkw.ccl` and `MATERIALS-pengrob.ccl` files and are grouped into a few different categories. The materials are suitable for dry gas flows or flows with change of phase.

12.1.1.1. Redlich Kwong Dry Steam

Only water vapor appears in this group. This material is suitable for real gas flows of steam.

12.1.1.2. Redlich Kwong Dry Refrigerants

This group contains several refrigerants including R11, R12, R123, R1234yf, R124, R125, R134, R134a, R141b, R143, R143a, R152a, R22, R23, R32, R410a, CO_2, and NH_3.

R410a is modeled using critical point properties, acentric factor and ideal gas specific heat coefficients obtained from NIST REFPROP V7.0 assuming a fixed composition of 50% R32 and 50% R125 by mass

12.1.1.3. Redlich Kwong Dry Hydrocarbons

This group contains a variety of common hydrocarbons including Methane (CH_4), Acetylene (C_2H_2), Ethene (C_2H_4), Ethane (C_2H_6), Ethanol (C_2H_6O), Propene (C_3H_6), Propane (C_3H_8), 1-Butyne (C_4H_6), Cyclobutane (C_4H_8), Butane (C_4H_{10}), Pentane (C_5H_{12}), Benzene (C_6H_6), Hexane (C_6H_{14}), Toluene (C_7H_8), Octane (C_8H_{18}), Nonane (C_9H_{20}), and Decane ($C_{10}H_{22}$).

12.1.1.4. Dry Redlich Kwong

This group contains various pure fluids that do not appear in any of the other groups and includes Argon, Carbon Monoxide, Carbon Dioxide, Chlorine, Deuterium, Fluorine, Hydrogen, Hydrogen Sulfide, Helium, Nitrogen, Neon, Oxygen, Para-Hydrogen, Sulfur Dioxide, and Ozone.

12.1.2. Metastable States and Saturation Curve Clipping

When you use a cubic equation of state for a dry vapor calculation or for non-equilibrium phase change calculations (using the thermal phase change model or small droplet phase change model), the flow solver enables the vapor and liquid states to go inside the saturation dome locally. The properties are clipped at the vapor or liquid spinodal curve. The spinodal curve defines the boundary beyond which the equation of state is no longer valid because the local derivative of pressure with respect to volume becomes positive. State points predicted inside the dome, up to the spinodal curve, are called "metastable" because normally they only exist temporarily in small local regions until phase change occurs.

For example, metastable states may occur during the calculation when the vapor cools below the local saturation temperature due to rapid flow expansion (called supercooling) or the liquid temperature locally rises above saturation due to rapid heating or compression (called superheating). For cases without phase change these states can occur and continue to persist depending on the problem set-up.

As an example, Figure 12.1: Pressure Volume Diagram for Water (Peng Robinson Equation of State) (p. 494), and Figure 12.3: Vapour Pressure Diagram for Water (Peng Robinson Equation of State) (p. 495) illustrate what the spinodal curves look like on a pressure-volume diagram and pressure-temperature diagram for water when using the Peng Robinson equation of state. Figure 12.2: Pressure Volume Diagram for Water (Aungier Redlich Kwong Equation of State) (p. 494) and Figure 12.4: Vapour Pressure Diagram for Water (Aungier Redlich Kwong Equation of State) (p. 495) show the curves that are obtained when the Aungier Redlich Kwong equation of state is used.

Figure 12.1: Pressure Volume Diagram for Water (Peng Robinson Equation of State)

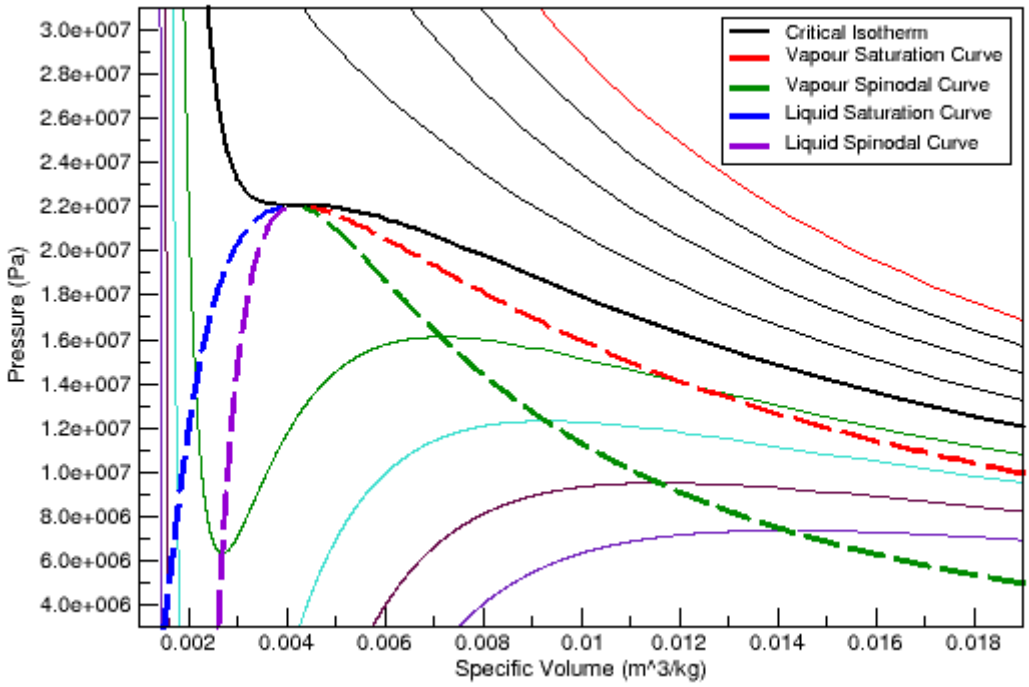


Figure 12.2: Pressure Volume Diagram for Water (Aungier Redlich Kwong Equation of State)

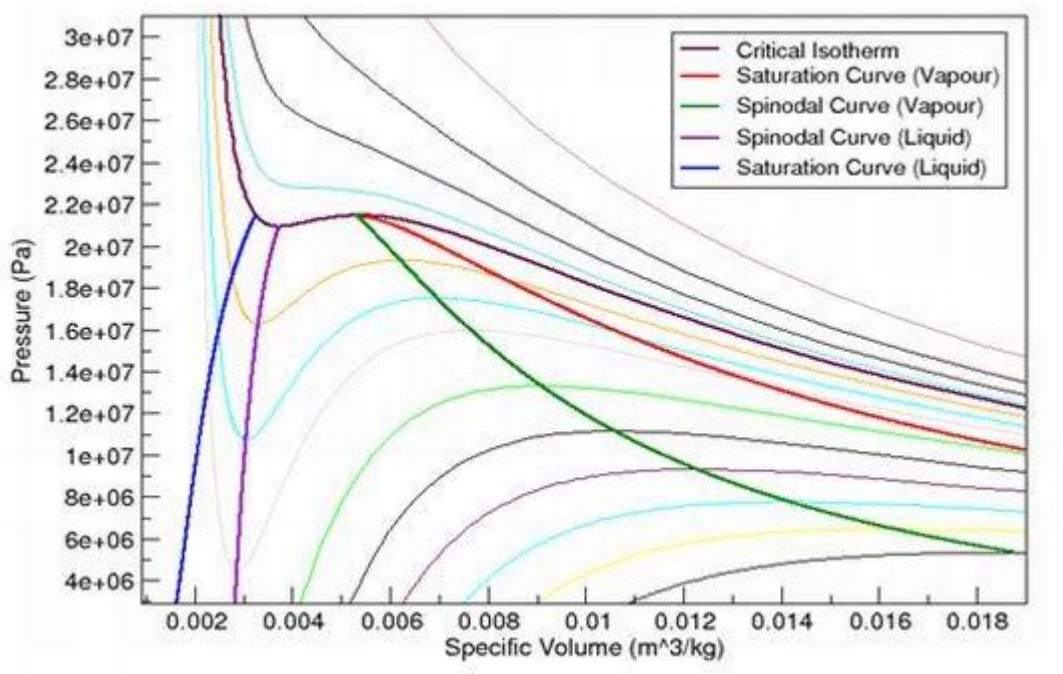


Figure 12.3: Vapour Pressure Diagram for Water (Peng Robinson Equation of State)

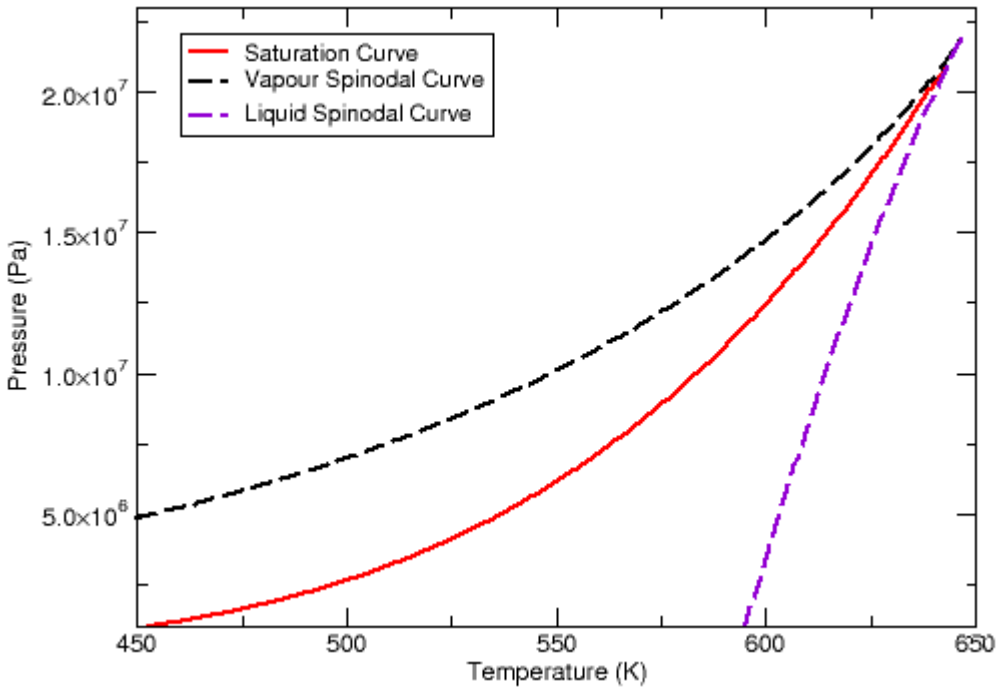
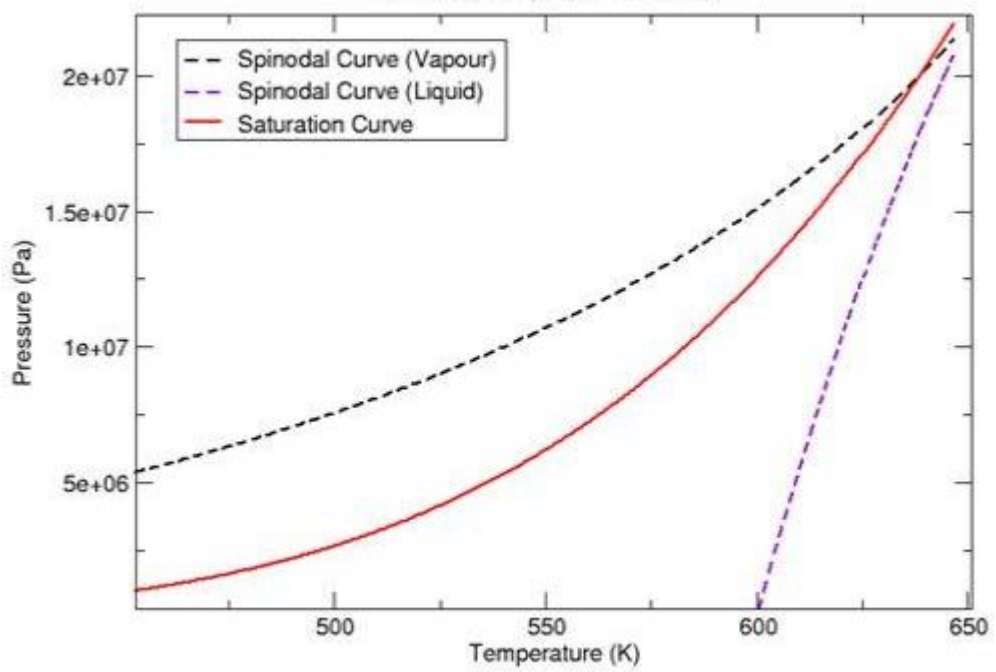


Figure 12.4: Vapour Pressure Diagram for Water (Aungier Redlich Kwong Equation of State)



As you can see, the spinodal states for water vapor enable the vapor to cool below the saturation temperature, and similarly for liquid water heat above the saturation temperature. Normally, though, these states do not remain stable for long before phase change happens.

When you run an equilibrium phase change calculation, it is not necessary to predict metastable states inside the saturation dome, so the vapor and liquid properties are clipped at the saturation curve instead of the spinodal curves.

The spinodal curves are derived directly from the equation of state by finding where $\left(dp/dv\right)\big|_T=0$. The vapor pressure curve can be derived from a general equation of state by Gibbs energy minimization. However, to reduce computational cost, the cubic equation of state models assume the saturation curve is given by an equation from Poling et al. [84]:

$$\log_{10}\left(\frac{p_v}{p_c}\right)=\frac{7}{3}\left(1+\omega\right)\left(1-\frac{T_c}{T}\right) \tag{12.1}$$

where ω is the acentric factor. The acentric factor is a property of a pure fluid and is tabulated in many different thermodynamics references (see, for example, [84]).This equation accurately predicts the saturation curve for most cubic equations of state. If the acentric factor is not tabulated it can easily be estimated using an alternative form of the vapor pressure curve (for example, the Antoine equation) and the following equation:

$$\omega=-\log_{10}\left(\frac{p_v}{p_c}\right)-1.0$$

where p_c is the critical pressure and the vapor pressure, p_v, is evaluated from the Antoine equation, with the temperature set equal to $T=0.7\ T_c$. However, using the Antoine equation also requires that the Antoine coefficients be known.

12.1.3. Additional Comments

As mentioned above, the flow solver also requires that you supply the ideal gas specific heat capacity coefficients. These coefficients allow the flow solver to calculate the specific heat capacity as a function of both temperature and pressure. The **Real Gas Zero Pressure Coefficients** option is used for this purpose and is described in the modeling documentation. For details, see Real Gas (p. 91).

When running this equation of state it is highly recommended that you select the **Rigid Non Interacting Sphere or Interacting Sphere Model** for dynamic viscosity and the **Modified Eucken Model** for thermal conductivity.

The use of the Redlich Kwong cubic state equation model for flows of almost entirely pure liquid (eg: single phase liquid problem, boiling, cavitation) is highly discouraged. If you want to use one of the cubic equations of state for this type of problem then you should use the Peng Robinson equation of state instead, which, by default will force the liquid phase properties to be dependent on temperature and pressure fully consistent with that equation of state.

12.1.4. Using the IAPWS Equation of State

The IAPWS Equation of State is recommended for water calculations. For details, see IAPWS Equation of State in the *CFX-Solver Theory Guide*. There are several predefined IAPWS material definitions available for use, which differ only in the pressure and temperature ranges they cover. If you need to operate outside of these ranges, it is recommended that you choose one of the predefined materials and change the table generation details to cover the required range

12.1.5. Using Real Gas Property (RGP) Tables

12.1.5.1. Loading `.rgp` files

If you have created your own `.rgp` file, you just create a new material in CFX-Pre in the materials tab. On the **Material Properties** form, set the **Option** to `Table` and fill in the filename and component-name fields. For details, see Table.

12.1.5.2. Comments on Pressure and Temperature Ranges

If you use an `.rgp` file for material properties, you must ensure that your application falls within the temperature and pressure ranges used to generate the tables. `.rgp` tables can be generated using your own table generator or, if you have the latest version of CFX-TASCflow, you can create an `.rgp` file using the Real Gas Properties generator. If you do not have CFX-TASCflow, then the built in Real Gas or IAPWS equations of state are a convenient option because the flow solver automatically calculates all necessary properties.

12.1.5.3. Saturation Curve Clipping

Even when you run a dry vapor calculation with an `.rgp` file the flow solver automatically clips the vapor properties at the saturation curve provided in the `.rgp` file. If you create your own `.rgp` file you must ensure that the one dimensional saturation curves stored with each two dimensional table and the critical point have correct values. Note that, if desired, these curves can also be filled in with spinodal data so that an `.rgp` file can be used for modeling non-equilibrium phase change applications. Either way, if these data are not filled in with valid values then the clipping will not work. The correct way to fill in these values is described in a later section. For details, see Real Gas Property (RGP) File Format (p. 510).

12.1.6. Using a General Set-up (CEL or User Fortran)

For any other real fluid calculation, you only need to set up a pure substance, set the **Thermodynamic State** to `Gas or Liquid`, set the properties option to `General Material`, then provide your own equation of state, by supplying an expression (CEL or User Fortran) for the fluid density and specific heat capacity. The flow solver automatically builds tables for both enthalpy and entropy

If you are unsure or not concerned about the real behavior of the specific heat capacity, then you can select the `Zero Pressure Polynomial` option, although note that this makes both specific heat and enthalpy a function of temperature only. For details, see Zero Pressure Polynomial (p. 91).

In addition, when you set the density or specific heat capacity using expressions or User Fortran, you must be careful to ensure that the values are consistent. Because the flow solver uses exact differentials of enthalpy and entropy to calculate tables, the coefficients of the differential terms for both enthalpy and entropy must obey the exact differential property. For details, see Static Enthalpy in the *CFX-Solver Theory Guide*.

For transport properties of real gases, like dynamic viscosity and thermal conductivity, it is recommended that you use one of the built in temperature-dependent models based on the kinetic theory of gases. For example, `Sutherlands Formula` for both properties or, like the Real Gas equations, simply combine the `Rigid Non Interacting Sphere Model` or `Interacting Sphere Model` and `Modified Eucken Model`. For details, see Transport Properties (p. 93). For real liquids,

none of the presupplied transport property models are appropriate and another selection should be used.

12.1.6.1. Saturation Curve Clipping

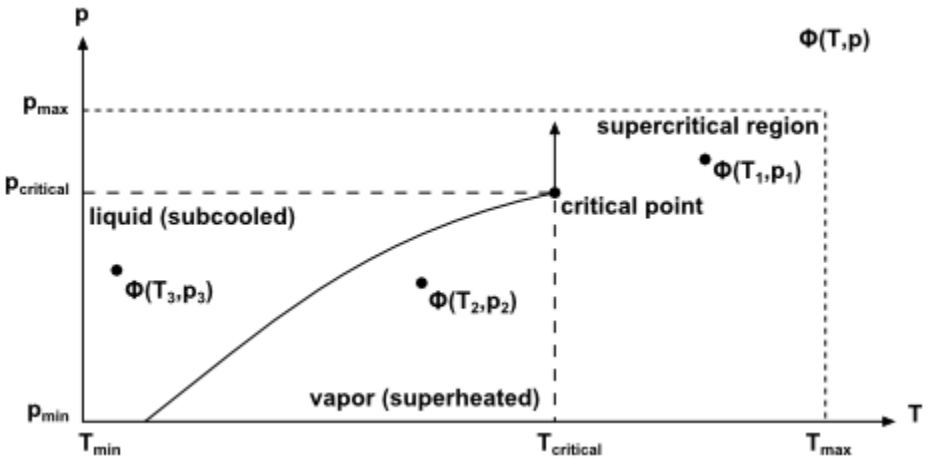
When using a general set-up, the flow solver makes no assumption about the form of the vapor pressure curve so it does not clip any properties at saturation. You will have to be careful to ensure that your equation of state is well-behaved if it approaches or crosses the vapor pressure curve to predict metastable states. Many equations of state are not well behaved inside the dome and robustness problems may result if the enthalpy and pressure (solved for by the flow solver) go past the spinodal curves.

12.2. Table Interpolation and Saturation Curve Clipping

As was mentioned in the previous sections, when running with the Redlich Kwong equation of state or a user supplied `.rgp` file, the flow solver clips two dimensional property tables at the spinodal or saturation curves. For any other definition of properties, no clipping is performed.

Because all properties stored inside the flow solver are in terms of pressure and temperature, the clipping process can be understood by analyzing it on a pressure-temperature diagram, which separates the liquid and vapor regions. The two dimensional tables typically span a given temperature and pressure range and are resolved with a number of points in both directions. Figure 12.5: Typical Pressure-Temperature Diagram (p. 498) shows a rectangular region where the pressure and temperature range span across the critical point.

Figure 12.5: Typical Pressure-Temperature Diagram



Below the critical point, the liquid and vapor regions are separated by the saturation curve and above the critical point, say for $p > p_{\text{critical}}$, there is no difference between the liquid and vapor states.

One problem with evaluating liquid properties is that not all equations of state accurately predict properties in the subcooled liquid region where $T > T_{\text{critical}}$. When using the Redlich Kwong equation of state the flow solver generates vapor property tables that cover the entire pressure and temperature range except for the subcooled liquid region below the critical pressure and temperature. In the vapor region below the critical temperature, the flow solver will automatically clip vapor properties along the spinodal curve or saturation curve as necessary.

When using an `.rgp` file, the one-dimensional saturation tables that are used for clipping the two-dimensional tables must be properly filled in with saturation or spinodal data as required by the model being run. So, for example, if you want to run the equilibrium phase change model, then the vapor values of $h_{\text{sat}}(p)=h(T_{\text{sat}}(p),p)$ for $p<p_{\text{critical}}$ and $h_{\text{sat}}(p)=h(T_{\text{critical}},p)$ for $p>p_{\text{critical}}$ must be supplied. If you generate your own `.rgp` files, you should always generate the saturation table data in this manner.

12.2.1. Table Interpolation

When the flow solver needs to interpolate a value of φ at a given temperature, T, and pressure, p, (for example, the common calculation of $h(T,p)$) there are three possible regions for the interpolation to happen: supercritical (point 1), vapor (point 2) and liquid (point 3).

For any of these regions there are three possible interpolation options available in the flow solver available by setting the following expert parameter:

```
EXPERT PARAMETER:
   prop interp option = 1, 2, or 3 (default)
END
```

Option 1 tells the flow solver to use no saturation clipping, option 2 always uses saturation clipping assuming that saturation data has been extended along the critical isotherm, and option 3 (Default) uses no saturation clipping above the critical pressure.

$\varphi(T_1,p_1)$: Interpolation in the supercritical region

For this region there is a slight difference between option 2 and option 3 interpolation.

Option 2: Interpolation for vapors or liquids is performed the same as if the point were below the critical point.

Option 3: First, the solver checks if $p<p_{\text{critical}}$ and $T<T_{\text{critical}}$. If this condition is met, then the procedures given below are used. If not, then a standard bilinear interpolation is used from the four table values that enclose the desired point. This would correspond to point 1 on the figure.

$\varphi(T_2,p_2)$: Vapor side interpolation below the critical point

Option 2 and option 3 are equivalent in this case.

If $p<p_{\text{critical}}$ and $T<T_{\text{critical}}$ then the flow solver detects if the interpolated value must be clipped to the saturation value. This is done by first looking up the table location where the value of p is located and then calculating $T_{\text{sat}}(p)$ and $\varphi_{\text{sat}}(p)$. If $T>T_{\text{sat}}(p)$ then the solver does not need to clip to saturation and standard bilinear interpolation is used. If $T<T_{\text{sat}}(p)$ then the solver simply sets the interpolated value of φ to $\varphi_{\text{sat}}(p)$.

$\varphi(T_3, p_3)$: Liquid side interpolation below the critical point

Important:

This feature is unsupported for `.rgp` files and the Real Gas equation of state, but is fully supported for the IAPWS equation of state. It can be accessed only when using an additional expert parameter with the Redlich Kwong equation of state or an `.rgp` file and some custom flow solver CCL that forces the flow solver to read the two dimensional tables for a liquid. In a future release this feature will be superseded.

For liquids the recipe is almost the same as for a vapor. First the solver looks up the table location where the value of p is located and calculates $T_{\text{sat}}(p)$ and $\varphi_{\text{sat}}(p)$. If $T<T_{\text{sat}}(p)$ then the solver does not need to clip to saturation and standard bilinear interpolation is used. If $T>T_{\text{sat}}(p)$ then the solver simply sets the interpolated value of φ to $\varphi_{\text{sat}}(p)$.

12.2.2. Table Inversion

There are many cases where the flow solver must calculate one of the table independent variables when given the dependent variable and the other independent variable. For example, the flow solver calculates static enthalpy and pressure from the flow solution. To calculate static temperature, you take those values and invert the $h(T,p)$ table. This is the most common example of table inversion used by the flow solver. The table inversion algorithm that includes saturation clipping, is similar to the interpolation algorithm. The `prop interp option` expert parameter also applies to table inversion.

The following two examples assume that p and φ are given and you need to compute T (φ in this case is usually enthalpy).

$\varphi(T_1, p_1)$: Supercritical table inversion

Option 2: Table inversion is performed in the same way as for below the critical point.

Option 3: Standard table inversion, without saturation clipping is used if $p>p_{\text{critical}}$. If $p<p_{\text{critical}}$ then saturation clipping is used.

An example of standard table inversion is calculating the value of temperature, given enthalpy and pressure. First the location of the value of p is located along the pressure axis. At that location, the solver searches along the two adjacent isobars for the enthalpy values that bound the given value of h. Once the table location is found (both p and h ordinates) the temperature is backed out using bilinear interpolation.

$\varphi(T_2, p_2)$: Vapor side table inversion

Option 2 and option 3 are equivalent in this region.

The recipe is slightly different when you include saturation clipping. First the flow solver looks up the table location where the value of p occurs and calculates $T_{\text{sat}}(p)$ and $h_{\text{sat}}(p)$. If $h>h_{\text{sat}}(p)$ then the

flow solver searches for the table location where h occurs starting at the saturation curve rather than at $T_{\min}$, ignoring the liquid region to the left. Once the flow solver finds both table locations (p and h ordinates) the temperature is backed out using bilinear interpolation. Alternatively, if $h<h_{\text{sat}}(p)$ then temperature is simply set to $T_{\text{sat}}(p)$. (Liquid side inversion works the same way but in the opposite sense.)

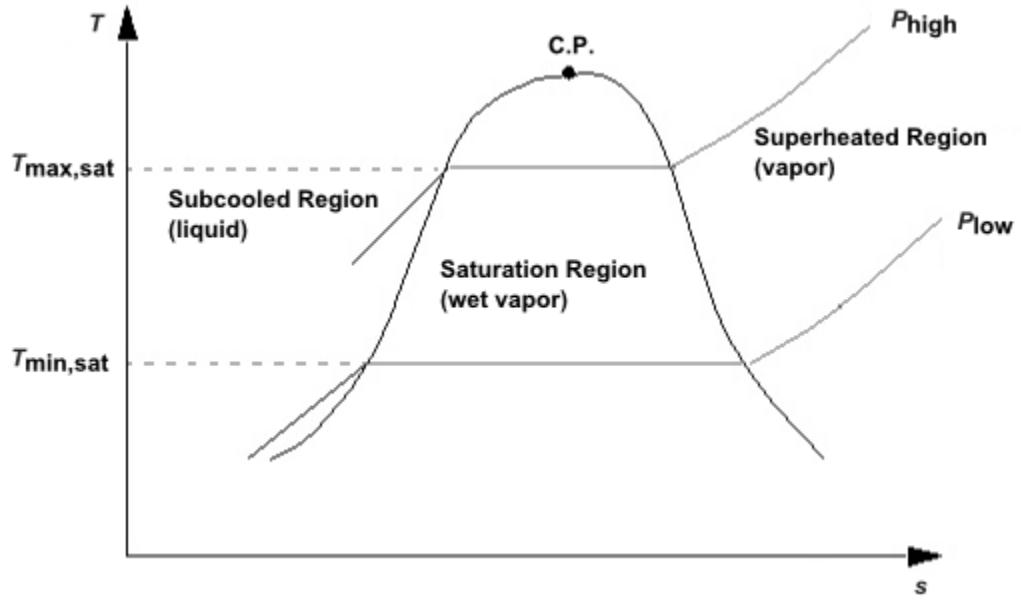
12.3. Equilibrium Phase Change Model

The equilibrium phase change model is a single fluid, multicomponent phase change model. The model is especially suitable for flows of condensing vapors (for example, wet steam or refrigerants) with small liquid mass fractions but it can also be used for some melting or solidification problems as well.

This model assumes that the mixture of the two phases is in local thermodynamic equilibrium. This means that the two phases have the same temperature and that the phase change occurs very rapidly, such that the mass fractions may be determined directly from the phase diagram.

In addition to the superheat region, the equilibrium phase change model requires knowledge of the phase diagram of the material being modeled. Consider Figure 12.6: Phase Diagram (p. 501), which shows two lines of constant pressure that go through the saturation dome on a temperature-entropy diagram for a vapor-liquid mixture.

Figure 12.6: Phase Diagram



To the left of the dome, the entropy is lower than the saturation entropy of the liquid, and the mixture is all liquid. To the right, the entropy is greater than the saturation entropy of the vapor, and the mixture is all vapor. In the remaining region, underneath the saturation dome, the mixture contains both liquid and vapor.

To determine the mass fraction of the vapor, or quality (x), the flow solver uses the lever rule:

$$x=\frac{h_{\text{mix}}-h_{\text{sat,l}}(p)}{h_{\text{sat,v}}(p)-h_{\text{sat,l}}(p)} \tag{12.2}$$

where h_{mix} is the mixture static enthalpy calculated by the flow solver (either directly or from total enthalpy), and $h_{\text{sat,l}}(p)$ and $h_{\text{sat,v}}(p)$ are the saturation enthalpies of the liquid and vapor respectively as a function of pressure. The following observations can be made about the quality:

- When $x<0$, the mixture is 100% subcooled liquid so the liquid properties are selected.
- When $x>1$, the mixture is 100% superheated vapor so the vapor properties are selected.
- When $0\leq x\leq 1$, the mixture contains liquid and vapor. The bulk mixture properties are calculated using the lever rule. Using the mass fraction of vapor and the saturated liquid and vapor properties this gives:

$$\varphi_{\text{mix}}(p)=(1-x_v)\,\varphi_{\text{sat,l}}(p)+x_v\,\varphi_{\text{sat,v}}(p) \tag{12.3}$$

where φ is a property such as entropy, enthalpy, specific heat, thermal conductivity or dynamic viscosity. Because density is volumetric, saturated density is calculated using harmonic averaging instead:

$$\frac{1}{\rho_{mix}}=\frac{1-x_v}{\rho_{sat,l}}+\frac{x_v}{\rho_{sat,v}} \tag{12.4}$$

Derivatives of density with respect to pressure along the saturation line are derived from the harmonic average expression using the chain rule:

$$\left(\frac{\partial\rho_{mix}}{\partial p}\right)_X=\rho_{mix}^2\left(\frac{1-x_v}{\rho_{sat,l}^2}\left(\frac{\partial\rho_{sat,l}}{\partial p}\right)_X+\frac{x_v}{\rho_{sat,v}^2}\left(\frac{\partial\rho_{sat,v}}{\partial p}\right)_X\right) \tag{12.5}$$

where we have not yet decided which variable, X, is held constant. If the densities are a function of pressure, which they usually are, expressions for the speed of sound can be derived using Equation 12.5 (p. 502), and the standard thermodynamic relationship:

$$\frac{1}{c_{mix}^2}=\left(\frac{\partial\rho_{mix}}{\partial p}\right)_X \tag{12.6}$$

For the purposes of the equilibrium phase change model the isentropic speed of sound is output as the variable "Local Speed of Sound" and is computed as follows:

$$\frac{1}{c_{mix}^2}=\left(\frac{\partial\rho_{mix}}{\partial p}\right)_s=\frac{c_{v,mix}}{c_{p,mix}}\left(\frac{\partial\rho_{mix}}{\partial p}\right)_T \tag{12.7}$$

substituting Equation 12.5 (p. 502) held at constant temperature. The mixture specific heat capacities are evaluated using Equation 12.3 (p. 502). Equation 12.7 (p. 502) is applied in all regions, however, the mixing rule for the isothermal derivative, Equation 12.5 (p. 502), is only applied in regions where the equilibrium fraction is non-zero or unity.

While the formulas just presented assume that you are modeling a mixture of liquid and vapor, the same algorithm also holds for other types of phase change, such as the melting and solidification of solid and liquid water, respectively.

12.4. Setting up an Equilibrium Phase Change Simulation

The Equilibrium Phase Change model requires that consistent material properties be supplied for the two pure substance states involved in the phase change, as well as the saturation curve (vaporization curve for vapor-liquid phase change, fusion line for liquid-solid phase change or the sublimation line

for solid-vapor phase change). Although general material properties can be used with this model, in general, it is best practice to use this model with a real gas equation of state for the liquid, vapor and saturation properties. Saturation properties are defined by using a Homogeneous Binary Mixture (HBM). A homogeneous binary mixture is a mixture of two states (solid, liquid or vapor) of the same pure substance.

There are three methods available to set up saturation properties in CFX:

- Using a pre-calculated Real Gas Property `.rgp` file.
- Using the Real Gas equation of state library.
- Using a general set-up, by incorporating two consistent materials (representing the two thermodynamic states) into your own homogeneous binary mixture. For details, see Saturation Properties Tab in the *CFX-Pre User's Guide*.

12.4.1. Using a Real Gas Equation of State

This method is only appropriate for vapor-liquid phase change and cannot be used for other types of phase change. It is required that each material in the mixture uses the same Real Gas equation of state with all the appropriate data set. CFX presupplies a number of suitable materials, which are categorized into several groups, that contain three materials for each pure substance. In each group the vapor material has a "v" at the end of the material name, the liquid has an "l" and the homogeneous binary mixture has a "vl" at the end.

12.4.1.1. Redlich Kwong Wet Steam

Only water vapor, liquid water and the homogeneous binary mixture of the two phases appear in this group. Material is suitable for condensing steam flows with small liquid mass fractions.

12.4.1.2. Redlich Kwong Wet Refrigerants

This group contains several refrigerants including R11, R12, R123, R124, R125, R134, R134a, R141b, R143, R143a, R152a, R22, R23, R32, R410a, CO_2, and NH_3.

R410a is modeled using critical point properties, acentric factor and ideal gas specific heat coefficients obtained from NIST REFPROP V7.0 assuming a fixed composition of 50% R32 and 50% R125 by mass.

12.4.1.3. Redlich Kwong Wet Hydrocarbons

This group contains a variety of common hydrocarbons including Methane (CH_4), Acetylene (C_2H_2), Ethene (C_2H_4), Ethane (C_2H_6), Ethanol (C_2H_6O), Propene (C_3H_6), Propane (C_3H_8), 1-Butyne (C_4H_6), Cyclobutane (C_4H_8), Butane (C_4H_{10}), Pentane (C_5H_{12}), Benzene (C_6H_6), Hexane (C_6H_{14}), Toluene (C_7H_8), Octane (C_8H_{18}), Nonane (C_9H_{20}), and Decane ($C_{10}H_{22}$).

12.4.1.4. Wet Redlich Kwong

This group contains various pure fluids that do not appear in the other groups and includes Argon, Carbon Monoxide, Carbon Dioxide, Chlorine, Deuterium, Fluorine, Hydrogen, Hydrogen Sulfide, Helium, Nitrogen, Neon, Oxygen, Para-Hydrogen, Sulfur Dioxide and Ozone.

12.4.1.5. Creating a Liquid Phase Material

If you have already created a vapor phase material, then an easy way to set the data for the liquid phase material is to make a copy of the vapor phase material, change the name to something appropriate, and then simply update the **Thermodynamic State** to **Liquid** instead of **Gas**.

12.4.1.6. Creating the Homogeneous Binary Mixture

When creating the homogeneous binary mixture, select the two materials that make up the mixture in the **Material 1** and **Material 2** fields. The thermodynamic state of one fluid should be liquid and the thermodynamic state of the other should be gas. Under **Saturation Properties**, set the **Option** to `Redlich Kwong`. No other option is available or valid.

12.4.2. Using Real Gas Property (.rgp) table files

12.4.2.1. Loading an .rgp file

Create the liquid and vapor phase materials as described in the dry RGP section. `.rgp` files can only be used for liquid-vapor phase change calculations.

12.4.2.2. Creating the Homogeneous Binary Mixture

When creating the homogeneous binary mixture, select the two materials that make up the mixture in the **Material1** and **Material2** fields. The thermodynamic states must be Liquid-Gas. Under **Saturation Properties**, set the **Option** to `Table` and provide the table name (which will be the same as the table for the two components already referenced). The **Component Name** is a required field, because `.rgp` files can hold data for more than one pure substance. The saturation date must exist in the `.rgp` file.

12.4.3. Using a General Set-up

You may also combine two materials together to create your own homogeneous binary mixture. To do this, first set up the two materials representing the two thermodynamic states.

It is very important that the two material definitions use consistent reference states for the enthalpy and entropy. The difference between the two material enthalpies must equal the latent heat at the selected reference temperature and pressure. To make this easier, CFX-Pre allows homogeneous binary mixtures to be created only from materials within the same material group.

When creating the homogeneous binary mixture, select the two materials that make up the mixture in the **Material1** and **Material2** fields. The thermodynamic state of one fluid should be liquid and the thermodynamic state of the other should be gas.

Under **Saturation Properties**, set the **Option** to `General`. You must then set the saturation properties (vapor pressure as a function of temperature) for the mixture. This may be done in one of two ways.

12.4.3.1. Constants or Expressions

If you choose to use constant saturation conditions, then, because they are dependent, you should set values for saturation pressure and temperature that are consistent. For example, for a mixture of saturated liquid and water vapor at 1 atm, the saturation temperature is 100 °C; for a mixture of saturated liquid and water vapor at 100 °C, the saturation pressure is 1 atm.

If you use CEL expressions or User Fortran for both pressure and temperature, you should ensure that your function or expression guarantees that $T_{\text{sat}}\left(p_{\text{sat}}\left(T\right)\right)=T$ for all T on the saturation curve.

12.4.3.2. Antoine Equation

You can choose to use this equation for the saturation pressure and supply the flow solver with the necessary model constants. These are tabulated in several thermodynamics texts, see, for example, [84]. Automatic is the only option allowed for temperature in this case because the flow solver calculates saturation temperature from the Antoine equation.

The Antoine Equation correlates vapor pressure as a function of temperature using three coefficients as follows:

$$p_{\text{sat}}=p_{\text{scale}}\exp\left(A-\frac{B}{T+C}\right) \tag{12.8}$$

where p_{scale} is the Pressure Scale used to scale the units for the vapor pressure, A is the Antoine Reference State Constant, B is the Antoine Enthalpic Coefficient and C is the Antoine Temperature Coefficient. Depending on the reference you use to obtain the constants, Equation 12.8 (p. 505) can also be written in different forms, so some care should be taken when entering coefficients from the literature. For example, the tables in [84] assume that the Antoine equation is written as follows:

$$\log_{10} p_{\text{sat}}=A-\frac{B}{T+C-273.15\,K} \tag{12.9}$$

where p_{sat} is in bars and p_{scale} is 1.0 bar. So, if you are running the flow solver in the SI unit system, this requires that you add 5.0 to A and multiply that result by the natural logarithm of 10.0, multiply B by the natural logarithm of 10 and subtract 273.15 K from the tabulated values of C. If you are running in a non-SI unit system this would require that you also convert the constants to return the right units of pressure.

12.4.4. CFX-Pre Domain Models Set-up

The equilibrium phase-change model is set up like any other multi-component fluid (MCF) mixture calculation.

1. You should select the **Homogeneous Binary Mixture** as your domain fluid.
2. On the Fluid Models form, you should select one of the mixture components to be the **Equilibrium Fraction** and the other to be the **Equilibrium Constraint**.

3. Now proceed to set up initial and boundary conditions just as you would for a standard MCF model.

Hint: For liquid-vapor calculations, if you chose the vapor phase, then the equilibrium fraction is the "quality." If you chose the liquid phase, then the equilibrium fraction is the "wetness."

12.5. Important Considerations

The following topics will be discussed:

- Properties (p. 506)
- Inlet Boundary Conditions (p. 506)

12.5.1. Properties

If you use your own RGP table and at any point during the solution, the temperature or pressure falls outside of the range in the table, you will receive a notice from the CFX-Solver that it has clipped or extrapolated the value and that the calculation will continue. For example, this may occur during the first few iterations as the solution begins to settle down from the initial guess. If the messages do not persist then you can safely continue with the solution. If these message do persist, then you should generate a table with a larger pressure and/or temperature range.

Pure substances that use real equations of state (non-ideal or non-constant density) can be used in the same way as any other material in CFX. They can form components in multicomponent fluids or be used in multiphase simulations.

You cannot specify mixture properties for homogeneous binary mixtures.

All boundary condition specifications use real fluid relations for conversions from total to static conditions.

12.5.2. Inlet Boundary Conditions

If you are using the total pressure and total enthalpy boundary condition with the equilibrium phase change model then you do not have to set the equilibrium fraction, this is automatically derived for you by the flow solver. If the value is set, then the flow solver will ignore it.

Alternatively, if you are using the total pressure and total temperature boundary condition with the equilibrium phase change model, you have the option to enter the value of the equilibrium fraction.

The flow solver applies the specified value of equilibrium fraction as a "total" equilibrium fraction value. If the total conditions are dry vapor, this does not necessarily mean that the static conditions are also dry. For example, if the inlet equilibrium fraction is set to be all vapor (total quality is 1), then the flow solver uses the specified total temperature and pressure, as a starting point to determine the static conditions. If the total conditions were just above the saturation dome, then it is very likely that the final static condition is under the dome and the inlet condition will be a bit wet.

If you specify a value for quality between zero and unity then you are saying that the inlet total conditions are under the dome, and total temperature and pressure are no longer independent. In this case, to ensure consistency with the properties, the flow solver ignores the specified total tem-

perature and evaluates total temperature from the vapor pressure curve instead, that is, $T_{\text{total}}=T_{\text{sat}}\left(P_{\text{total}}\right)$.

12.6. Real Gas Property (RGP) File Contents

Evaluating fluid properties through a complex equation of state during a solution can lead to significant computing overhead. To avoid this computational expense, CFX uses a number of data tables during the CFD solution. Accessing these tables is much faster than evaluating properties directly from the equation of state.

12.6.1. Superheat Region

The superheat region in Figure 12.7: Representation of Superheat Tables Associated with a Material (p. 508) is represented through the following nine tables as functions of temperature and pressure:

1. $h\left(P,T\right)$: Specific enthalpy.
2. $c\left(P,T\right)$: Speed of sound.
3. $v\left(P,T\right)$: Specific volume.
4. $c_v\left(P,T\right)$: Specific heat at constant volume.
5. $c_p\left(P,T\right)$: Specific heat at constant pressure.
6. $\left(\partial P/\ \partial v\right)_T\left(P,T\right)$: Partial derivative of pressure with respect to specific volume at constant temperature.
7. $s\left(P,T\right)$: Specific entropy.
8. $\mu\left(P,T\right)$: Dynamic viscosity.
9. $k\left(P,T\right)$: Thermal conductivity.

All of the tables, with the exception of dynamic viscosity and thermal conductivity are required. Dynamic viscosity and thermal conductivity can also be specified with constant values in the `.rgp` file header and as constants or CEL expressions in the CCL material definition. Constant values of dynamic viscosity and thermal conductivity are often acceptable in turbulent flows because turbulent diffusion normally dominates molecular diffusion processes.

The superheat tables, in a discrete manner, represent functions dependent on p and T and are shown in graphic form in Figure 12.7: Representation of Superheat Tables Associated with a Material (p. 508), where φ represents any one of the properties h, c, v, c_v, c_p, $\left(\partial P/\ \partial v\right)_T$, s, μ and k. As indicated in Figure 12.7: Representation of Superheat Tables Associated with a Material (p. 508), for a given material, saturation conditions may run through all of the superheat tables. To handle this case, associated 1D tables (for each 2D superheat table) must be included in the `.rgp` file and represent $\varphi_{\text{sat}}\left(P\right)$ and $T_{\text{sat}}\left(P\right)$, which are shown as graphs in Figure 12.8: Representation of 1D Tables for a Saturation

Property in Terms of Pressure (p. 508). With these 1D tables, property evaluations near saturation can be consistently handled using bilinear interpolation. If the pressure range of the two-dimensional table goes above the critical pressure then the one-dimensional $\varphi_{\text{sat}}(P)$ and $T_{\text{sat}}(P)$ curves should be set to $\varphi(T_{\text{sat}}(P),P)$ and T_{crit}, respectively. It is very important to get this right because the flow solver relies on these data being correct in order to accurately interpolate values from your tables.

Figure 12.7: Representation of Superheat Tables Associated with a Material

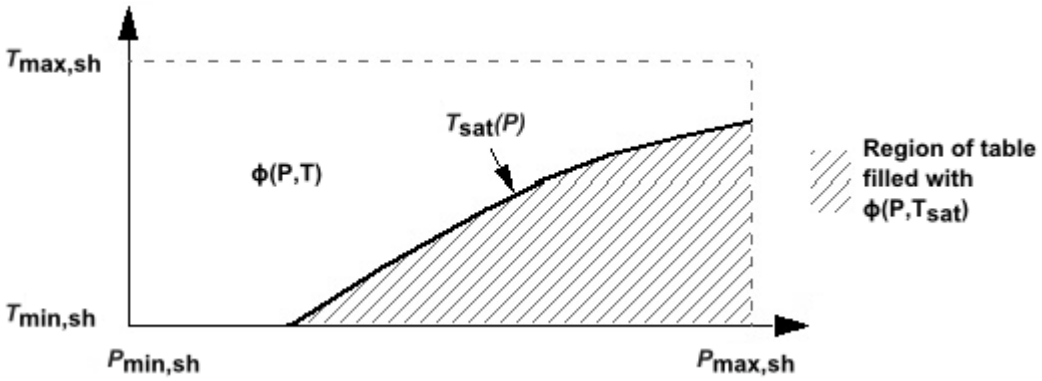


Figure 12.8: Representation of 1D Tables for a Saturation Property in Terms of Pressure

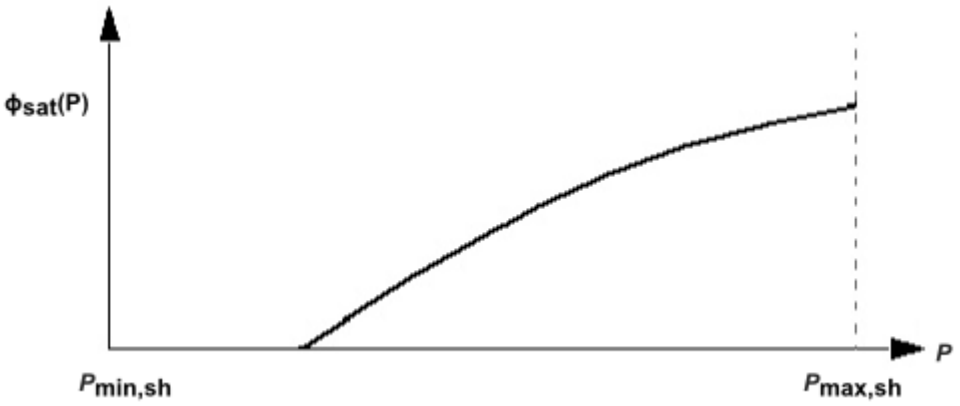
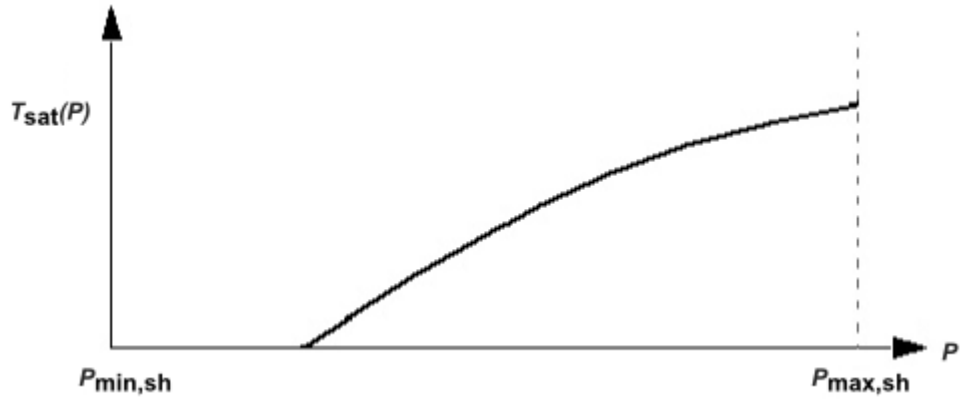


Figure 12.9: Representation of 1D Tables for Saturation Temperature in Terms of Pressure



12.6.2. Saturated Region

Up to this point, the description of the data tables has been directed toward the superheat tables. For the wet/dry equilibrium real fluid model, however, data tables for equilibrium saturation information (as functions of saturation pressure) are also required.

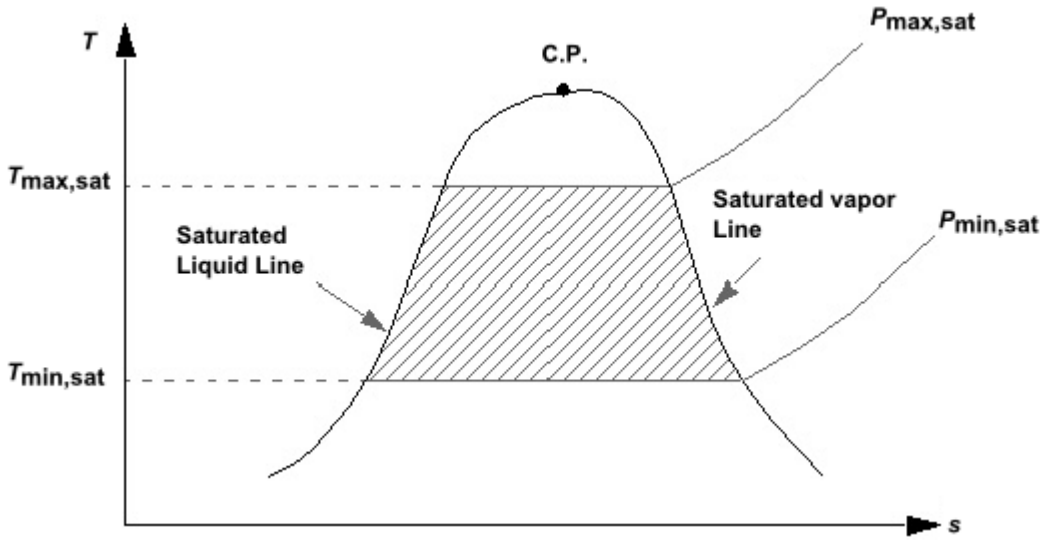
These tables hold property information related to the saturated liquid and vapor lines of the material, as shown in Figure 12.10: Representation of a Saturation Table Associated with a Given Material (p. 510), and can be classified as either single valued or dual valued. Single valued information contains saturation properties that are required along the saturated vapor line or along the saturated liquid line (not both). The properties of this type used in CFX, are:

1. Saturation pressure.
2. Saturation temperature.

Dual valued information contains saturation properties that are required along both the saturated vapor and liquid lines as a function of saturation pressure. Properties of this type required for CFX are:

1. Saturated liquid and vapor enthalpy, h_{l} and h_{g}.
2. Saturated liquid and vapor specific heat at constant pressure, c_{pl} and c_{pg}.
3. Saturated liquid and vapor density, ρ_{l} and ρ_{g}.
4. Saturated liquid and vapor change in density with pressure at constant temperature.
5. Saturated liquid and vapor entropy, s_{l} and s_{g}.
6. Saturated liquid and vapor specific heat at constant volume, c_{vl} and c_{vg}.
7. Saturated liquid and vapor speed of sound, c_{l} and c_{g}.
8. Saturated liquid and vapor molecular viscosity, μ_{l} and μ_{g}.
9. Saturated liquid and vapor thermal conductivity, k_{l} and k_{g}.

Figure 12.10: Representation of a Saturation Table Associated with a Given Material



12.7. Real Gas Property (RGP) File Format

The following provides a description of the real gas property (`.rgp`) file format. The `.rgp` file enables you to input your own specialized real fluid properties into a CFD calculation. The `.rgp` file can be used in conjunction with all models available in CFX.

An `.rgp` file can contain properties for any number of pure component materials. However, the `.rgp` files supplied with CFX only contain one pure component each. Once you load the CCL for a particular equation of state, all materials that use that equation of state are loaded into CFX-Pre.

If you want to maintain your own `.rgp` file that contains multiple pure components, then each new material in the `.rgp` file begins with the `$$$<component>` access key, where `<component>` is the name of the material. Any number of these access keys may exist in the file.

For each fluid associated with an access key, several blocks of information are available, including parameter definition, superheat table, and saturation table information. However, because many calculations do not require all of this information, a less complete set of property information can be included in the `.rgp` file. In this release of CFX, only the dry superheated vapor model is supported, corresponding to `MODEL=3` in the `.rgp` file. Other settings of this parameter are ignored by the CFX-Solver, so all the saturation data that would normally be read when using the non-equilibrium (`MODEL=1`) or equilibrium (`MODEL=2`) models will be ignored. This means that the `$$SAT_TABLE` section does not have to exist under a `$$$Database` access key, that the `SUPERCOOLING` parameter can be zero and that only the `$$SUPER_TABLE` section is necessary.

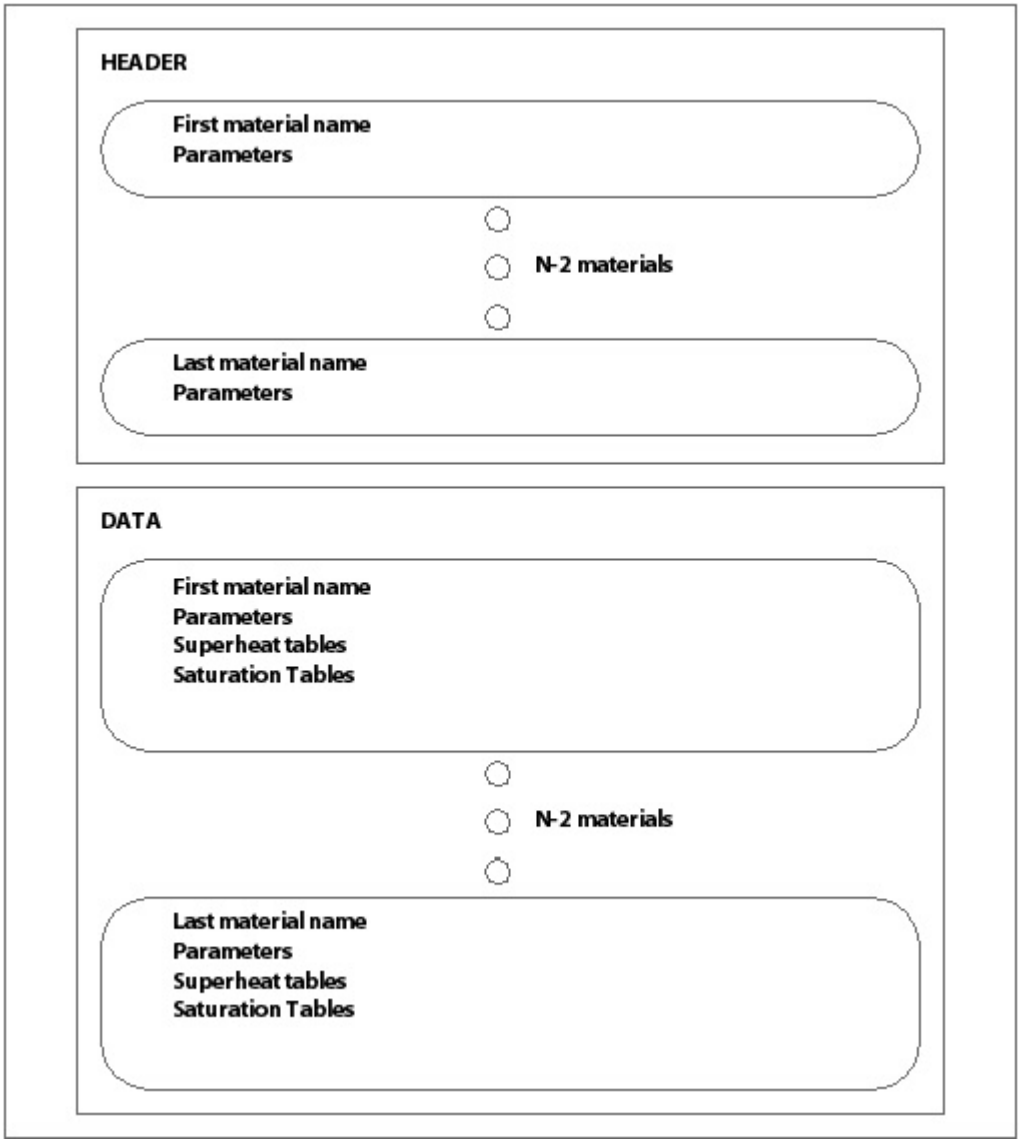
12.7.1. Organization of an .rgp File

The following is a functional example of a `.rgp` file. Note that specific values may be different from those provided by third-party material property databases.

Figure 12.11: Schematic of the .rgp File Contents (p. 511) is a schematic representation of the `.rgp` file organization. The `.rgp` file is divided into HEADER and DATA sections. The first section begins with the line `$$$$HEADER` in the `.rgp` file, and contains all the parameter information for each

material found in the DATA section of the `.rgp` file. The DATA section begins with the line $$$$DATA and contains parameter, superheat and saturation information for a given material. Within both of these major sections, all of the available materials are identified by an access key of the type $$$<component>, where <component> is the name of the material. This is also the name that you will see when selecting a material from CFX-Pre. The purpose of the HEADER section is to enable quick access to material information from CFX-Pre. Information on the order in which the property information is written to the `.rgp` file is available in Real Gas Property (RGP) File Contents (p. 507).

Figure 12.11: Schematic of the .rgp File Contents



12.7.1.1. Detailed .rgp File Format

```
                Type            Description
                ----            -----------
$$$$HEADER
See NOTES 1 and 2
$$$$DATA
$$$<component>   character*8     (key into the .rgp file see NOTE 3)
   nn            integer         (this line is ignored in CFX)
$$PARAM                          See NOTE 4
```

```
   nn            integer         (number of parameters)
DESCRIPTION
aa               character*50    (description of the material)
NAME
aa               character*8     (material name, same as $$$<component>)
INDEX
aa               character*50    (index into clients RGDB program)
MODEL
   nn            integer         (level of property info available 1,2,or 3)
UNITS
   nn            integer         (unit system of 1,2,3,4 or 5)
PMIN_SUPERHEAT                   See NOTE  5
   rr            real
PMAX_SUPERHEAT
   rr            real
TMIN_SUPERHEAT
   rr            real
TMAX_SUPERHEAT
   rr            real
TMIN_SATURATION                  See NOTE 6
   rr            real
TMAX_SATURATION
   rr            real
SUPERCOOLING
   rr            real            (supercooling level in superheat tables)
P_CRITICAL
   rr            real
P_TRIPLE                         See NOTE 7
   rr            real
T_CRITICAL
   rr            real
T_TRIPLE
   rr            real
GAS_CONSTANT
rr               real
TABLE_1                          See NOTE 8
   nt np         integer
TABLE_2
   nt np         integer
TABLE_3
   nt np         integer
TABLE_4
   nt np         integer
TABLE_5
   nt np         integer
TABLE_6
   nt np         integer
TABLE_7
   nt np         integer
TABLE_8                          See NOTE 9
   nt np         integer
TABLE_9
   nt np         integer
SAT_TABLE
npsat nmaxp1 nmaxp2   integer
$$SUPER_TABLE
   nn            integer         (number of superheat tables, nn = 9)
```

(The following is repeated for each superheat table, currently at 9. This section relates to Figure 12.7: Representation of Superheat Tables Associated with a Material (p. 508).)

```
$TABLE_nn
   nt np         integer         (size of superheat arrays, See NOTE 10)
write (iunit,*) (t(i),i=1,nt)              |all following real
write (iunit,*) (p(i),i=1,np)              |
write (iunit,*) ((phi(i,j),i=1,nt),j=1,np) |
write (iunit,*) (tsat(i),i=1,np)           |
write (iunit,*) (phisat(i),i=1,np)         |
```

(This section writes out the saturation information described by Figure 12.10: Representation of a Saturation Table Associated with a Given Material (p. 510))

```
$$SAT_TABLE
npsat nmaxp1 nmaxp2          integer       (size of saturation arrays, see NOTE 11)
do j=1,nmaxp1                    |all following real
write(iunit,*) (prop1(i,j),i=1,npsat)      |
do k=1,2                         |
do j=1,nmaxp2                    |j=1 loop writes sat. liquid data
write(iunit,*) (prop2(i,j,k),i=1,npsat)    |j=2 loop writes sat. vapour data
```

Notes:

1. Information between $$$$HEADER and $$$$DATA is a "stripped down" version of all the property information following the $$$$DATA header. The $$$$DATA section of the .rgp file contains all of the property information relative to the material. This may become a very large file. The $$$$HEADER was added to the top of the .rgp file to enable rapid selection of materials. CFX-Pre reads in and lists this information for you during selection of a material.

2. The $$$$HEADER section contains everything that is in the $$$$DATA section for a given material except data following the $$SUPER_TABLE and $$SAT_TABLE headers.

3. $$$<component> must be an 8 character or less name identifying the material in the .rgp file. For example, for refrigerant R134A, the key may read $$$R134A so that the material can be easily identified in the .rgp file.

4. The $$PARAM parameters may be listed in any order.

5. The parameters PMIN_SUPERHEAT, PMAX_SUPERHEAT, TMIN_SUPERHEAT, and TMAX_SUPERHEAT indicate the limits on the superheat tables. All superheat tables are built as functions of pressure and temperature because most users are able to estimate the pressure and temperature limits in their processes. Supercooling can be easily built into these tables if the equations of state enable it.

6. The parameters TMIN_SATURATION and TMAX_SATURATION indicate the limits for building the saturation tables. The limits should cover the range over which wetness is expected in the solution. The saturation properties should be written to the .rgp file, in equal increments of pressure, starting at TMIN_SATURATION and ending at TMAX_SATURATION.

7. The parameters P_TRIPLE, P_CRITICAL, T_TRIPLE and T_CRITICAL refer to the pressure and temperature triple and critical point values for the substance. The critical pressure and temperature must be supplied to the flow solver and, therefore, must be accurate. If they are not, then the table interpolation will not work correctly.

8. The GAS_CONSTANT for the pure fluid should be supplied and is required if the fluid is to be used in a multi-component fluid calculation.

9. The parameters TABLE_1, TABLE_2, TABLE_3, TABLE_4, TABLE_5, TABLE_6, TABLE_7, TABLE_8 and TABLE_9 refer to the size of the superheat tables for materials found in the $$SUPER_TABLE section. The parameter SAT_TABLE refers to the size of the saturation table for a material found in the $$SAT_TABLE section.

10. The superheat tables 8 and 9 contain information on molecular (dynamic) viscosity and thermal conductivity, respectively. Note that these tables are not required if they are set as constants

or CEL expressions in CFX-Pre. This allows the omission of these tables in the `.rgp` file if superheat information for these transport properties (as functions of pressure and temperature) is unknown.

11. The order in which the superheat tables is written to the `.rgp` file is the same as the order shown in Real Gas Property (RGP) File Contents. For details, see Real Gas Property (RGP) File Contents (p. 507).

12. The values for `nmaxp1` and `nmaxp2` are 4 and 9, respectively. The integer `nmaxp1` refers to the number of single valued properties in the saturation tables, and `nmaxp2` to the number of dual valued properties in the saturation tables. The order that the saturation tables are written to the `.rgp` file is the same as the order shown in the Real Gas Property (RGP) File Contents. For details, see Real Gas Property (RGP) File Contents (p. 507). For the dual valued properties, the saturated liquid properties are written first (`j=1`) and the saturated vapor properties are written last (`j=2`) to the `.rgp` file.

12.7.1.2. Example .rgp File

```
$$$$HEADER
$$$R134A
1
$$PARAM
26
DESCRIPTION
Refrigerant R134a from NIST
NAME
R134A
INDEX
R134A
MODEL
2
UNITS
1
PMIN_SUPERHEAT
55000.00000000000
PMAX_SUPERHEAT
400000.0000000000
TMIN_SUPERHEAT
240.0000000000000
TMAX_SUPERHEAT
400.0000000000000
TMIN_SATURATION
240.0000000000000
TMAX_SATURATION
350.0000000000000
SUPERCOOLING
0.0000000000000000E+00
P_CRITICAL
405928.0E+3
P_TRIPLE
0.0000000
T_CRITICAL
374.21
T_TRIPLE
169.85
TABLE_1
5 5
TABLE_2
5 5
TABLE_3
5 5
TABLE_4
5 5
```

```
TABLE_5
5 5
TABLE_6
5 5
TABLE_7
5 5
TABLE_8
5 5
TABLE_9
5 5
SAT_TABLE
5 4 9
$$$$DATA
$$$R134A
1
$$PARAM
26
DESCRIPTION
 Refrigerant R134a from NIST
NAME
R134A
INDEX
R134A
MODEL
2
UNITS
1
PMIN_SUPERHEAT
55000.00000000000
PMAX_SUPERHEAT
400000.0000000000
TMIN_SUPERHEAT
240.0000000000000
TMAX_SUPERHEAT
400.0000000000000
TMIN_SATURATION
240.0000000000000
TMAX_SATURATION
350.0000000000000
SUPERCOOLING
0.0000000000000000E+00
P_CRITICAL
405928.0E+3
P_TRIPLE
0.00000000
T_CRITICAL
374.21
T_TRIPLE
169.85
TABLE_1
5 5
TABLE_2
5 5
TABLE_3
5 5
TABLE_4
5 5
TABLE_5
5 5
TABLE_6
5 5
TABLE_7
5 5
TABLE_8
5 5
TABLE_9
5 5
SAT_TABLE
5 4 9
$$SUPER_TABLE
9
```

```
$TABLE_1
5 5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   380537.4694438053          411338.2663531548          444977.3409773413
   481319.1776109927          520216.5694145825          389653.7412125674
   409809.2455431806          443799.3654908382          480400.4939131565
   519497.1970340972          397134.4582379403          408220.8885809304
   442595.4896125616          479470.0507244235          518772.4705830857
   402478.8117714382          406565.3815292667          441363.8343756341
   478527.3570462614          518042.2612321422          406650.5418732642
   406650.5418732642          440102.2828142202          477571.8870479560
   517306.4350445317
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   376541.6333194062          389653.7412125674          397134.4582379403
   402478.8117714382          406650.5418732642
$TABLE_2
      5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   146.0231331684759          157.8078814382528          168.5506303571756
   178.5170722927099          187.8737728104760          147.0869026517344
   155.2427165817653          166.8523505651282          177.3642207508735
   187.0763724397798          147.8329014568262          152.5632728477825
   165.1148711540457          176.1977255190386          186.2743889887011
   147.9104560214784          149.7535601390501          163.3352394640403
   175.0170180126285          185.4677211216712          147.6607838596043
   147.6607838596043          161.5101241729302          173.8214905237701
   184.6562641923786
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   144.3034887320350          147.0869026517344          147.8329014568262
   147.9104560214784          147.6607838596043
$TABLE_3
      5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   0.3481914037990427         0.4095738848363432         0.4702915495291079
   0.5305832630019515         0.5905860753062941         0.1402248442644343
   0.1561641066947830         0.1807449135220719         0.2048688905909286
   0.2286916876895624         8.9273098371410883E-02     444320344302E-02
   0.1107157321741076         0.1261131151219874         0.1411960889105597
   6.5633087958005529E-02     6.7151770525718760E-02     7.9166190580894386E-02
   9.0648541948697259E-02     0.1018021936800781         5.1900452073168755E-02
   5.1900452073168755E-02     6.1203439903817498E-02     7.0470885355410795E-02
   7.9394150509035255E-02
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   0.3397008120139503         0.1402248442644343         8.9273098371410883E-02
   6.5633087958005529E-02     5.1900452073168755E-02
$TABLE_4
   5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   644.8541874954598          719.7151797641214          790.5095338665236
   857.2366695715677          919.8970271275263          674.7791960407488
   721.6690087704523          792.3443057727616          858.9628531024553
   921.5203908806476          698.8825060917976          723.6988581975571
   794.2197800381258          860.7115954107751          923.1563593500967
   716.9720727393429          725.8148416296926          796.1389707941393
   862.4838781958006          924.8052629190319          731.8370306406068
   731.8370306406068          798.1052760769886          864.2807537481988
```

```
   926.4674455748727
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   634.2715599484291          674.7791960407488          698.8825060917976
   716.9720727393429          731.8370306406068
$TABLE_5
      5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   733.5432454120784          806.0266427729431          875.3572680256013
   941.1190584653727          1003.111496904734          773.6370689985604
   816.3711389524509          882.8108960718668          946.7561020285661
   1007.512627818587          807.0628915610768          827.9980795292242
   890.7840080089826          952.6232673331618          1012.020341520216
   834.4867573220491          841.2008081815245          899.3408892587429
   958.7368554939568          1016.639020749522          858.8621950799120
   858.8621950799120          908.5575294975291          965.1148207606369
   1021.373299533599
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   723.4007165554241          773.6370689985604          807.0628915610768
   834.4867573220491          858.8621950799120
$TABLE_6
      5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
  -154611.9428101669         -132557.4464307218         -115997.3804840207
  -103111.8528686964         -92801.69633261529         -959671.2544114459
  -873595.5996881851         -764854.3591592711         -680011.5179211511
  -612056.5815781737         -2374639.983863590         -2262401.182121443
  -1982996.713344134         -1763666.875581090         -1587621.548245155
  -4363498.860521937         -4291069.408039705         -3768294.493979145
  -3353439.511168068         -3019297.561697437         -6897297.389986640
  -6897297.389986640         -6117240.959939656         -5448329.203176694
  -4906781.742740840
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
  -158218.3298083634         -959671.2544114459         -2374639.983863590
  -4363498.860521937         -6897297.389986640
$TABLE_7
      5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   1786.030158361605          1904.585719752846          2016.779512855124
   2123.714780772325          2226.113661461630          1748.483241728080
   1823.905112007991          1937.275071424368          2044.975937996867
   2147.900948384664          1740.481415370240          1781.062492767768
   1895.721351626920          2004.229786438356          2107.697911720815
   1735.932309549977          1750.655644805153          1866.736275194941
   1976.098388594144          2080.127578672003          1732.877658407725
   1732.877658407725          1844.072019825660          1954.338672102516
   2058.947921108171
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   1769.188083636742          1748.483241728080          1740.481415370240
   1735.932309549977          1732.877658407725
$TABLE_8
      5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   9.4859076076546339E-06     1.1098072605476093E-05     1.2682727526284134E-05
   1.4225406873221409E-05     1.5718844908709814E-05     1.0113426130489583E-05
   1.1128384712488209E-05     1.2707428862857411E-05     1.4246015436604838E-05
```

```
   1.5736277818145713E-05     1.0622849104772714E-05     1.1161623164463395E-05
   1.2733802630234965E-05     1.4267707066828921E-05     1.5754472822231917E-05
   1.1006164112931968E-05     1.1198285735280963E-05     1.2761997824348757E-05
   1.4290540551654799E-05     1.5773456592676736E-05     1.1321670531250205E-05
   1.1321670531250205E-05     1.2792178337213511E-05     1.4314589371365448E-05
   1.5793272119540815E-05
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   9.2638729380455867E-06     1.0113426130489583E-05     1.0622849104772714E-05
   1.1006164112931968E-05     1.1321670531250205E-05
$TABLE_9
      5      5
   240.0000000000000          280.0000000000000          320.0000000000000
   360.0000000000000          400.0000000000000
   55000.00000000000          141250.0000000000          227500.0000000000
   313750.0000000000          400000.0000000000
   8.7318132315962922E-03     1.2367010749671227E-02     1.5479028969457832E-02
   1.8065003454933646E-02     2.0123880410625419E-02     1.0144646731520802E-02
   1.2375365154798446E-02     1.5478277206731493E-02     1.8061024664563677E-02
   2.0119085407073307E-02     1.1241534315205219E-02     1.2399427073087721E-02
   1.5485543810413213E-02     1.8061449100116474E-02     2.0116675809444373E-02
   1.2033532782386015E-02     1.2438788590707257E-02     1.5500573740125312E-02
   1.8066241345977847E-02     2.0116840814255434E-02     1.2665788438301477E-02
   1.2665788438301477E-02     1.5523214978202122E-02     1.8075268343013946E-02
   2.0119597534766078E-02
   234.5147959861216          254.6474401194407          266.4385386783982
   275.1224493134379          282.1192618430768
   8.1928902502712009E-03     1.0144646731520802E-02     1.1241534315205219E-02
   1.2033532782386015E-02     1.2665788438301477E-02
$$SAT_TABLE
      5      4      9
   72429.02057212396          669962.0863334293          1267495.152094735
   1865028.217856040          2462561.283617346
   239.9999762675480          298.3735220799146          321.5184369582214
   337.4761886205648          349.9999999385916
   0.0000000000000000E+00     0.0000000000000000E+00     0.0000000000000000E+00
   0.0000000000000000E+00     0.0000000000000000E+00
   0.0000000000000000E+00     0.0000000000000000E+00     0.0000000000000000E+00
   0.0000000000000000E+00     0.0000000000000000E+00
   157792.0768617229          234423.4028017849          268281.8836128900
   293391.1384832433          314607.4539245256
   1226.028409346856          1411.621580596332          1538.623093445163
   1688.584812454093          1901.162632743923
   1394.714486693549          1204.132429873541          1105.717252974825
   1023.909256983076          947.2617335591748
   7.3133047353247582E-06     1.5554466347974513E-05     2.3546196865974526E-05
   3.4054173679783401E-05     4.9653987218547186E-05
   836.0313698668695          1119.425611702024          1226.996801517600
   1301.491841955899          1361.442808503099
   1087.583619605917          1180.740355522880          1205.127882782043
   1215.472691163284          1218.190684502114
   0.0000000000000000E+00     0.0000000000000000E+00     0.0000000000000000E+00
   0.0000000000000000E+00     0.0000000000000000E+00
   4.5254108276952397E-04     2.0762809729902491E-04     1.5172962190515123E-04
   1.1920087721741867E-04     9.5575862436921458E-05
   0.1106897637097442         8.1042418175005723E-02     6.9512380598733353E-02
   6.1769627812551062E-02     5.5819411652293066E-02
   380136.7395374610          415734.4668987678          426586.7644804174
   431942.8510520858          434298.6702392859
   736.4416185000921          926.5982109988553          1070.936300636404
   1242.721436965689          1480.769140200031
   3.808368423257648          32.02662176918217          61.85130414931603
   94.90868505077478          132.4954540908795
   5.4108323518481296E-05     5.6676277290207862E-05     6.5936536523112708E-05
   7.9426352920085037E-05     9.9732676665902821E-05
   1762.467546546392          1727.090336441192          1719.363218938642
   1712.044400273131          1703.417712356647
   645.2669936904850          767.7650176481674          824.1454305224304
   868.8335090051181          909.2881576124906
   145.2336985804320          145.9254060428890          140.3837459659026
   134.1949902861215          127.7833214614285
```

```
9.4933329019507708E-06     1.2087890198389884E-05     1.3328835395482605E-05
1.4417814786729988E-05     1.5601940814775012E-05
8.7357640893468182E-03     1.4138793457254478E-02     1.6414352714865677E-02
1.8434500108658915E-02     2.0797428027421110E-02
```

Note:

If you want to use the conditions above, copy the text and paste it into a text editor, taking care to match the formatting exactly, and save it as a `.rgp` file. In CFX-Pre, create a new material that refers to this `.rgp` file. For details, see Table in the *CFX-Pre User's Guide*. Note that the above example, with only a 5x5 table, does not provide sufficient resolution to produce meaningful results.

12.8. Parameters in the .rgp File Controlling the Real Gas Model

The following topics will be discussed:

- REAL_GAS_MODEL (p. 519)
- P_CRITICAL, T_CRITICAL (p. 519)
- UNITS (p. 520)
- MIN_PROPERTY_T, MAX_PROPERTY_T, MIN_PROPERTY_P and MAX_PROPERTY_P (p. 520)
- MOLECULAR_VISCOSITY, MOLECULAR_CONDUCTIVITY (p. 520)
- GAS_CONSTANT (p. 520)
- TMIN_SATURATION, TMAX_SATURATION (p. 520)
- P_TRIPLE, T_TRIPLE (p. 520)
- SUPERCOOLING (p. 521)

12.8.1. REAL_GAS_MODEL

This parameter, normally specified in the CFX-TASCflow PRM file, is ignored by the CFX-Solver because the real gas model in CFX is automatically enabled as soon as you set material properties as functions of pressure and temperature, or select a homogeneous binary mixture as the fluid for your domain.

12.8.2. P_CRITICAL, T_CRITICAL

These two values are required and must be accurate. If they are not, then the table interpolation will not work because the flow solver uses knowledge of the critical point to determine where interpolation is being carried out (above or below the critical point).

12.8.3. UNITS

The equations of state for a real gas are normally formulated in a particular unit system and, therefore, it is necessary to know the unit system being used for the RGP properties so that the CFX-Solver can automatically convert the properties to solution units. Five unit systems are available when using an `.rgp` file as follows:

1. `UNITS=1` (kg, m, s, K)
2. `UNITS=2` (g, cm, s, K)
3. `UNITS=3` (lbm, in, s, R)
4. `UNITS=4` (slugs, ft, s, R)
5. `UNITS=5` (slugs, in, s, R)

If desired, the solver solution units can be set to the same units as the `.rgp` file in CFX-Pre. For details, see Setting the Solution Units in the *CFX-Pre User's Guide*.

12.8.4. MIN_PROPERTY_T, MAX_PROPERTY_T, MIN_PROPERTY_P and MAX_PROPERTY_P

These parameters correspond to the minimum and maximum temperature and pressure limits used for the superheat tables. Pressure and temperature are used to establish the limits because in most instances you will have an idea of the operating range of these quantities in your problem. It is important to make sure that the `MIN_PROPERTY_T` and `MIN_PROPERTY_P` parameters (when combined) do not represent a thermodynamic state below the saturated vapor curve for the substance (that is, represent a subcooled liquid). It is also advisable to determine, prior to running the Real Fluid capability, the expected variation in temperature and pressure for the problem and set the table limit parameters accordingly. This will avoid wasting computational time with an unnecessarily large table.

12.8.5. MOLECULAR_VISCOSITY, MOLECULAR_CONDUCTIVITY

If no superheat tables are given for these two properties, constant values can be specified instead in the parameter section with these two parameters. The CFX-Solver reads and use these values.

12.8.6. GAS_CONSTANT

The material gas constant is used, along with the universal gas constant to compute the molar mass. This is required by the flow solver.

12.8.7. TMIN_SATURATION, TMAX_SATURATION

These are read by the CFX-Solver and can be used for both equilibrium phase change model.

12.8.8. P_TRIPLE, T_TRIPLE

These parameters are not yet used by the flow solver.

12.8.9. SUPERCOOLING

This parameter is not used by the flow solver.

Chapter 13: Operating Maps and Operating Point Cases

You can use CFX to create *operating maps* (in some cases also known as *performance maps*), which show how simulation results vary against selected parameters. For example, you can create an operating map for a compressor with speed contours plotted for varying total pressure ratio and inlet mass flow rate.

Any particular set of parameter values defines an *operating point*. After solving a set of operating points, CFX constructs an operating map using output from the results.

The following topics are discussed:

13.1. Defining an Operating Point Case

13.2. Limitations

13.3. Running an Operating Point Case

13.4. Post-processing an Operating Point Case

13.1. Defining an Operating Point Case

An operating point case includes the specification of a set of operating points. This specification is held by an `Operating Points` object, which appears in CFX-Pre in the **Outline** tree view under **Simulation Control**.

You can set up an operating point case in CFX-Pre as follows:

1. Create an operating point input parameter table with each row corresponding to an operating point. One column acts as a row index, holding a different integer value for each row. The remaining columns hold input parameter data. Note that the table file format is similar to the CFX profile data file format (see Profile Data Format in the *CFX-Pre User's Guide*).

   You can create/edit the table in CFX-Pre, or using a text editor.

   An example table file is shown below:

```
[Name]
Compressor Operating Points

[Index Fields]
Index

[Data]
Index, massFlow [kg s^-1], speed [rev min^-1]
1, 1.23000002e+00, 1.45000000e+03
2, 1.24000001e+00, 1.55000000e+03
3, 1.25000000e+00, 1.65000000e+03
4, 1.27999997e+00, 1.78000000e+03
```

```
            #-- End of profile Input Table--
```

For details on table editing, see Establishing Table Data and Creating Table Functions in the *CFX-Pre User's Guide*.

2. Load the table, either from the table editor or by selecting **Tools** > **Initialize Profile Data** and selecting the table data file.

   CFX-Pre automatically creates a user function for retrieving table data. The user function is listed in the **Outline** tree view under **Expressions, Functions and Variables** > **User Functions**. For details, see Table User Functions in the *CFX-Pre User's Guide*.

3. Define at least one solution monitor to represent the result for an operating point as a single value.

   For example, you could create a monitor for total pressure ratio. For details, see Working with Monitors in the *CFX-Pre User's Guide*.

   For transient simulations, you can use a statistical quantity to represent the result of an operating point simulation. For details, see [Monitor Name]: Monitor Statistics in the *CFX-Pre User's Guide*.

   The monitors that you define will become output parameter values in the resulting operating point parameter table, which is viewable in CFD-Post and can be exported. The operating point parameter table is essentially the operating point input parameter table (defined in CFX-Pre) with added columns showing output parameter values (each output parameter corresponding to a monitor defined in CFX-Pre).

4. Define an `Operating Points` object that refers to the operating point input parameter table.

   The `Operating Points` object, once defined, is listed in the **Outline** tree view under `Simulation Control`.

   Within the `Operating Points` object, you can optionally define one or more operating maps by specifying a pair of independent variables for each map. The available independent variables are: operating point input parameters provided in the table data; defined solution monitors (see Monitor Tab in the *CFX-Pre User's Guide*). For details, see Operating Points in the *CFX-Pre User's Guide*. Note that you can also define operating maps in CFD-Post. Nevertheless, the `Operating Points` object is required for operating point cases.

5. Optionally define expressions to be used for input parameters, making use of the aforementioned user function as needed to retrieve table data.

   For example, you could define an expression as `Compressor Operating Points.massFlowOut(Index)` so that it accepts a table row index value and returns the corresponding value from the column for mass flow rate.

6. Develop an operating point simulation, specifying physics conditions using the table function.

   For example, you could specify the mass flow rate for an inlet boundary using an expression (whether defined earlier or typed in-line) that retrieves values via the table function.

13.2. Limitations

- Operating point cases are not supported in Ansys Workbench.
- Operating points are not supported in a System Coupling analysis.
- Modifying the internally stored command language or table data in an operating point results file (`mres`) can lead to:
  - Unexpected results during postprocessing
  - Failure of the CFX-Solver to detect modified operating points during a restart.
- CFD-Post cannot process an operating point case that has different meshes between any two operating points. Although operating points can have mesh changes, for example due to mesh deformation or the use of multiple configurations, CFD-Post requires that all operating points in an operating point case have corresponding mesh changes.
- User locations that are created directly in CFX-Pre (for example, a user line with the `Two Points` option) cannot be changed between operating points.
- For Transient Blade Row simulations, CFX-Pre must be able to evaluate the Transient Method "Timestep" value when the Solver Input File is written; this value cannot change between operating points. This implies that the Time Period cannot be specified using an expression that depends upon any operating point table functions.

13.3. Running an Operating Point Case

An operating point case definition can be held by either:

- An `mdef` file and its supporting files, such as a table file (defining operating points)
- An `mres` file and its supporting files, such as:
  - A table file (defining operating points)
  - Results files for partially/fully completed operating point jobs (each job simulates a single operating point).

Each operating point job in an operating point run is initialized in the same way: according to the initialization settings. For details, see Initialization in the *CFX-Pre User's Guide* and Files Used by the CFX-Solver in the *CFX-Solver Manager User's Guide*.

An operating point run that is defined by an `mdef` file can be continued from the `mres` file of a previous (completed or interrupted) operating point run, retaining some or all results of previously run operating point jobs.

When continuing an operating point run, the settings of the **Operating Points** tab (found in CFX-Pre and in CFX-Solver Manager) control the handling of any existing results from previously run operating point jobs. The settings of the **Operating Points** tab in CFX-Pre are saved with the case and, when the case is used as the Solver Input File, are used to initialize the corresponding settings in CFX-Solver

Manager. For details, see Operating Points Tab in the *CFX-Pre User's Guide* and Operating Points Tab in the *CFX-Solver Manager User's Guide*.

While an operating point case runs, you can monitor the status of each operating point job in CFX-Solver Manager on the operating point run history page. For details, see Operating Point Run History Page in the *CFX-Solver Manager User's Guide*.

Note:

Each `mres` file has a corresponding directory that contains operating point job results. You can move the subdirectory and `mres` file together, but you must not rename the `mres` file or change the subdirectory's name or contents.

13.4. Post-processing an Operating Point Case

When you postprocess an operating point case in CFD-Post:

- The `Cases` and `Report` objects each list the defined operating maps for each loaded operating point case. The `Cases` object also lists, for each operating map, the data series objects defined in the operating map. The `Report` object also lists, for each loaded operating point case, the operating point parameter table, which is essentially the operating point input parameter table (shown in CFX-Pre) with added columns showing output parameter values.
- The **Operating Points Viewer** is available to show, for a given case, the operating point parameter table and an operating map based on that table. For details, see CFD-Post Operating Points Viewer in the *CFD-Post User's Guide*.

  As for other charts, you can show an operating map in a separate window by right-clicking the operating map in the **Outline** tree view and selecting **Show in Separate Window**.
- To switch to a different operating map, right-click an operating map object in the **Outline** tree view and select **Edit**.
- You can create new operating maps by:

  1. Do either of the following:

     – Right-click **Cases** > **[case name]** > **Operating Maps** and select **Insert** > **Operating Map Chart**.

     – Right-click **Report** > **Operating Points** > **[case name]** and select **Insert** > **Operating Map Chart**.

     The details view for the operating map appears.
  2. Configure the settings as appropriate.

     For details, see Operating Maps in the *CFD-Post User's Guide*.
  3. Click **Apply**.

- You can check the quality of the response surface as follows:

1. Right-click **Cases** > **[case name]** and select **Edit**.

   The details view for the case appears.

2. In the details view, on the `Operating Points` tab, observe the tabulated quality metrics for all of the generated response surfaces. Each row of the table corresponds to one response surface for one output variable. The quality metrics table gains a new row every time a new response surface is generated.

   Note that a response surface for a given output variable is generated only when you make an operating map that uses response points for its data source and that plots the output variable or uses the output variable as a chart axis.

The same table of quality metrics appears in the `Report` object and the **Report Viewer**.

The quality metrics are the same as those used in Ansys DesignXplorer. For details, see Goodness of Fit for Output Parameters in a Response Surface in the *DesignXplorer User's Guide*.

Chapter 14: Coupling CFX to an External Solver

There are several main approaches to coupling CFX to an External Solver:

14.1. Coupling CFX to an External Solver: System Coupling Simulations

14.2. Coupling CFX to an External Solver: Functional Mock-up Interface (FMI) Co-simulation

14.3. Coupling CFX to an External Solver: GT-SUITE Coupling Simulations

14.1. Coupling CFX to an External Solver: System Coupling Simulations

You can use Ansys System Coupling to perform coupled simulations that involve multiple physics solvers, coupling active co-simulation participants and/or importing static data from an external data source. For example, you can run Ansys Mechanical and Ansys CFX in a single analysis and/or import data from an output file into an Ansys CFX analysis. Once the physics and coupling setups are complete, the coupled analysis is executed and managed by System Coupling.

System Coupling can be used in the following contexts:

- **System Coupling's user interfaces**:

  System Coupling is run from its *graphical user interface (GUI)* or its *command-line interface (CLI)*.

  System Coupling's interfaces provide enhanced control over coupled simulation processes, including automatic starts and restarts for participants, the ability to manipulate System Coupling's data model, and an interactive solution workflow.

  In these contexts, you still set up participant physics in the participant's user interface, but you'll perform the coupled analysis—starting System Coupling, loading participants, specifying values for coupling-related analysis settings, and automatically starting participants—using System Coupling's GUI or CLI. Alternatively, if a coupled analysis setup was exported from Workbench, you can open it and execute it in the GUI or CLI.

  An extensive list of coupling participants is given in Supported Coupling Participants.

  For more information, see Using System Coupling's User Interfaces.

- **System Coupling in Workbench**:

  System Coupling is run from the Workbench interface.

  In this context, you'll connect the **Setup** cell from a Fluid Flow (CFX) analysis system to the **Setup** cell for the System Coupling component system, signaling that the CFX solver will act as a co-simulation participant in a coupled analysis. Most of the coupling-related analysis settings are made using the System Coupling system's **Setup** cell.

Once the physics and coupling setups are completed in Workbench, you can either execute the coupled analysis in Workbench, or export the setup for execution in one of System Coupling's user interfaces.

Participants that can be coupled with CFX include:

– Static Structural

– Transient Structural

– Transient Thermal

– Steady-State Thermal

An extensive list of coupling participants is given in Supported Coupling Participants for System Coupling in Workbench.

For more information, see Using System Coupling in Workbench in the *System Coupling User's Guide*.

Additional information can be found in the following sections:

14.1.1. Supported Capabilities and Limitations
14.1.2. Variables Available for System Coupling
14.1.3. System Coupling Related Settings in CFX
14.1.4. Restarting CFX Analyses as Part of System Coupling
14.1.5. Initializing a Coupled CFX Analysis from an Independent CFX Analysis
14.1.6. Running CFX as a Participant from System Coupling's GUI or CLI
14.1.7. Product Licensing Considerations when using System Coupling

14.1.1. Supported Capabilities and Limitations

Ansys CFX supports the following capabilities when used in a System Coupling analysis:

- Force and displacement data transfers on wall boundaries:
  - Input of incremental displacements data from System Coupling allows moving and deforming mesh specification on boundary wall regions. Note that the option for **Mesh Motion** on the corresponding wall boundaries needs to be **System Coupling**.
  - Output of force data on boundary wall regions. CFX is able to serve the total force variable (viscous and normal) on all wall boundaries (with or without moving/deforming boundaries). See Variables Available for System Coupling (p. 532) for details about the forces transferred from the wall boundaries.
- Thermal data transfers on wall boundaries:
  - Input of temperature data from System Coupling on all wall boundaries.
  - Output of EITHER

    → heat flow, OR

→ (for cases not involving radiation) heat transfer coefficient reference temperature plus heat transfer coefficient

to System Coupling on all wall boundaries.

- Full support for local and distributed parallel solver execution.
- The convergence data for the continuity and momentum equations in CFX is shared with System Coupling at run time.

  If using System Coupling in Workbench, the CFX equation residuals of a System Coupling simulation can be viewed using the **Solution** cell of the System Coupling component system. The CFX equation residuals can also be viewed from the **Solution** cell of the CFX system.

Note the following limitations when using CFX in a System Coupling analysis:

- For data transfers, only SI solution units and the global Cartesian coordinate frame are supported. CFX does not convert values if the data received uses a different unit system or any local coordinate frames.
- Monitor point data is not shared with System Coupling but can be seen from CFX-Solver Manager.
- Data transfer regions can only be on wall boundaries that are not on solid or porous domains solved by CFX.
- Multi-configuration cases are not supported, including cases that involve remeshing during an analysis.
- Operating Point cases are not supported.
- A CFX simulation that models radiation can only send heat flow data; it cannot send heat transfer coefficient reference temperature and heat transfer coefficient data.
- An immersed boundary cannot be used as a System Coupling interface boundary.
- The rigid body solver is not supported.
- When using thin surfaces/baffles in CFX as System Coupling interfaces, you need to create a separate wall boundary condition for each side of the thin surface. These are paired with two sets of shell elements in Ansys Mechanical. For more information, see Coupling Thin Surfaces in CFX in the *Mechanical User's Guide*.
- Particle forces do not contribute to the force passed to System Coupling on a wall boundary.
- The CFX-Solver log is not viewable from System Coupling.
- In Workbench, CFX-Pre-specific errors do not cause a message box to appear in Workbench or in CFX-Pre. However, such errors are reported in the CFX system's **Setup** cell properties when the cell is marked "Needs input".
- If the maximum elapsed time (wall clock time) is specified (from CFX), it will be ignored.

- You cannot interrupt a run from the CFX. However, you can from any of System Coupling's contexts.
- You cannot start a new System Coupling simulation if the CFX portion of the simulation is initialized with results from a transient, non-System Coupling, CFX case involving a Theta shift (for example, a transient rotor stator case). It is a requirement that new cases are initialized without the **Continue History From** option; however, for CFX cases involving a Theta shift, the interpolation process does not reset the simulation time, which would be required to begin a new case.
- For one-way System Coupling where CFX sends heat flow or heat transfer coefficient reference temperature plus heat transfer coefficient, the temperature on the FSI interface might not be set properly.
- The bulk heat transfer coefficient reference temperature and heat transfer coefficient data generated for a System Coupling interface in a multiphase case are not available for callbacks or User Fortran; this data is only used by System Coupling.
- Coupling jobs submitted to RSM are run in the foreground, so the CFX **Solution** cell's **Update Option** property must be set to **Run in Foreground**.

14.1.2. Variables Available for System Coupling

The following variables are available on all boundary wall regions.

Table 14.1: Variables On Boundary Wall Regions

Display Name / Internal Name	Transfer Direction	Tensor Type	Extensive/Intensive	Quantity Type
force / `force`	Output	VectorXYZ*	Extensive	Force
displacement / `displacement`	Input	VectorXYZ*	Intensive	Length
temperature / `temperature`	Input	Scalar	Intensive	Temperature
heat flow / `heatflow`	Output	Scalar	Extensive	Heat Rate
heat transfer coefficient / `heat-transfer-coefficient`	Output	Scalar	Intensive	Heat Transfer Coefficient
HTC Reference Temperature / `HTC Reference Temperature`	Output	Scalar	Intensive	Temperature

For information on the data transfers supported by System Coupling for each participant in each coupling context, see the tables in System Coupling Capabilities in the *System Coupling User's Guide*.

14.1.2.1. Wall Force Data transferred from CFX System to System Coupling System

Force transferred from a CFX wall boundary in a System Coupling case is, by default, based on the solved pressure, which is relative to the specified reference pressure. You can change the force calculation to be based on absolute pressure by setting the expert parameter `include pref in forces = t`.

Viscous forces are included in the force transferred.

14.1.2.2. Displacement transferred from System Coupling System to CFX System

The following applies in a coupled analysis where displacement is received by CFX.

- The displacement data received by CFX represents the incremental displacement for the current time step (the incremental displacement since the end of the previous time step).
- When the System Coupling analysis type is "general" or "steady" (as opposed to "transient"), and when CFX solves before or simultaneously to the solver sending the displacement (such as Mechanical), then during the first coupling iteration of each coupling step the displacement received is 0 [m].

14.1.2.3. Thermal Data exchange between CFX and System Coupling

CFX provides EITHER

- heat flow, OR
- (for cases not involving radiation) heat transfer coefficient reference temperature plus heat transfer coefficient to System Coupling.

CFX receives wall temperature from System Coupling.

For multiphase flows:

- If CFX is configured to send heat flow to System Coupling, the heat flow is a bulk quantity.

  If instead, CFX is configured to send the heat transfer coefficient reference temperature plus heat transfer coefficient to System Coupling, the heat transfer coefficient reference temperature and heat transfer coefficient are bulk quantities.
- The received temperature is applied as a bulk temperature.

14.1.3. System Coupling Related Settings in CFX

- When using System Coupling in Workbench, you should set the **Solution** cell property "Keep Latest Solution Data Only" to "False" (which is not the default) to retain old solution data. "Keep Latest Solution Data Only" is described in Properties Pane in the *CFX Introduction*.
- Run Calculation
  - When running as part of a transient coupled analysis, the step size for the analysis and duration of the analysis are controlled by System Coupling.
    - → The time step size(s) specified in the CFX setup is ignored. The CFX solution will be advanced using the time step size specified as part of the CFX setup.
    - → The number of time steps specified in the CFX setup is also ignored.
  - The specified **Maximum Number of Iterations** corresponds to the maximum number of nonlinear solver iterations performed per coupling iteration.
- Wall boundary participating in CFX

  - The **System Coupling** option must be selected for the **Mesh Motion**.

- Heat Transfer

  - Heat transfer is only available on walls. It is not available on inlets/outlets/openings.

  - The heat transfer option for a wall boundary must be set to `System Coupling` (**Boundary Details** > **Heat Transfer** > **Option** set to `System Coupling`) in order to obtain wall temperature from other participants taking part in the coupled analysis.

  - A proper value for the htc reference temperature promotes fast solution convergence.

    By default, the htc reference temperature is set to the near wall temperature. This default behavior is usually suitable when the y+ value (near wall element thickness) is small, particularly when there is also a large variation in wall temperature.

    In cases where the y+ value is large, the reference temperature might not be accurate, in which case you can override the default behavior by manually specifying a value for the htc reference temperature.

    To do this, select **Expert Parameters** > **Physical Models** > **Miscellaneous** > **tbulk for htc** and then enter a (constant) temperature value.

    A specified near wall reference temperature applies to the entire wall. If there is a large temperature variation over the wall, you should specify an average expected wall temperature.

- Execution Control

  - System Coupling input files (`.sci`) or System Coupling Participant files (`.scp`) are always written if there are boundary conditions set to use the `System Coupling` option. Additionally, you can request that an `.scp` file is written regardless of the boundary condition options, by selecting **Always write System Coupling Input File**. The System Coupling Participant file is written alongside the CFX-Solver input (`.def`) file and has the same name except for the file extension (`.scp`).

14.1.4. Restarting CFX Analyses as Part of System Coupling

Go to the section Stops and Restarts of System Coupling Analyses for general information on restarting a coupled analysis and links to context-specific information.

- For **System Coupling in Workbench**: Restarting a Coupled Analysis in Workbench.

- For **System Coupling's GUI or CLI**: Restarting a Coupled Analysis.

To restart your coupled analysis, you will also need restart information specific to the participants connected to System Coupling.

For details on other participant systems connected to System Coupling, see the following lists of supported systems and references to their corresponding documentation regarding restarts:

- Supported Coupling Participants for System Coupling in Workbench

- Supported Coupling Participants

14.1.4.1. Generating CFX Restart Files
14.1.4.2. Specify a Restart Point in CFX
14.1.4.3. Making Changes in CFX Before Restarting
14.1.4.4. Recovering the CFX Restart Point after a Workbench Crash
14.1.4.5. Restarting from the Beginning of the Coupled Analysis in Workbench

14.1.4.1. Generating CFX Restart Files

A restart of a System Coupling analysis requires corresponding restart points to exist in the coupling service and in each of the solvers participating in the analysis.

CFX will automatically generate the corresponding backup (`.bak`) files based upon requests received from System Coupling. Note that these files are generated in addition to any other manually requested backup files (or results files) and are retained even if the **Backup Data Retention** option (Option in the *CFX-Pre User's Guide*) is set to `Delete Old Files`.

14.1.4.2. Specify a Restart Point in CFX

When the CFX solution is restarted, the default behavior is that the results file (.res) corresponding to the current restart point will be used. To specify a different restart point:

1. In **CFX-Solver Manager**, on the **Define Run** dialog box, select the **Initial Values** tab. For the **Initialization Option**, select **Initial Conditions**.
2. Enable the **Initial Values Specification**.
3. Next to **File Name**, select the browse icon ().
4. Browse to the backup files (`.bak`) or previous results file (`.res`), and select the file that matches the desired restart point. Backup files from a v1.0 coupled analysis in Workbench will have "SC_ess" in their name. Backup files from a v2.0 command line coupled analysis will have "SC_CP_ess" in their name.

   Make sure that you select the restart point that corresponds to the restart points selected for tSystem Coupling and the other coupling participant.

   If the case that you are restarting was initialized from an independent CFX analysis, the time listed on the backup file (`.bak`) or results file (`.res`) might not match the corresponding restart points in System Coupling and the other participant system (such as Mechanical). Use the CFX-Solver Output file to determine which time steps correspond to the System Coupling restart points.
5. Click **Open** to read-in the data for the selected restart point.
6. **Continue History From** can be enabled or disabled. Click **Save Settings**.
7. Close CFX by selecting **File** > **Close CFX-Solver Manager**.

14.1.4.3. Making Changes in CFX Before Restarting

1. Make the needed setup changes in CFX-Pre.
2. Close CFX-Pre by selecting **File** > **Close CFX-Pre**.

14.1.4.4. Recovering the CFX Restart Point after a Workbench Crash

1. In **CFX-Solver Manager**, on the **Define Run** dialog box, select the **Initial Values** tab. For the **Initialization Option**, select **Initial Conditions**.
2. Enable the **Initial Values Specification**.
3. Next to **File Name**, select the browse icon ().
4. Browse to the backup files (`.bak`) or previous results file (`.res`).
5. If Workbench crashed while writing the final file, that file will be corrupt. In this case, select any of the previous files. Backup files from a v1.0 coupled analysis in Workbench will have "SC_ess" in their name. Backup files from a v2.0 command line coupled analysis will have "SC_CP_ess" in their name.

   Make sure that you select the restart point that corresponds to the restart points selected for System Coupling and the other coupling participant.
6. Click **Open** to read-in the data for the selected restart point.
7. **Continue History From** can be checked or unchecked. Click **Save Settings**.
8. Close CFX by selecting **File** > **Close CFX-Solver Manager**.

14.1.4.5. Restarting from the Beginning of the Coupled Analysis in Workbench

1. To restart the solution from the first time step, in the **Project Schematic** right-click System Coupling's **Solution** cell and select **Clear Generated Data**.
2. Any initial values in the CFX system from a results file or backup file will continue to exist and be used in the CFX setup. You need to clear that data manually.
3. Restart the run from the System Coupling system.

14.1.5. Initializing a Coupled CFX Analysis from an Independent CFX Analysis

The results from an independent CFX analysis (transient or steady-state) can be used to initialize the CFX analysis in a coupled analysis.

1. In **CFX-Solver Manager**, on the **Define Run** dialog box, select the **Initial Values** tab. For the **Initialization Option**, select **Initial Conditions**.
2. Enable the **Initial Values Specification**.

3. Next to **File Name**, select the browse icon ().

4. Select the results file from the CFX analysis that you are using to initialize the coupled analysis (this may be a steady-state or transient file).

5. If initializing from a transient analysis, ensure that **Continue History From** is not selected.

   If initializing from a steady-state analysis, **Continue History From** can be selected or not selected.

Note:

In Workbench, do not connect the **Solution** cells of the independent transient CFX analysis and the coupled CFX analysis. This connection causes the **Continue History From** setting to be selected, and does not allow you the control to change this selection.

14.1.6. Running CFX as a Participant from System Coupling's GUI or CLI

System Coupling analyses can be run using either of System Coupling's user interfaces. To run Ansys CFX as a coupling participant:

- **For coupled analyses set up in a System Coupling interface**:
  - To set up a coupled analysis in System Coupling's GUI or CLI, perform the steps outlined in Preparing for a Coupled Analysis, Creating a Coupled Analysis, and Modifying Coupled Analysis Settings.
  - To run the coupled analysis in System Coupling's GUI or CLI, perform the steps outlined in Running a Coupled Analysis.
- **For coupled analyses set up using System Coupling in Workbench**:

  System Coupling analyses that are set up in Workbench can be exported and executed using one of System Coupling's user interfaces.
  - To set up a coupled analysis in Workbench, follow the steps outlined in Setting Up a Coupled Analysis in Workbench.
  - To export a coupled analysis setup, follow the steps outlined in Exporting a System Coupling Setup.
  - To execute the exported setup in System Coupling's GUI or CLI, follow the steps outlined in Running an Exported System Coupling Setup.

For information on how to restart an analysis in one of System Coupling's user interfaces, see Restarting a Coupled Analysis.

For details on alternate workflows and capabilities, see Common Coupled Analysis Tasks and Advanced Coupled Analysis Tasks.

14.1.7. Product Licensing Considerations when using System Coupling

A distinct license is required for each coupling participant product, but no additional licenses are required for the System Coupling infrastructure itself. For more information on applicable licenses, see Coupled Analysis Licensing Requirements in the *System Coupling User's Guide*.

14.2. Coupling CFX to an External Solver: Functional Mock-up Interface (FMI) Co-simulation

Ansys CFX can run co-simulations using Functional Mock-up Interface (FMI) technology (http://fmi-standard.org/). In a co-simulation, the CFX-Solver interacts with a solver executable contained in a Functional Mock-up Unit (FMU) file (`*.fmu`). The CFX-Solver has two-way communication with the FMU solver via a user function of type `Functional Mockup Unit Instance`. This user function provides CEL access to:

- FMU input variables
- FMU input parameters
- FMU output variables

To set up a co-simulation using Functional Mock-up Interface (FMI) technology:

- Obtain an FMU file from third party software. The executable contained in the FMU file should be compatible with (compiled for) the operating system that will run the co-simulation.
- With your case open in CFX-Pre, initialize the FMU file as described in Initialize FMU Instance in the *CFX-Pre User's Guide*. When the FMU file is initialized, a user function is created in the **Outline** tree view under **Simulation** > **Expressions, Functions and Variables** > **User Functions** with the `Functional Mockup Unit Instance` option selected.
- Complete the function definition as described in Functional Mockup Unit Instance in the *CFX-Pre User's Guide*.

  In order to complete the function definition, you might need to create CEL expressions that can be used to supply values for FMU input variables and FMU input parameters.
- Make use of the FMU output variables in your CFX case. For example, use an FMU output variable to control a conditional domain interface or an inlet temperature.
- Recommended: Define expression monitors for the FMU input variables and output variables.

The following topics are discussed:

14.2.1. Limitations of using CFX with FMI

14.2.2. Units of FMU Output Variables

14.2.1. Limitations of using CFX with FMI

- The CFX implementation only supports "co-simulation mode" using Functional Mockup Interface 2.0, not "model exchange".

- Only transient analysis is supported. Steady-state analysis is not supported. Time must be the independent variable in the Functional Mockup Units.
- If an FMU does not specify the units, you will need to ensure that the correct values are passed in the arguments. Simply divide the input by the expected quantity dimension. For details, see Units of FMU Output Variables (p. 539).
- Both integer and logical parameters are represented by real values inside the CFX-Solver, and converted during communication with FMUs. CFX implementation does not support FMU parameters of strings or enumeration types.
- You cannot specify a value for an FMU input parameter using an expression that resolves to different values. You can specify a value for an FMU input parameter using a numerical value or an expression that resolves to a constant value.
- Special ASCII characters are supported for FMU scalar variables via the **Value Name** setting (see Functional Mockup Unit Instance in the *CFX-Pre User's Guide*); CFX Object names are altered to avoid such characters. Non-ASCII Unicode characters are not supported.
- Clean solution restarts are not possible if the FMU cannot serialize the state (canSerializeFMUState is equal to "false" or is not set in the `.xml` file within the `.fmu` file).
- If you specify more than one FMU, those FMUs are assumed to be order-independent. CFX provides no control of the order of execution.

14.2.2. Units of FMU Output Variables

CFX expressions and the variables they reference have an internal quantity type, but the returned output has whatever units are required. Applications generating FMUs should, but do not necessarily, specify the quantity type and units. If the expected units are not specified, the input and output unit types will be dimensionless. You must ensure that the FMU inputs have the units required by the FMU and that the FMU outputs have units appropriate for CFX. You can do this by dividing by a constant that has the intended units. Here are some examples:

- To convert a force to Newtons, divide by 1 Newton:

  ```
  xForceN = force_x()@wall / 1 [N]
  ```

- To convert a pressure (force per unit area) into head (height of a column of water):

  ```
  hydrodynamicHead = areaAve(Pressure)@outlet / (1000 [kg/m^3] * g)
  ```

  or

  ```
  hydrodynamicHead = areaAve(Pressure)@outlet / 1 [m water]
  ```

  In CFX, `1 [water] = 1 [Wg] = 9806.38 [kg/(m^2 s^2)] = 9806.38 [Pa/m]`.

14.3. Coupling CFX to an External Solver: GT-SUITE Coupling Simulations

CFX can run simulations coupled with GT-SUITE. For details on GT-SUITE, see https://www.gtisoft.com.

Note:

GT-SUITE `.gtm` files should not be confused with GTM files that contain mesh data.

The following topics are discussed:

14.3.1. Supported Capabilities and Limitations
14.3.2. GT-SUITE Coupling Related Settings in CFX
14.3.3. Setting up a CFX Simulation to use GT-SUITE
14.3.4. Restarting CFX Analyses as Part of GT-SUITE Coupling

14.3.1. Supported Capabilities and Limitations

- CFX can only couple to one GT-SUITE model in a single simulation or configuration.
- CFX only supports the interface type of 1 (vel-flow) for all coupling patches in the corresponding GT-SUITE model.
- CFX does not support multiphase flows for simulations with GT-SUITE coupling.
- CFX does not support **Edit Run in Progress** for GT-SUITE coupling. The solver will only print a warning message to the CFX-Solver Output file and ignore the modified command file.
- There is no support for running a GT-SUITE coupling case with CFX running on one platform and GT-SUITE running on another platform. You can, however, set up a GT-SUITE coupling case on one platform then run it on another platform.
- CFX has been tested with coupling to GT-SUITE v2019 Build 3 and v2020 Build 1. Other versions might work but have not been certified.

14.3.2. GT-SUITE Coupling Related Settings in CFX

- The **Tools** > **Initialize GT-SUITE Coupling** command and the **Initialize GT-SUITE Coupling** dialog box.

  For details, see Initialize GT-SUITE Coupling in the *CFX-Pre User's Guide*.

- The `GT-SUITE Models` object in the **Outline** tree view.

  For details see GT-SUITE Model Object in the *CFX-Pre User's Guide*.

- Functions of type `GT-SUITE Patch`.

  For details, see GT-SUITE Patch Functions in the *CFX-Pre User's Guide*.

- Domain settings to receive shaft speed data from GT-SUITE.

For details, see GT-SUITE Domain Models in the *CFX-Pre User's Guide*.

- Boundary condition settings for quantities to be supplied by GT-SUITE.

  For details, see Use GT-SUITE Coupling in the *CFX-Pre User's Guide*.

- The **GT-SUITE Initialization** > **Selectable Versions** preference, which allows you to add new GT-SUITE versions to the list of available versions for some settings.

  For details, see User Interface in the *CFX-Pre User's Guide*.

14.3.3. Setting up a CFX Simulation to use GT-SUITE

1. Set up a GT model using GT-SUITE and produce a `.gtm` file.

   Note that CFD Coupling interfaces must be created in GT-SUITE. The interface type must be set to 1 (velocity-flow type).

2. Create a domain in CFX.

   This requires importing a mesh or loading an existing case beforehand.

3. Set the analysis type to be transient.

4. Add material species in CFX that are consistent with the material species in the GT model.

   You can do this manually in CFX-Pre, or you can import GT-SUITE species from a material library CCL file later when you initialize the GT-SUITE model in CFX-Pre.

   Note:

   A material library CCL file might be obtainable from Gamma Technologies (gtisoft.com).

5. Ensure that domain initialization is consistent with the GT-SUITE model.

6. Initialize the GT-SUITE model, creating a new model object or re-initializing an existing one.

   For details, see Initializing or Re-initializing a GT-SUITE Model (p. 542).

   If the setup changes later, you can re-initialize the model, or, to adjust the GT-SUITE version number, installation path, or material association, you can edit the `GT-SUITE Model` object. For details, see Editing a GT-SUITE Model (p. 544). You can also review and edit the GT-SUITE Patch functions if needed. For details, see GT-SUITE Patch Functions in the *CFX-Pre User's Guide*. You can also review and edit the GT-SUITE Turboshaft functions if needed. For details, see GT-SUITE Turboshaft Functions in the *CFX-Pre User's Guide*.

7. If, in the **Initialize GT-SUITE Coupling** dialog box, on the **Boundary Association** tab, you did not choose the `Specify Boundary Names` option and select **Modify boundaries to use GT-SUITE Coupling**, you should modify CFX boundaries to use GT-SUITE Patch functions as appropriate.

   For details, see Use GT-SUITE Coupling in the *CFX-Pre User's Guide*.

8. If, in the **Initialize GT-SUITE Coupling** dialog box, on the **Turboshaft Setup** tab, you did not select **Modify domain motion to use GT-SUITE Coupling**, you should modify CFX domains to use GT-SUITE Turboshaft functions as appropriate.

   For details, see GT-SUITE Domain Models in the *CFX-Pre User's Guide*.

9. If, in the **Initialize GT-SUITE Coupling** dialog box, on the **Monitoring** tab, you did not specify all of the desired solution monitors, note that you can add monitors manually.

   For details, see Working with Monitors in the *CFX-Pre User's Guide*.

14.3.3.1. Initializing or Re-initializing a GT-SUITE Model

Before initializing a new GT-SUITE model, consider creating the CFX domains. With domains already created, you can create domain boundaries that use GT-SUITE, or modify existing domain boundaries to use GT-SUITE, as part of the GT-SUITE Coupling initialization process. Without any domains, however, you can still initialize the model and then later set up the boundaries, materials, and functions.

1. Select **Tools** > **Initialize GT-SUITE Coupling**.

   The Initialize GT-SUITE Coupling dialog box appears. For details, see Initialize GT-SUITE Coupling in the *CFX-Pre User's Guide*.

2. At the top of the dialog box, select an option:

   - **Create new GT-SUITE model**

     Choose this option to create and initialize a coupling model.

   - **Re-initialize existing GT-SUITE model**

     Choose this option to make changes to, and then re-initialize, an existing GT-SUITE model.

3. If, in the previous step, you chose to re-initialize an existing GT-SUITE model, set **Model Name** to the name of the existing model.

4. Set **Model File** to the name of a `.gtm` file that contains the GT-SUITE model.

   To do this, you can optionally click *Browse*  to use the **Select GT-SUITE File** dialog box.

5. Set **GT-SUITE Version** according to the version of GT-SUITE that wrote the `.gtm` file.

6. If a `.dat` file is available, optionally select **Read setup from related .dat file if available** to quicken the setup process.

   Note:

   You must ensure that the `.dat` file is up-to-date with the corresponding GT-SUITE `.gtm` file.

7. Optionally select **Specify GT-SUITE Installation** and set **GT-SUITE Installation** to the name of the installation directory, which is typically similar to the following on a Windows system:

   ```
   C:\Program Files (x86)\GTI
   ```

   You can optionally click *Browse*  to browse using the **Select Directory** dialog box.

   By specifying a path, you override the default path set by environment variable `GTIHOME`. The specified installation directory must contain the version-specific subdirectory indicated by the **GT-SUITE Version** setting.

8. Click **Read Setup From File**.

   CFX-Pre then reads the coupling information directly from the GT-SUITE `.gtm` file (or `.dat` file as applicable).

   After some time (significantly less time if a `.dat` file is used), additional tabs appear in the dialog box.

9. Configure the **Boundary Association** tab.

   For details, see Boundary Association Tab in the *CFX-Pre User's Guide*.

10. Configure the **Material Association** tab.

    For details, see Material Association Tab in the *CFX-Pre User's Guide*.

11. Configure the **Turboshaft Setup** tab.

    For details, see Turboshaft Setup Tab in the *CFX-Pre User's Guide*.

12. Configure the **Monitoring** tab.

    For details, see Monitoring Tab in the *CFX-Pre User's Guide*.

13. Configure the **Function Names** tab.

    For details, see Function Names Tab in the *CFX-Pre User's Guide*.

14. Click **OK** (or click **Apply** then **Close**).

Initialization of GT-SUITE Coupling results in:

- Functions of type `GT-SUITE Patch` and `GT-SUITE Turboshaft`.

  For details, see GT-SUITE Patch Functions in the *CFX-Pre User's Guide*

- A new/updated `GT-SUITE Models` object in the `Outline` tree view under `Simulation` > `Expressions, Functions and Variables` > `GT-SUITE Models`.

  For details, see GT-SUITE Model Object in the *CFX-Pre User's Guide*.

- For turboshaft cases, GT-SUITE domain motion settings in selected domains.

For details, see GT-SUITE Domain Models in the *CFX-Pre User's Guide*.

- **GT-SUITE Boundary Conditions** settings in the selected boundaries.

  For details, see Use GT-SUITE Coupling in the *CFX-Pre User's Guide*.

- Solution monitors, as specified on the **Monitoring** tab of the **Initialize GT-SUITE Coupling** dialog box.

  For details, see Monitoring Tab in the *CFX-Pre User's Guide*.

14.3.3.2. Editing a GT-SUITE Model

The GT-SUITE Model object contains a mapping between materials in CFX and materials in GT-SUITE. When you initialize a GT-SUITE model, material names are read from the GT-SUITE definition (`.gtm`) file. You must set up mapping between the materials in the `.gtm` file and the materials available in CFX-Pre. In the **Material Mapping** list, choose a GT-SUITE material, then in the **Material Name** field below the list, choose the corresponding CFX material.

14.3.4. Restarting CFX Analyses as Part of GT-SUITE Coupling

CFX does not manage any GT SUITE files (such as `.gtm`, `.dat`, `.cfdreload`, etc.). You may need to back up GT-SUITE files manually before restarting a GT-SUITE coupling simulation.

Chapter 15: Aerodynamic Noise Analysis

This chapter describes:

15.1. Overview of Aerodynamic Noise Analysis
15.2. Types of Noise Sources
15.3. Noise Source Strength Estimation in CFX
15.4. The CGNS Export Data Format

15.1. Overview of Aerodynamic Noise Analysis

Aerodynamic noise results from the propagation of disturbances through a compressible fluid (usually air), caused by the movement of objects or the fluid at some point in time and space. Whether the disturbance is caused by an object or the fluid itself, this results in fluctuations in fluid pressure (or equivalently, density) that propagate outwards from the source at the speed of sound relative to the fluid. The size of the pressure fluctuation relative to the background pressure is called the sound pressure level. Due to the extremely large range over which the human ear is sensitive, the sound pressure level is given on a logarithmic scale as a function of the pressure fluctuation:

$$SPL = 20 \operatorname{Log}_{10}\left(\frac{p'}{p_{\text{ref}}}\right) \tag{15.1}$$

where p' is the pressure fluctuation relative to the background pressure and p_{ref} is a reference pressure level, commonly chosen as 10^{-5} Pa. The units of sound pressure level are the Decibel [dB]. When evaluating aerodynamic designs, predicting the sound power per unit area, or sound intensity level, is also important. This is related to the sound pressure level by:

$$I = \frac{p'^2}{2\rho c} \tag{15.2}$$

where ρ is the density and c is the speed of sound in the background fluid. The sound intensity is also usually given on a logarithmic scale and is calculated using an equation of the same form as Equation 15.1 (p. 545) using the intensity level and a reference intensity of 10^{-12} W/m^2.

In the context of a CFD calculation, aerodynamically generated noise is of primary interest. Ideally, one would like to directly predict the pressure fluctuations at any distance from the device. However, practical considerations limit the ability to make direct predictions with a CFD solution.

Pressure and velocity fluctuations, as well as movement of the simulated components on the CFD mesh, generally result in near field noise, although this may depend on the exact definition of the "near field." Regardless of its definition, the near field will likely include the locations where the maximum noise levels occur. For most CFD applications, the extent of the computational domain is usually such that near field noise can be both analyzed and predicted.

In contrast, far field noise is the noise level that occurs at some distance from the device, usually outside of the CFD computational domain. The near field pressure and velocity fluctuations play a key role in determining far field noise levels by acting as a localized source of noise for far field noise prediction. Usually far field noise is what most aerodynamic designers are concerned about. For example, it might be desirable to create designs that minimize the sound pressure level at a certain distance from the near field source.

15.1.1. Near Field Noise Prediction

Near field noise is of interest when you want to get a feeling for the maximum sound levels occurring directly adjacent to your device, or when designing devices that take advantage of acoustic-flow couplings. For example, maximizing acoustic resonance at a given frequency in some combustor designs is used to enhance combustion efficiency (pulse combustors are a common example). Using a time accurate solution of the compressible Navier-Stokes equations, CFX can directly analyze near field noise and its effects upon the flow field. The CFX expression language can easily be used to calculate the near field sound pressure levels using both Equation 15.1 (p. 545) and Equation 15.2 (p. 545) and algebraic Additional Variables.

15.1.2. Far Field Noise Prediction

Direct prediction of mid- to far field noise by a CFD calculation is made somewhat impractical because of the meshing and timestep requirements. Firstly, you must solve the time-accurate, compressible Navier-Stokes equations. Accurately resolving the propagation of acoustic waves imposes timestep restrictions such that the Courant number is in the range of 1-2 at most. This restriction can be highly costly. Secondly, the CFD mesh must span all the way to the reception points with enough spatial resolution to directly resolve acoustic waves over the propagation distance with minimal to no numerical damping. These requirements do not make practical sense for many industrial applications.

Practical predications of far field sound pressure levels are made by first starting with a time accurate CFD calculation of the near field region, usually using URANS, LES, DES, SAS or possibly even DNS. Then, assuming that the fluid is a perfect gas, a forced version of the homogeneous wave equation is solved using models for boundary and interior noise sources taken as input from the transient CFD calculation. This solution strategy decouples the source of the noise from the sound propagation and is called the Lighthill, [178] and [179], *acoustic analogy*.

So, rather than directly predicting far field noise with a time accurate compressible Navier-Stokes solution, you solve for the density fluctuations about a user selected ambient condition:

$$\frac{\partial^2 \rho'}{\partial t^2} - c^2 \frac{\partial^2 \rho'}{\partial x_i^2} = \frac{\partial^2 T_{ij}}{\partial x_i \partial x_j} \tag{15.3}$$

where T_{ij} is the Lighthill tensor and ρ' is the density fluctuation with respect to the ambient condition. Equation 15.3 (p. 546) is the non-homogeneous (driven) wave equation and sometimes called the Lighthill equation. It assumes that the ambient fluid is an isothermal polytropic gas so that the pressure fluctuation is directly proportional to the density fluctuation. As a result, the equation can be written in terms of either pressure or density variation. The Lighthill tensor has three components and is given by:

$$T_{ij} = \rho u_i u_j + \tau_{ij} - c^2 \rho' \delta_{ij} \tag{15.4}$$

where the first term is the instantaneous Reynolds stress and τ_{ij} is the stress tensor (normal, including pressure, and shear components). Solutions of the Lighthill equation give the spatial and temporal

magnitude and distribution of density fluctuations due to interior noise sources generated by the Lighthill tensor. Interior sources result from flow structures such as wakes and shear layers. Surface based sources, not shown in Equation 15.3 (p. 546), can also contribute to density fluctuations and are further discussed in the following section.

One approach to solving the Lighthill equation would be to predict the noise source strength on a separate acoustic mesh, which is normally much coarser than the original CFD mesh, and may cover a much larger spatial extent. This approach significantly reduces the cost of solving for the far field noise over that of a full solution of the compressible Navier-Stokes equations but can still account for complex effects such as reflections, diffraction or absorption by boundary conditions. Other, less computationally expensive approaches are also possible.

Any method that you use to solve the Lighthill equation is a one-way approach that ignores coupling between acoustics and the flow field. Hence, processes that consider this coupling cannot be analyzed with this approach. As a result, the one way methodology is limited to predicting noise mainly for far field regions around aerodynamic devices. Prediction of noise using this approach for a confined, internal flow applications will be of limited usefulness because the flow tends to be more strongly coupled with the acoustics.

Additionally, without even solving the Lighthill equation in some way, it is valuable to evaluate the strength of the various noise sources. This is the approach that can be directly used within CFX to compare designs, as well as to analyze the relative strengths of the noise sources. In addition, the surface noise sources can be written from the flow solver and post processed using software that solves the Lighthill equation.

15.2. Types of Noise Sources

As mentioned in the last section, noise sources for the Lighthill equation can arise from several sources. These include displacement of the fluid by a moving boundary, pressure fluctuations on surfaces within the CFD calculation (these may or may not be wall boundary conditions), as well as interior flow features. The first of these sources is called a monopole source. The second is called a dipole source, if the surface is stationary, or it is called a rotating dipole source, if the surface is rotating at a fixed angular velocity about an axis. Finally, interior sources are usually referred to as quadrupole sources.

A generalized version of the Lighthill equation that represents the three possible noise sources was first presented by Williams and Hawkins [180] as follows:

$$\frac{\partial^2 \rho'}{\partial t^2} - c^2 \frac{\partial^2 \rho'}{\partial x_i^2} = \frac{\partial}{\partial t}\left\{\rho_0 v_n \delta(f)\right\} - \frac{\partial}{\partial x_i}\left\{\tau_{ij}\,\delta(f)\right\} + \frac{\partial^2 T_{ij}}{\partial x_i\,\partial x_j} \tag{15.5}$$

where $\delta(f)$ is the Dirac delta function describing the moving surface geometry (f is non-zero in the fluid region and zero at the moving boundary), v_n is the normal velocity of the surface, and τ_{ij} is the fluid stress tensor given by:

$$\tau_{ij} = p\,\delta_{ij} + \mu_{eff}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\,\mu_{\text{eff}}\,\frac{\partial u_k}{\partial x_k}\,\delta_{ij} \tag{15.6}$$

The first term on the right hand side of Equation 15.5 (p. 547) represents the monopole source. The second term represent the dipole sources (possibly rotating). The third term represents the quadrupole sources.

15.2.1. Monopole Sources

This is a source that occurs as a result of fluid being displaced by the swept volume of a moving boundary as it propagates through its range of motion. Sometimes this noise source is also called the "self noise" or the "thickness noise" of the device. Monopole source strength is a function of the normal velocity of the surface.

15.2.2. Dipole and Rotating Dipole Sources

Dipoles and rotating dipole sources arise from the fluctuation of forces on the surface. Sometimes these sources are also referred to as "loading noise." Strictly speaking, these forces cannot be evaluated without knowledge of the acoustic field. However, the Lighthill analogy neglects any two-way coupling, and the forces due to acoustic fluctuations are much smaller than the dynamic forces. Hence, it is perfectly valid to only consider the dynamic forces created by the bulk fluid flow.

In addition, it is common to neglect the viscous component of the force because that is usually a smaller fraction of the normal (static pressure based) component of the force.

15.2.3. Quadrupole Sources

Quadrupole sources arise in the interior of the flow and result from turbulence fluctuations in wakes, shear layers, or interaction of flow features. Because many of these features are three-dimensional in nature, quadrupole sources are very expensive to evaluate and store. This is mainly because transient data for a symmetric tensor is needed over some three dimensional region of the CFD calculation.

Usually the density fluctuation term in the Lighthill tensor Equation 15.4 (p. 546) is neglected because for turbulent flows, it is much smaller than the combined instantaneous Reynolds and compressive stresses.

15.3. Noise Source Strength Estimation in CFX

Without directly solving either the Lighthill or FW-H equation to predict the sound pressure levels or power at some distance from the source, it is still useful to analyze the four types of sources (monopole, dipole, rotating dipole, and quadrupole). Analysis of the various source terms allows comparative evaluations of different designs, as well as comparison of the relative strengths of the various types of source terms. The main objectives of analysis of the various noise sources are:

- Comparison of the relative noise source strengths for a given configuration. This type of comparison will enable you to determine which type of noise source dominates in your problem (monopole, dipole, or quadrupole). Once the dominant noise source is known, this reduces the cost of far field noise predictions because less overall data can be written.
- Designs can be compared against one another to determine which produces less noise.
- The maximum noise source level can be deduced by postprocessing the noise sources.
- The noise source strengths can be written to files and used for input into any far field noise prediction software (for example, LMS Sysnoise or Virtual Lab Acoustics, or your own model).

There are various approaches for analysis of noise sources in CFX discussed in the following sections. Near field noise can be analyzed using CEL expressions and flow solver monitor points to create plots

of sound pressure levels within the computational domain. While far field noise cannot be directly predicted, the common sources can be post processed and can also be written as transient solution fields from the flow solver on selected boundary conditions. You can control the data export using the **Export Results** tab on the **Output Control** dialog box in CFX-Pre. All files are written in a binary format called CGNS (http://www.cgns.org). The CGNS exported data is described in a later section.

15.3.1. Monopole Sources

15.3.1.1. Monopole Data in the CFX-Solver Manager

Monopole sources are related to the movement of the surface of the sources. Averaged values, as well as minimum and maximum values, of the monopole sources can be monitored in the CFX-Solver Manager using monitor points created in CFX-Pre. The monitor point data will appear in the **User Points** tab in the CFX-Solver Manager. These data can be exported from within the CFX-Solver Manager, if desired. For details, see Exporting Plot Variables and Coordinates in the *CFX-Solver Manager User's Guide*.

15.3.1.2. Exporting Surface Monopole Data

For a rotating device, the monopole source strength depends on the component of the rotation velocity normal to the rotating surface.

Monopole source output is a built-in option for the results export. To export the data, you need to create an export results object with an export surface that uses the `Acoustic Monopole` option. In this case, the CFX-Solver writes out the acoustic monopole data on the selected boundaries to a file in the requested file format.

Acoustic monopole data is available for transient simulations only.

15.3.2. Dipole or Rotating Dipole Sources

15.3.2.1. Dipole Data in the CFX-Solver Manager

Dipole and rotating dipole sources result from the fluctuation of the fluid pressure and shear stress field at the boundaries of the CFD domain. Averaged, minimum or maximum values of the pressure or wall shear stress can be monitored in the CFX-Solver Manager using monitor points created in CFX-Pre. The monitor point data will appear in the **User Points** tab in the CFX-Solver Manager. These data can be exported for external analysis if so desired. For details, see Working with Monitors in the *CFX-Pre User's Guide* and Exporting Plot Variables and Coordinates in the *CFX-Solver Manager User's Guide*.

15.3.2.2. Exporting Surface Dipole Data

Dipole and rotating dipole source output are built in options for the results export. You simply create an export results object, again with an export surface that uses either the `Acoustic Dipole` or `Acoustic Rotating Dipole` options. For these two options, the flow solver automatically writes the necessary data to CGNS files during the run. A single mesh file is written at the beginning of the run, which contains the surface grid coordinates and mesh topology. For rotating dipole sources, the mesh file also includes the normal area vectors in addition to the surface grid and topology.

In both cases, the solver writes the static pressure on the selected boundaries to a file for each timestep. It is important to note that the shear stress contribution to the force on the boundaries is neglected in this case. The second and third terms in Equation 15.6 (p. 547) are ignored. This is normally a reasonable assumption, especially for aerodynamic flows, because the normal force will dominate the noise source.

15.3.3. Important Boundary Results Export Notes

- The mesh file is written once at the beginning of the run and contains the surface grid coordinates, the element topology (element to node connectivity) and, for some sources, the nodal area vectors.
- When multiple boundary conditions are selected the flow solver will concatenate these together to form one large region, as long as the boundaries do not span CFX domains. When the selected boundary conditions span CFX domains, they are broken up into one export surface per CFX domain.
- Multiple export surfaces can be used for each export object and each export surface can be of a different type.
- The filename for the export will be set to the name of the export results object. This can be overridden using the `Filename Prefix` parameter.

15.3.4. Quadrupole Sources

Quadrupole sources are related to turbulence fluctuations in the flow field. The Lighthill stress tensor Equation 15.4 (p. 546), neglecting the pressure fluctuation term, is available from the CFX-Solver and can be monitored in the CFX-Solver Manager. It is not only possible to monitor averaged or minimum and maximum values of the quadrupole sources, but also to monitor individual components of the Lighthill stress tensor. These values can be monitored using monitor points created in CFX-Pre. The monitor point data will appear in the **User Points** tab in the CFX-Solver Manager. These data can be exported if desired.

15.3.4.1. Exporting Acoustic Quadrupole Data

Quadrupole source data is available for transient simulations only. The Lighthill stress tensor and its components are available to export to results, backup and transient results files. To write the data to file, selecting the Lighthill stress tensor from the list of selected variables is required.

15.4. The CGNS Export Data Format

By default, all boundary results export files are written in CGNS V2.4 format. This is a completely open format and the source code for the library can be downloaded from http://www.sourceforge.net. Examples and documentation are also available from the official CGNS website at http://www.cgns.org. You might also find it useful to download and compile the CGNS Tools from the source forge site as well. The ADF Viewer in particular is useful for inspecting the CGNS files written by the flow solver.

Even though the format is fully open, there is still considerable room for picking various options such as how files are named and exactly how they are internally structured. This section documents most of that information.

15.4.1. File Naming Conventions

During the run, the flow solver writes the mesh and the solution data to the temporary run directory using the following naming conventions:

`<prefix>_mesh.cgns` mesh file

`<prefix>_<timestep>.cgns` solution file

where:

- `<prefix>` is the filename prefix. By default, this is set to the export results object name. The default can be overridden by you during preprocessing.
- `<timestep>` is the timestep number. This is a six-character string containing the timestep number. The string is formatted and right justified. For example, 000010, 000020, 000030, ... This enables files to show up in order when displayed using a directory listing. It also makes batch postprocessing the files easier.

At the end of the run, the files are stored in a saved version of the temporary directory called:

`<filename>_<run number>`

where `<filename>` is the basename of the CFX-Solver input file or results file from which the run was started and `<run number>` is the three character string indicating how many runs have been performed for this case. The run number is accumulated across restarts so that directories over a sequence of two or more runs might be labeled:

`lescylinder_001/`, `lescylinder_002/`, `lescylinder_003/`, ...

where the flow solver, in this case, was originally started using a CFX-Solver input file named `lescylinder.def`. For each run, an independent set of CGNS files will be stored separately in each directory.

15.4.1.1. Important Notes

- The flow solver makes no attempt to keep track of previously written source files across restarts. Files are simply written out during each run and it is up to you to keep track of the sequence.
- The mesh data is written only once at the beginning of the run to minimizes the data output. For rotating dipole results, the normal area vectors are also stored in the initial mesh file.

15.4.2. CGNS File Structure

The generic details of how to read CGNS files and the file structure are fully documented in the documentation available from the CGNS website.

In the following sections, it may be useful to work from an example of the CFX preprocessing setup in this case. The following flow solver CCL was created using a transient version of the CFX axial turbine MFR tutorial:

```
OUTPUT CONTROL:
    EXPORT RESULTS: export1
```

```
      Option = Surface Data
      EXPORT FORMAT:
        Filename Prefix = trouser
        Option = CGNS
      END
      EXPORT FREQUENCY:
        Option = Time Interval
        Time Interval = 2.124E-5 [s]
      END
      EXPORT SURFACE: rotating dipole
        Option = Acoustic Rotating Dipole
        Output Boundary List = Blade
      END
      EXPORT SURFACE: regularities
        Option = Acoustic Dipole
        Output Boundary List = Shroud,Shroud 2
      END
    END
    EXPORT RESULTS: export2
      Option = Surface Data
      EXPORT FREQUENCY:
        Option = Time Interval
        Time Interval = 2.124E-5 [s]
      END
      EXPORT SURFACE: dipole
        Option = Acoustic Dipole
        Output Boundary List = Blade 2
      END
    END
  END
```

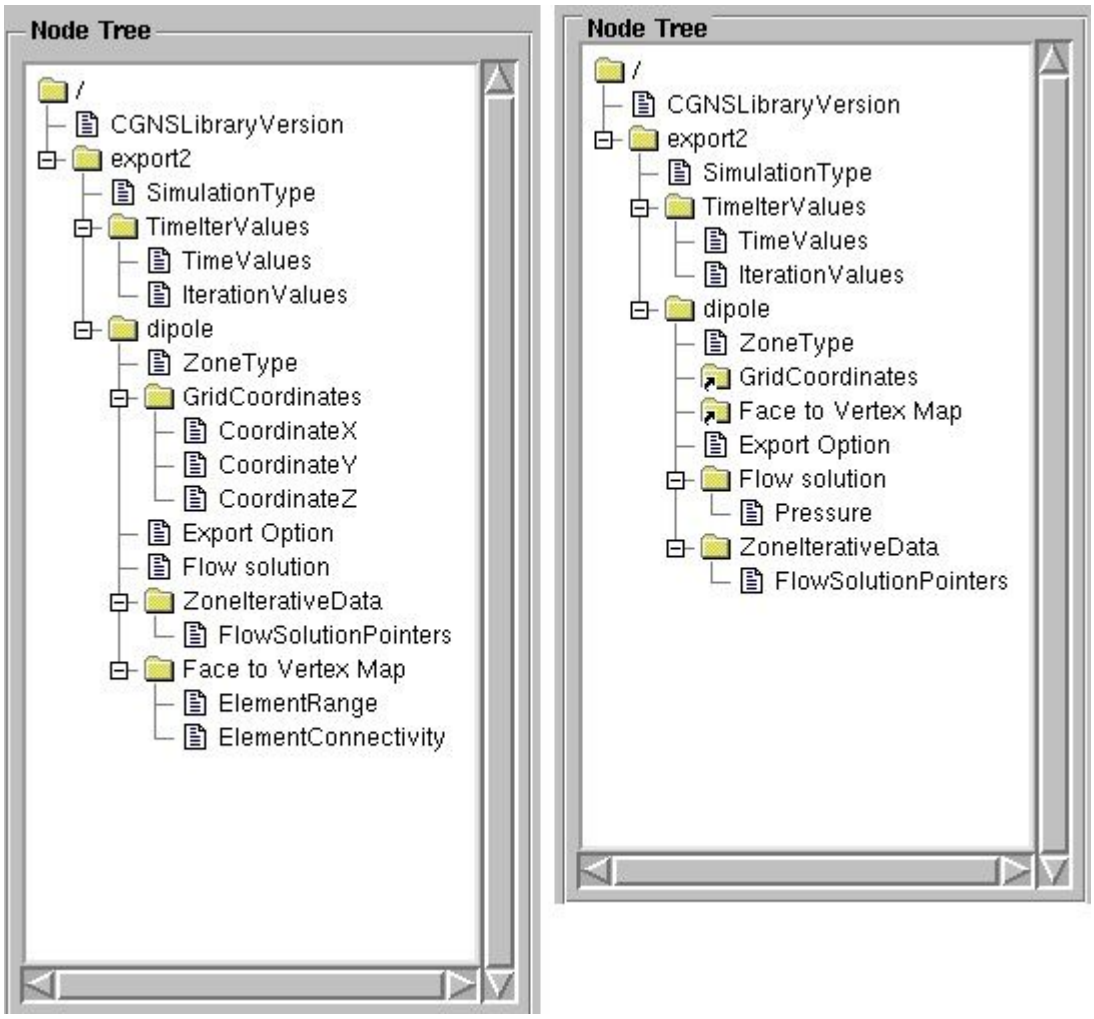
In this example, the user has selected to output two different groups of files corresponding to the two objects, `export1` and `export2`, at the same time interval. The object names could be any string and by default the object name is used as the file prefix unless the user overrides that by setting the `Filename Prefix` option under `EXPORT FORMAT`.

For export1, the user has selected to output a rotating dipole surface source called `rotatingdipole` on the boundary called `Blade` (which happens to be in a rotating domain). In addition, a regular dipole source, called `regulardipole`, will be written on boundaries `Shroud` and `Shroud 2`. In this example, `Shroud` is in a rotating domain and `Shroud 2` is in a stationary domain. This is an example of where the flow solver will split the surface source into two different regions because the surface spans across two CFX domains.

For `export2`, they have selected to output a dipole source, called `dipole`, on `Blade 2`, which is in a stationary domain.

The export surface names in both cases are also provided by the user, in addition to the export results names, during CFX pre-processing.

The tree structure of the `export2` results output looks as follows:



On the left is the grid file and on the right is a solution file. All CGNS files contain a root node called "/" and are set up like a file system below that root node. Below the root node is CGNSLibraryVersion, which gives the CGNS version used to write the file. Both mesh and solution files contain a base node called `export2`, which corresponds to the results object name and the file prefix. Below that node, the following nodes are defined:

15.4.2.1. Common Nodes for Mesh and Solution Files

`SimulationType` simply contains a string stating that the solution is written from a time-accurate or nontime-accurate simulation. Acoustic output will always be from a time-accurate simulation in CFX.

`TimeIterValues` is a `BaseIterativeData_t` node that contains the time values and timestep data for the solutions in the file.

`TimeIterValues/TimeValues` is a `DataArray_t` node that contains a list of real values corresponding to the simulation times stored in the file. In our case, there is only ever one time value inside a single CGNS file.

`TimeIterValues/IterationValues` is also a `DataArray_t` node that contains the integer value corresponding to the timestep number. This number is exactly the same as that encoded in the file name.

dipole is a CGNS Zone_t node and contains the grid coordinates and nodal area vectors in the mesh file. The solution file contains the boundary pressures and links back to the grid coordinates and the element to vertex map.

ZoneType is a processor_t node that contains the type of grid written to the file and is always set to Unstructured. This is a default CGNS name.

Export Option is written for all regions and contains a string that tells what kind of surface export option was used to create the data. In the two examples, this string with be either Acoustic Dipole or Acoustic Rotating Dipole.

Flow Solution is a FlowSolution_t node that contains the solution values below it written on the element nodes. In the mesh file, this directory contains nothing for this case because the file only contains data for a dipole source. For rotating dipole source cases, this directory contains the nodal area vector components called Nodal Area VectorX, Nodal Area VectorY, Nodal Area VectorZ. (Normally, the CGNS documentation has a suggested naming convention for variables written to FlowSolution_t nodes. There is no suggested name for Nodal Area Vectors but this format follows the suggested naming convention for vectors.)

ZoneIterativeData is a ZoneIterativeData_t node that contains a single node below it called FlowSolutionPointers. This node is a DataArray_t node that contains an array of character strings pointing to the directory names for flow solutions at different timestep values. Because only one timestep is written to each CGNS file, this node only holds one character string called Flow Solution.

15.4.2.2. Mesh File Specific Nodes

GridCoordinates is a GridCoordinates_t node that contains the surface mesh <x,y,z> coordinates in the flow solver coordinate system.

CoordinateX/Y/Z are just DataArray_t nodes with the grid coordinate components.

Face to Vertex Map is an Elements_t directory node that contains the element range and connectivity.

Face to Vertex Map/ElementRange is an IndexRange_t node that gives the range of elements assigned to the region. In all cases, this node gives two numbers 1 and N, where N is the number of surface elements written for the region. Face to Vertex Map / ElementConnectivity is a DataArray_t node that contains the fact to vertex map for each boundary face (2D QUAD_4) element. An example of this array with both triangular and quadrilateral faces is as follows:

```
465 152 155 155
459 176 178 178
579 154 157 157
579 156 154 154
465 157 152 152
2 32 31 1
1 31 462 461
3 33 32 2
4 34 33 3
5 35 34 4
6 36 35 5
7 37 36 6
```

```
8 38 37 7
9 39 38 8
```

The first five elements on this boundary are actually triangular faces written as degenerate quadrilaterals. The remaining elements are real quadrilaterals. The elements are always written to the file with the CGNS element type QUAD_4.

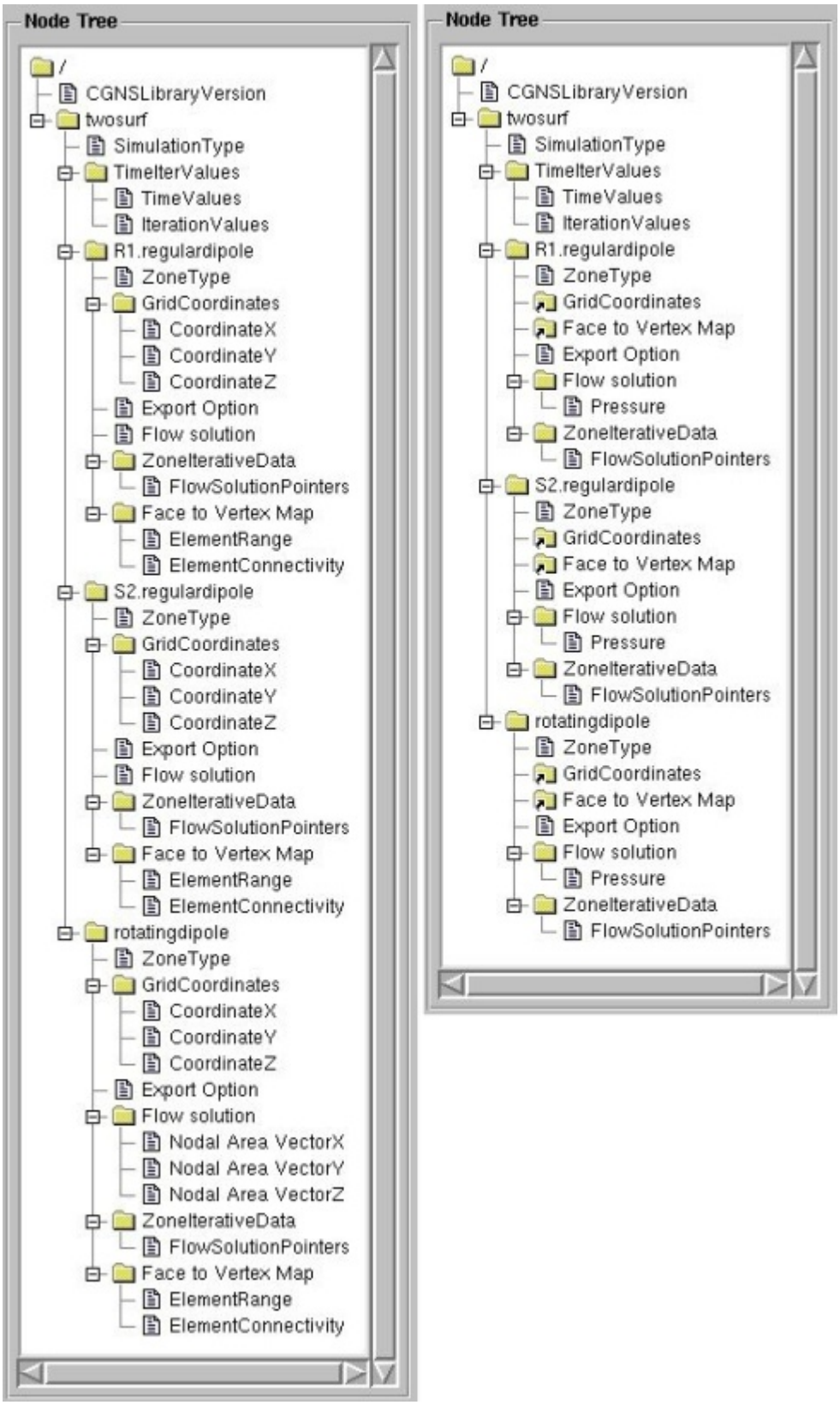
15.4.2.3. Solution File Specific Nodes

`GridCoordinates` is only a link node that gives the filename of the mesh file corresponding to the solution.

`Face to Vertex Map` is another link node that gives the filename of the mesh file corresponding to the solution.

`FlowSolution/Pressure` is a `DataArray_t` node that contains the boundary pressures on element vertices.

The last example shows a case where the dipole export surface has been broken into two regions because the boundary list spans across more than one CFX domain. In this case, one domain is rotating (R1) and the other is stationary (S2):



The main difference between this example and the last one is that the split regions have been prefixed with the corresponding CFX domain name and a period character. All other nodes contain the exact data as the simple case previously described.

Chapter 16: Advice on Flow Modeling

This chapter describes:

16.1. Solving Problems with Ansys CFX
16.2. Modeling 2D Problems
16.3. Mesh Issues
16.4. Timestep Selection
16.5. Advection Scheme Selection
16.6. Advanced Options: Dynamic Model Control
16.7. Pressure Level Information
16.8. Interpolation Scheme
16.9. Temperature Damping
16.10. Monitoring and Obtaining Convergence
16.11. Solver Issues
16.12. How CEL Interacts with the CFX-Solver
16.13. Best Practice Guides

16.1. Solving Problems with Ansys CFX

CFX-Solver is extremely robust and can reliably give solutions to a wide range of complex flow problems. However, there are some situations where obtaining a converged solution is not straightforward.

This section provides advice on setting appropriate solver parameters and on identifying, diagnosing and solving potential problems quickly and easily.

Issues associated with multi-component or multiphase flow are available in Multicomponent Flow (p. 64) and Multiphase Flow Modeling (p. 295).

Modeling advice for free surface flow is available in Modeling Advice for Free Surface Flow (p. 348).

16.2. Modeling 2D Problems

The following is advice for modeling 2D problems:

- Make the mesh only 1 element thick. More elements will slow computational time and require more memory.
- For planar 2D geometries, apply symmetry conditions to the front and back planes. Free Surface Flow Over a Bump is one example of a case that uses this setup. Do not use free slip walls; doing so will

hurt accuracy because control volume gradients will not be computed. The extrusion distance should be on the order of the smallest mesh dimension.

- For axisymmetric 2D geometries, apply symmetry conditions to the high-theta and low-theta planes unless there is swirl anticipated in the flow, in which case, 1:1 periodic connections should be applied instead. Do not use GGI periodic connections; doing so will hurt accuracy. The extrusion rotation angle for axisymmetric geometries should be small (for example, 1 to 5 degrees).

16.3. Mesh Issues

The ability to obtain accurate results is significantly affected by the mesh distribution. In the discussion that follows, the relationship between the mesh distribution and the following topics is addressed:

- Physical Modeling Errors: YPLUS and Mesh Resolution Near the Wall (p. 558)
- Measures of Mesh Quality (p. 558)

16.3.1. Physical Modeling Errors: YPLUS and Mesh Resolution Near the Wall

Unless you are specifically interested in resolving the boundary layer profile in your simulation, Ansys CFX will use wall functions to model the near wall region in a turbulent flow simulation.

Wall functions are extremely useful in reducing computational load, but you should be aware of their limitations. These limitations are best understood in terms of the parameter y^+ (Yplus), which is a non-dimensional variable representing the distance from the wall to the first node away from the wall. It is therefore dependent on the size of the mesh in the wall region. If the value of y^+ is too large, then the wall function will impose wall type conditions further from the wall than would normally be physically appropriate. The use of scalable wall functions in Ansys CFX has removed problems associated with the lower valid limit for y^+.

You can plot the values of y^+ at the wall in CFD-Post to see where the mesh needs refining (or coarsening) to reduce (or increase) the value of y^+ in that region.

For more information see Scalable Wall Functions in the *CFX-Solver Theory Guide* and Solver Yplus and Yplus in the *CFX-Solver Theory Guide*.

16.3.2. Measures of Mesh Quality

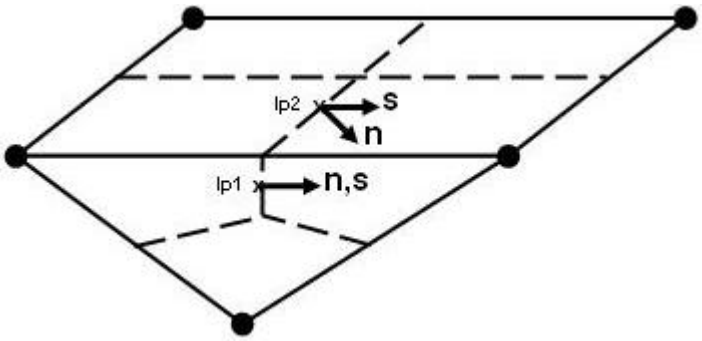
Using a mesh of adequate geometrical mesh quality is an important part of controlling discretization error. For details, see Discretization Errors in the *CFX-Solver Theory Guide*. It is also important for avoiding round-off errors during, for example, the solution of the linear equations that are produced by the discretization process. Significant measures of mesh quality may be broadly categorized as measures of mesh orthogonality, expansion and aspect ratio (or stretching).

Various forms of these measures are presented during different stages of the simulation process (for example, mesh generation, physics preprocessing, solution, and so on). It is important to realize, however, that the most relevant form of the orthogonality, expansion and aspect ratio measures is intimately related to the discrete approximations employed by the field solver being used. Moreover, acceptable ranges for the values of these measures also depend heavily upon the discretizations used.

The discussion that follows focuses on the measures that are most relevant to the CFX-Solver. The relationship between these measures and some of the values that are available in mesh generators or physics preprocessors or postprocessors is also noted.

16.3.2.1. Mesh Orthogonality

The concept of mesh orthogonality relates to how close the angles between adjacent element faces or adjacent element edges are to some optimal angle (for example, 90° for quadrilateral faced elements and 60° for triangular faces elements). The most relevant measure of mesh orthogonality for the CFX-Solver is illustrated below. It involves the angle between the vector that joins two mesh (or control volume) nodes (s) and the normal vector for each integration point surface (n) associated with that edge. Significant orthogonality and non-orthogonality are illustrated at ip1 and ip2, respectively.



The orthogonality angle, three related measures, and acceptable ranges are tabulated below with other measures that are available through the CFD-Post postprocessor and Ansys ICEM CFD meshing tools. Values outside of the suggested acceptable range will increase both sources and the amplification of discretization error. Poor convergence and divergence can be expected under these conditions.

Orthogonality Measure	Acceptable Range	Description and Notes
Orthogonality Angle	> 20°	Area weighted average of $90° - a\cos(n \cdot s)$ for all integration point surfaces of a control volume • Large values indicate good orthogonality • One of the best indicators of how and where poor orthogonality will adversely affect general solution accuracy and robustness
Orthogonality Factor	$> \frac{1}{3}$	Area weighted average of all integration point surface scalar products of unit n and s vectors (that is, n·s) associated with each control volume • Large values indicate good orthogonality

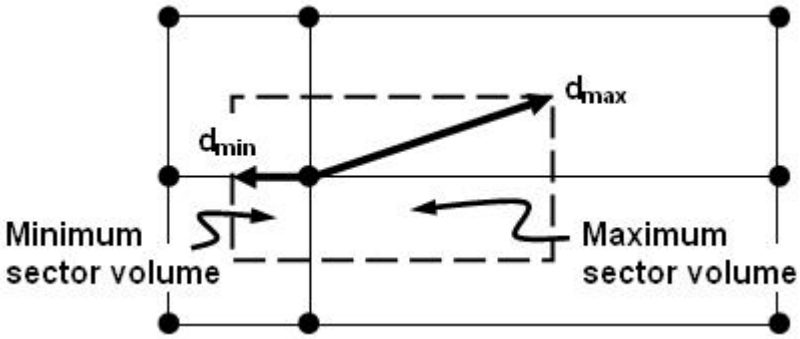
Orthogonality Measure	Acceptable Range	Description and Notes
		• One of the best indicators of how and where poor orthogonality will adversely affect general solution accuracy and robustness
Orthogonality Angle Minimum	> 10°	Minimum of $90° - a\cos(n \cdot s)$ for all integration point surfaces of a control volume • Similar to orthogonality angle, but adverse effects are likely to be less globally significant
Orthogonality Factor Minimum	$> \frac{1}{6}$	Minimum of all integration point surface scalar products of unit n and s vectors (that is, n·s) associated with each control volume • Similar to orthogonality angle, but adverse effects are likely to be less globally significant
Minimum/ Maximum Face Angle (CFD-Post)	> 10° or < 170°	Minimum/maximum angle between edges of each face that touches a node • Usefulness of this orthogonality measure is limited by its two-dimensional nature • Acceptable minimum/maximum values do not ensure acceptable orthogonality (for example, a flattened tetrahedron can have minimum angles of 30°)
Minimum/ Maximum Dihedral Angle(Ansys ICEM CFD)	> 10° or < 170°	Minimum/maximum angle between element faces • Inherent three-dimensional nature increases the usefulness of this measure • Most similar measure to orthogonality angle minimum

16.3.2.2. Mesh Expansion

The concept of mesh expansion relates to rate of change in the magnitude of adjacent element face areas or volumes.

The most relevant measure of mesh expansion for the CFX-Solver is illustrated below. It involves the ratio of the maximum to minimum distance between the control volume node and the control volume boundaries. Because this measure is relatively expensive to calculate for arbitrarily shaped

control volumes, an alternative formulation, the ratio of maximum to minimum sector volumes, is used.

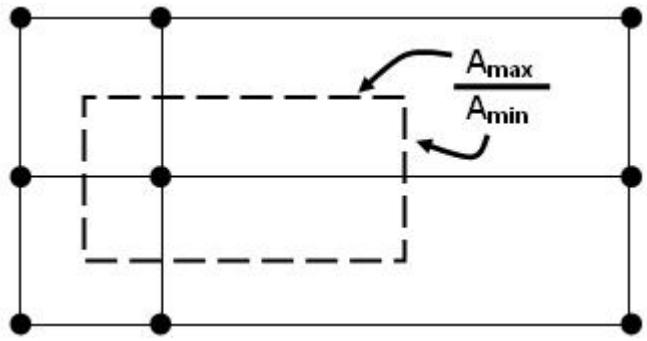


This measure and its acceptable values are tabulated below, along with a measure that is available through the CFD-Post postprocessor. Values outside of the suggested acceptable range will increase sources of error that are due to the discretization of transient and body force terms.

Mesh Expansion Measure	Acceptable Range	Description and Notes
Mesh Expansion Factor	<20	Ratio of largest to smallest sector volumes for each control volume
Element Volume Ratio (CFD-Post)	<20	Ratio of largest to smallest element volumes that surround a node

16.3.2.3. Mesh Aspect Ratio

The concept of the mesh aspect ratio relates to the degree that mesh elements are stretched. The most relevant measure of aspect ratio for the CFX-Solver is illustrated below. It involves the ratio of the maximum to minimum integration point surface areas in all elements. Nodal (that is, control volume) values are calculated as the maximum of all element aspect ratios that are adjacent to the node.



The area based measure is tabulated below along with another measure that is available through the CFD-Post postprocessor. Values outside of the suggested acceptable range will lead to round-off errors and associated difficulties converging the discretized equations.

Mesh Aspect Ratio Measure	Acceptable Range	Description and Notes
Aspect Ratio	<100[a]	Largest ratio of maximum to minimum integration point surface areas for all elements adjacent to a node
Edge Length Ratio (CFD-Post)	<100	Largest ratio of maximum to minimum integration edge lengths for all edges of element faces that touch a node

[a] The acceptable range for both measurements is less than 1000 if running double precision.

16.4. Timestep Selection

The selection of an appropriate timestep size is essential in order to obtain good convergence rates for your simulation. A general discussion on when a steady-state or a transient solution should be attempted is available in Steady State and Transient Flows (p. 41).

16.4.1. Steady-state Time Scale Control

For steady-state cases, and for transient blade row cases that use the `Harmonic Balance` transient method, the CFX-Solver applies a false timestep as a means of under-relaxing the equations as they iterate towards the final solution. Because the solver formulation is robust and fully implicit, a relatively large time scale can typically be selected, so that the convergence is as fast as possible.

These cases typically require between 50 and 100 outer loop iterations to achieve convergence. If you expect that the actual flow being simulated would take a long time to reach a steady-state condition, given the initial conditions you have specified, then a greater number of outer loop iterations may be required. In this case, you can usually reduce the number of iterations required by setting initial conditions that more closely resemble the steady-state flow.

Although a relatively large time scale can be used, if the time scale is too large the resulting convergence behavior will be "bouncy." If this is observed, then the first thing you should try is to reduce the time scale, say, by a factor of four. If there is no noticeable improvement, then the convergence problem may be caused by another factor. If the time scale is too small, then the convergence will be very slow.

In addition to the advice in the following sections, you will probably require a small physical time scale for the following situations:

- Poor mesh quality
- Transonic flow
- Large regions of separated flow
- Openings with simultaneous inflow and outflow
- Free Surface flows: In these cases it is often sufficient to use a smaller timestep only for the Volume Fraction equations

- Multiphase flows

16.4.1.1. Max. Iterations

This sets the number of outer loop iterations for the CFX-Solver. The CFX-Solver will terminate the run after this number of iterations, even if specified convergence criteria have not been reached. Most simulations will require between 50 and 100 iterations for adequate convergence. After a solver run, you should always check the reason why the CFX-Solver finished; that is, has the convergence criteria been met or has the maximum number of iterations been reached? If the CFX-Solver finished because the maximum number of iterations was reached, you may want to continue the run until the convergence criteria is achieved.

If you are using Mesh Adaption, the maximum number of iterations specified is the maximum number of iterations in the final adaption step. The total maximum number of iterations will depend on the number of adaption steps and the maximum number of iterations per adaption step.

16.4.1.2. Minimum Number of Iterations

The minimum number of iterations the CFX-Solver will run.

16.4.1.3. Time Scale Control

The time scale used by the CFX-Solver can be controlled using one of three methods:

- Auto Timescale (p. 563)
- Local Time Scale Factor (p. 565)
- Physical Time Scale (p. 565)

You can control these settings from CFX-Pre by going to the **Outline** tree view and right-clicking **Solver Control** > **Edit**. Alternatively, you can use the Command File Editor to achieve more control over the Ansys CFX-Solver. (See Controlling the Time Scale with the Command File Editor (p. 564) for details.)

16.4.1.3.1. Auto Timescale

Auto Timescale is a fluid timescale control option that uses an internally calculated physical time scale based on the boundary conditions, flow conditions, physics, and domain geometry. Information on how the CFX-Solver calculates the **Auto Timescale** is available in Automatic Time Scale Calculation in the *CFX-Solver Theory Guide*

Auto Timescale is the default timescale control setting. However, be aware that the **Auto Timescale** calculated by the solver is often conservative. This is usually robust, but faster convergence is often possible using a more aggressive setting. This can be done by:

- Setting **Length Scale Option** to **Aggressive** (see Length Scale Option (p. 564))
- Increasing the **Time Scale Factor** (see Timescale Factor (p. 564))
- Changing to an appropriate **Physical Time Scale** (see Physical Time Scale (p. 565))

It is also worth noting that, in some instances, the internal time scale calculation fails to find an appropriate velocity or length scale on which to base the time scale. For example, a buoyancy-driven flow problem can be specified using a Subdomain heat source in a cavity. If no velocity field or temperature difference is specified, a very large time scale (several orders of magnitude) can result. If this is the case, then you should try either:

- Specifying a small fixed physical time scale and follow the general guidelines about increasing or decreasing time scale accordingly
- Specifying an initial guess for velocity and/or temperature.

16.4.1.3.1.1. Length Scale Option

When using the **Auto Timescale** option for fluid domains in a steady-state simulation, the **Length Scale Option** may also be set. The length scale is used by the solver to evaluate a time scale based on L/U. The options are:

- **Conservative**: The cube root of the domain volume is used. This is the default.
- **Aggressive**: The maximum extent of the domain volume is used.
- **Specified Length Scale**: Uses a user-specified value.

16.4.1.3.1.2. Timescale Factor

The **Timescale Factor** option, set automatically to a default value of 1, can be used to multiply the **Auto Timescale** value calculated by the solver.

16.4.1.3.1.3. Maximum Timescale

The **Maximum Timescale** can be used to bound the internally-calculated time scale. The Ansys CFX-Solver uses the smaller of the internally calculated time scale and the user maximum value.

16.4.1.3.1.4. Controlling the Time Scale with the Command File Editor

Advanced control over the auto timestepping algorithm is available using CCL directly or using the Command File Editor. The following additional parameters are available:

- Timescale Update Frequency: Controls the frequency with which a new timescale is updated (default is every 5 iterations)
- Number of Timescale Updates: Controls the number of times that a new timescale is calculated (unbounded by default)
- Timescale Ramping Factor: If the Number of Timescale Updates has been passed, and the Maximum Timescale is larger than the internally-calculated timescale, the solver increases the timescale every iteration by a factor of the Timescale Ramping Factor until the Maximum Timescale is reached. The default is 1.2.

In the CCL example that follows, the solver starts with an internally calculated timescale multiplied by the Timescale Factor of 1.5. The internally calculated timescale is updated every 7 iterations, but is not permitted to exceed the Maximum Timescale of 5 [s]. The timescale is updated every 7 iterations for a total number of 12 updates. After this period, the timescale increases every iteration by a factor of 1.3 until it reaches the Maximum Timescale.

```
SOLVER CONTROL:
  CONVERGENCE CONTROL:
     Maximum Number of Iterations = 300
     Timescale Control = Auto Timescale
     Timescale Factor = 1.5
     Number of Timescale Updates = 12
     Timescale Update Frequency = 7
     Maximum Timescale = 5 [s]
     Timescale Ramping Factor = 1.3
  END
END
```

16.4.1.3.2. Local Time Scale Factor

The **Local Timescale Factor** option enables different time scales to be used in different regions of the calculation domain. The value you enter is a multiplier of a local element-based time scale. Smaller time scales are applied to regions of the flow where the local time scale is very short (such as fast flow), and larger time scales to those regions where the time scale is locally large (such as slow flow). The default value is 5. Any value less than this can be considered a "small" timestep.

This option is very useful when there are widely varying velocity scales in the simulation, for example, jet flow into a plenum chamber. The main disadvantage of this method is that very small timescales may locally result for small elements or large aspect ratio elements, which may in turn reduce the overall convergence rate. For this reason, it is best used on meshes of uniform element size and moderate aspect ratios.

16.4.1.3.3. Physical Time Scale

The **Physical Timescale** option enables a fixed time scale to be used for the selected equations over the entire flow domain.

In general, there are two situations in which you would use a physical time scale:

- To provide sufficient relaxation of the equation nonlinearities so that a converged steady-state solution is obtained, or,
- To evolve the solution through time in order to provide transient information about a time-dependent simulation.

For advection dominated flows, the physical time scale should be some fraction of a length scale divided by a velocity scale. A good approximation is the Dynamical Time for the flow. This is the time taken for a point in the flow to make its way through the fluid domain. For many simulations, a reasonable time estimate t is easy to make based on the length of the fluid domain L, and the mean velocity U, for example:

$$\Delta t \approx \frac{L}{U} \tag{16.1}$$

For external flow, L should be based upon a characteristic geometric length (for example, chord length).

If the domain contains largely varying velocity and length scales, you should try to estimate an average value. If you get divergence, check in the CFX-Solver Output file that your time scale is not larger than the **Advection Time Scale** value given in the **Average Scale Information** at the end of the run. A "small" time scale can be considered to be less than 1/3 the smallest

length / velocity scale in the simulation. Values higher than this can be considered a "large" time scale.

It is often necessary to alter the physical time scale for buoyancy driven flows in order to achieve convergence. The maximum physical time scale may be estimated using the following relationship:

$$\Delta t_{\max} \approx \sqrt{\frac{dl}{\beta \boldsymbol{g} \Delta T}} \tag{16.2}$$

where:

- l is a length scale associated with the vertical temperature gradient
- ΔT is the temperature variation in the fluid
- β is the thermal expansivity of the fluid
- $\boldsymbol{g}$ is the gravitational acceleration.

Physical time scales that are too large are characterized by bouncy convergence or results that do not converge. Time scales that are too small are characterized by very slow, steady convergence.

Depending on your initial guess, a significantly smaller time scale may be required for the first few iterations of a problem.

A judicious physical time scale value usually gives faster convergence than specifying a **Local Timescale Factor** value.

If you use an expression for the physical time scale, it must evaluate to a constant value. CEL variables are available to control the time scale depending on the iteration. For details, see Timestep, Timestep Interval, and Iteration Number Variables in the *CFX Reference Guide*.

16.4.1.4. Solid Time Scale Control

The **Solid Timescale Control** option is available if you have a solid domain and are performing a steady-state calculation. Because there is no flow in a solid domain, the relevant time scale for the solution in the solid part of the domain may be very much longer than the time scale in the fluid domain. It is usually safe to set a much larger time scale in a solid domain than would be used in a fluid domain. You can also set a time scale for the energy equation class in a solid domain (see Controlling the Timescale for Each Equation (p. 566)).

16.4.1.4.1. Solid Timescale Factor

When using the **Auto Timescale** option for the **Solid Timescale Control**, a **Solid Timescale Factor** is used. The default value is 1. How this value is used to calculate a solid time scale is described in Solid Time Scale Estimate in the *CFX-Solver Theory Guide*.

16.4.1.5. Controlling the Timescale for Each Equation

CFX gives you great flexibility in controlling the timescale used for each equation solved. You can specify a timescale on a global basis, a domain basis, a fluid basis, an equation-class basis, or an individual equation basis.

When setting timesteps like this, there will often be more than one timescale defined for any given equation. For example, you could define both a global timescale and a timescale for a single fluid in one domain. The different ways in which a timescale can be set are outlined below in decreasing order of precedence below. For a given equation, the option with highest precedence will always be used:

1. For a specific equation, in a specific phase, in a specific domain.
2. For a specific equation in a specific domain.
3. For a specific equation in a specific phase, in all domains.
4. For a specific equation in all domains.
5. For a specific equation class, in a specific phase, in a specific domain.
6. For a specific equation class in a specific domain.
7. For a specific equation class, in a specific phase, in all domains.
8. For a specific equation class in all domains.
9. For all equations, in a specific phase, in a specific domain.
10. For all equations in a specific domain.
11. For a specific phase in all domains.
12. For all equations globally.
13. For solid domains that use the global **Solid Timescale Factor** option (when **Solid Timescale Control** is set to **Auto Timescale**).

Note:

Solid Timescale and **Solid Timescale Control** are not allowed outside the global level. Options 1 to 11 can only be specified using the **Auto Timescale** or **Physical Timescale**.

An equation could be (but is not limited to): the momentum, continuity, or energy equation for a single fluid; the mass fraction equation for a single component; the volume fraction equation for a single fluid; one of the turbulence equations for a single fluid; an Additional Variable equation.

An equation class can include more than one equation. The equation classes include `momentum`, `continuity`, `energy`, `rs` (Reynolds stress), `ke` (turbulent kinetic energy), `ed` (turbulent eddy dissipation), `tef` (turbulent eddy frequency), `meshdisp` (mesh displacement), `mf` (mass fraction), `vf` (volume fraction), and `av` (Additional Variable). The first eight of these will include more than one equation only in a multiphase and/or multi-domain simulation. The `mf` class will include as many equations as there are components, the `vf` class will include as many equations as there are fluids, and the `av` class will include as many equations as there are Additional Variables.

The CFX-Pre user interface supports options 8, 12, and 13. To use other options, you must edit the CCL file.

The default approach for solid domains is therefore the lowest priority option. If you want to set a special timescale control for the energy equations in the fluid domain but not affect the solid domains, then you must use one of options 1, 2, 5, or 6. If you use options 1 to 8 for the energy equation and also specify a **Solid Timescale Factor**, the **Solid Timescale Factor** will be ignored.

16.4.2. Transient Timestep Control

Time dependent behavior for transient simulations in Ansys CFX is specified through time duration and timestep. The **Timesteps** option provides a way for Ansys CFX to track the progress of real time during the simulation, whereas **Time Duration** is a user-specified limit on the length of real time the simulation is to run.

The CFX-Solver uses the timestep and time duration specifications to determine whether to continue the simulation using the next timestep. For example, if the following are set:

- **Timesteps** is set to a single value of Δt.
- **Time Duration** is set to **Total Time** option with a value of T.

then the CFX-Solver will continue to compute solutions at each timestep iteration until $N\Delta t \geq T$, where N is the number of timesteps actually performed by the CFX-Solver. Note that the final simulation time may slightly exceed the specified time duration (for example, if adaptive timestepping is used or if the specified Δt does not divide evenly into T).

The following options can be set to control the time duration and timestep size in a transient simulation. Information on specifying these options in CFX-Pre is available; for details, see Analysis Type in the *CFX-Pre User's Guide*.

16.4.2.1. Time Duration

This is where the duration of the simulation in real time is specified. It is used to determine when to finish the transient run. It can be set to one of:

- **Total Time** - The real time, relative to the initial time for the analysis, at which to end the transient analysis. This should be greater than the initial time for a given run. Note that the initial time for the analysis is not automatically reset when continuing from a previous transient analysis.
- **Time per Run** - The real time, relative to the initial time for a given run, at which to end the transient analysis. Note that the initial time for the run is automatically reset when continuing from a previous transient analysis (for example, after a user-defined solver interrupt condition is triggered and remeshing is performed).
- **Maximum Number of Timesteps** - The timestep counter value at which to end the transient analysis. Note that the timestep counter value is not automatically reset when continuing from either a previous steady-state or transient analysis. Further, when continuing from steady-state analyses, the final outer loop iteration value is used to initialize the timestep counter value.
- **Number of Timesteps per Run** - The timestep counter value, relative to the initial value for a given run, at which to end the transient analysis.

16.4.2.2. Timesteps: Timesteps List

The **Timesteps** list is where you set the actual real time intervals at which the CFX-Solver solves for the flow field. The timesteps you select need to be based on the time scale of the transient behavior that you want to resolve in your flow simulation. The timesteps can be:

- A single timestep value, for example, `2.3`
- A comma separated list of timestep values, for example, `2.3, 3.0, 3.5, 0.1`
- Either of the above as a CEL expression, in which case units must be included. For example: `2.3 [s], 3.0 [s], 3.5 [s], 0.1 [s]`

When *not* entering the list as an expression, you can also specify a comma separated list of timestep values with multiple timesteps per item in the list. For example, specifying `5*0.1, 2*0.5, 10*1`, with the units `[s]` specified separately, would result in the following CCL:

```
0.1 [s], 0.1 [s], 0.1 [s], 0.1 [s], 0.1 [s], 0.5 [s], 0.5 [s], 1 [s],
1 [s], 1 [s], 1 [s], 1 [s], 1 [s], 1 [s], 1 [s], 1 [s], 1 [s]
```

If you accidentally enter `5*0.1 [s], 2*0.5 [s], 10*1 [s]` as an expression, the multiplication would be carried out, and the corresponding CCL that would be generated would be:

```
0.5 [s], 1.0 [s], 10.0 [s]
```

If a single value or expression is set, this timestep size is used to move forward in real time from the initial time to the end of the transient run. The end of the transient run is determined by the **Time Duration: Option** setting (see Time Duration (p. 568)).

If a list is set, each timestep size is used in turn to move forward in real time from the initial time to the end of the transient run. The end of the transient run is still determined by the **Time Duration: Option** setting. The end of the run can be reached before all timesteps in the list have been used, in which case the remaining timesteps are ignored. If all timesteps in the list are used before the end of the run is reached, then the last timestep in the list continues to be used repeatedly until the end of the transient run.

When restarting a transient run, the time completed in previous runs is considered. For example, if a list is specified and the first three timesteps in the list were completed in the initial run, then the run will restart at the fourth timestep in the list.

A simple timestep dependence study is useful to determine the effect of the timestep size on the accuracy of the results. You should initially choose a timestep size of the order of the time scales of interest, and then running a fixed but small period of the simulation in physical time. The same period is then re-run using double and half this timestep size, and the results compared. Significant differences between the initial and smaller timestep mean that an even smaller timestep is required, and this process can be repeated until the similarity in results is within some acceptable solution tolerance. Even if your timestep selection for the run is not the limiting case, this kind of analysis provides a good feel for the level of error introduced by your selection of timestep size. For details, see Max. Iter. Per Step (p. 571).

16.4.2.3. Timesteps: Timesteps for the Run List

This is the same as the **Timesteps** list described above, except that when restarting a run, the time completed in previous runs is ignored. When CCL is used to set both **Timesteps** and **Timesteps for the Run** lists, the **Timesteps** list takes precedence (see Timesteps: Timesteps List (p. 569)).

16.4.2.4. Timesteps: Adaptive

Adaptive timestepping automates the process of adapting the timestep size according to one of the following three options:

- **Num. Coeff. Loops** (Number of Coefficient Loops): a target minimum/maximum number of coefficient loops is specified. If the actual number of coefficient loops used is less than the **Target Minimum Coefficient Loops**, the timestep size is increased. If the actual number of coefficient loops used is greater than the **Target Maximum Coefficient Loops**, the timestep size is decreased. The increase and decrease factors for the timestep must also be provided by specifying values for **Timestep Increase Factor** and **Timestep Decrease Factor**. The actual min/max number of coefficient loops set in `SOLVER CONTROL` must encompass the range specified by **Target Minimum Coefficient Loops** and **Target Maximum Coefficient Loops**.

  Note:

  If short time scale transient effects need to be predicted accurately for a variable density flow, you should set a minimum of 3 coefficient loops per time step.

- **RMS Courant Number**: the timestep size is adjusted such that the specified **RMS Courant Number** condition is satisfied. The relaxation factors for the timestep must also be provided.
- **MAX Courant Number**: the timestep size is adjusted such that the specified **Maximum Courant Number** condition is satisfied. The relaxation factors for the timestep must also be provided.

In addition, each condition requires specification of following parameters:

- **First Update Time**: The timestep adaption algorithm is initiated when the simulation time reaches the specified first update time.
- **Timestep Update Frequency**: The adaptive timestep is updated with this frequency. A value of 1 means that it is updated every timestep.
- **Initial Timestep**: The adaptive timestepping algorithm is initialized with this timestep size. It is also used with the first update time is reached.
- **Minimum Timestep**: The adaptive timestep is not permitted to drop below this value.
- **Maximum Timestep**: The adaptive timestep is not permitted to rise above this value.

If adaptive timestepping is used, the final simulation time may slightly exceed the specified time duration.

16.4.2.5. Initial Time

The **Initial Time** for a transient simulation corresponds to the time before beginning the first timestep in the current simulation. The following options can be set to control the initial time:

- **Automatic**: automatically reads the **Initial Time** from an initial values file if one can be found; otherwise, it uses a default value of zero.
- **Automatic with Value**: automatically reads the **Initial Time** from an initial values file if one can be found; otherwise, it uses the specified value.
- **Value**: Always uses the specified value for the **Initial Time**. You should only use this option to reset the initial time to a specified value for a restarted run.

If the **Initial Time** is read from an initial values file, it is set to the last timestep value read from the results file. Information on specifying this option in CFX-Pre is specific.

16.4.2.6. Max. Iter. Per Step

At each timestep in a transient simulation, the CFX-Solver performs several coefficient iterations or loops, either to a specified maximum number or to the predefined residual tolerance. These settings are controlled on the Solver Control panel in CFX-Pre. The maximum number of iterations per timestep may not always be reached if the residual target level is achieved first.

Using a large number of coefficient iterations for transient runs is not recommended. Improved accuracy is much more efficiently achieved by reducing the timestep size. If the solution is not converging within each timestep, which is required to maintain a conservative discretization, it may signify that the timestep is too large for good transient accuracy. Decreasing the timestep size typically makes convergence within a timestep easier to achieve, and also improves the transient accuracy of the solution. Actual behavior is, of course, problem dependent. For your particular application, you need to make a compromise between timestep size, number of coefficient iterations and residual tolerance to get the most cost effective solution.

A value of 3 iterations per timestep should be sufficient for most single phase simulations, and values higher than 5 are unlikely to improve accuracy. In multiphase cases, the default value of 10 iterations per timestep may be more appropriate. Initially, you should use these values and then adjust the timesteps to control the balance between accuracy and solution time.

Long transient simulations can obviously end up being very computationally expensive. It is easy to define a problem that could potentially take weeks to compute, even with the coupled solution capability of Ansys CFX. It is very important, therefore, to balance the need for accuracy against run-time cost when performing these types of analyses.

- Timesteps: Timesteps List (p. 569)
- Steady State and Transient Flows (p. 41)

16.4.2.7. Transient Scheme

The transient scheme defines the discretization algorithm for the transient term.

The available options are:

16.4.2.7.1. First Order Backward Euler
16.4.2.7.2. Second Order Backward Euler
16.4.2.7.3. High Resolution
16.4.2.7.4. Harmonic Balance
16.4.2.7.5. Bounded Harmonic Balance
16.4.2.7.6. None

For related theory, see Transient Term in the *CFX-Solver Theory Guide*.

16.4.2.7.1. First Order Backward Euler

The `First Order Backward Euler` scheme is an implicit time-stepping scheme that is first-order accurate. Its behavior is analogous to the `Upwind` differencing scheme for advection terms, and suffers from similar numerical diffusion. Although it is useful for initial studies, its use is not recommended for production runs except for turbulence equations.

16.4.2.7.2. Second Order Backward Euler

The **Second Order Backward Euler** scheme is also an implicit time-stepping scheme, but is second-order accurate, and is the default in Ansys CFX. It is applicable for constant and variable timestep sizes. Like second-order advection schemes, however, it is not monotonic and is therefore inappropriate for some quantities that must remain bounded, such as turbulence quantities and volume fractions. When running the **Second Order Backward Euler** scheme, the transient scheme for turbulence equations will remain First Order, and the transient scheme for volume fraction equations will be set to a bounded second-order scheme, similar to the High Resolution scheme for advection. This scheme is generally recommended for most transient runs.

16.4.2.7.2.1. Timestep Initialization

When performing a transient run using the **Second Order Backward Euler** scheme, there are three options for initializing the solution within a timestep:

- **Previous Timestep:** The solution from the end of the previous timestep is used.
- **Extrapolation:** The solutions from the previous timesteps are extrapolated to the new timestep.
- **Automatic:** This option is a blend of the **Extrapolation** and **Previous Timestep** options:
  - If the local Courant number is below the specified **Lower Courant Number** (5 by default), then **Extrapolation** is used.
  - If the local Courant number is above the specified **Upper Courant Number** (10 by default), then **Previous Timestep** is used.

  For compressible cases, the local Courant number used is the so called "acoustic" Courant number, which is based on the speed of sound. For more details, see Courant Number in the *CFX-Solver Theory Guide*.

16.4.2.7.3. High Resolution

The High Resolution transient scheme uses the second order backward Euler scheme wherever and whenever possible and reverts to the first order backward Euler scheme when required to maintain a bounded solution.

16.4.2.7.4. Harmonic Balance

This option is available only for transient blade row cases that use the `Harmonic Balance` transient method.

16.4.2.7.5. Bounded Harmonic Balance

This option is available only for transient blade row cases that use the `Harmonic Balance` transient method, and is intended for solving variables that should always have positive values, for example, variables for turbulence kinetic energy or mass fraction.

16.4.2.7.6. None

This option is available only for transient blade row cases that use the `Harmonic Balance` transient method. It suppresses the transient term contribution of the specific equation.

16.5. Advection Scheme Selection

This section describes the **Basic Settings** tab in the **Solver Control** panel in Ansys CFX-Pre and how to choose these settings for your simulation.

You can select to use `Upwind`, `High Resolution`, or specify a blend factor to blend between first and second order advection schemes to calculate the advection terms in the discrete finite volume equations. Different schemes can be applied to each equation class by setting the **Equation Class Settings** in CFX-Pre.

The **CDS Blending** settings are available for the DES and SAS turbulence models only. They control the Courant-number-based blending between the specified advection scheme and the central difference scheme (CDS).

Additional information on the advection schemes used in Ansys CFX is available; for details, see Advection Term in the *CFX-Solver Theory Guide*.

16.5.1. Upwind

With this setting, advection terms are first-order accurate. This is equivalent to specifying a blend factor of 0.0. This setting gives the most robust performance of the CFX-Solver but suffers from numerical diffusion. Using this advection scheme is not recommended to obtain final results (except for the turbulence equations).

16.5.2. High Resolution

With this setting, the blend factor values vary throughout the domain based on the local solution field in order to enforce a boundedness criterion. In flow regions with low variable gradients, the

blend factor will be close to 1.0 for accuracy. In areas where the gradients change sharply, the blend factor will be closer to 0.0 to prevent overshoots and undershoots and maintain robustness.

Note that for vector quantities it is the components that are bounded between 0 and 1. Therefore, the magnitude of β for a vector quantity can be as large as $\sqrt{3}$.

16.5.3. Specified Blend Factor

This selection enables you to set a blend factor between 0.0 and 1.0 for the advection scheme. A value of 0.0 is equivalent to using the first order advection scheme and is the most robust option. A value of 1.0 uses second order differencing for the advection terms; this is *not* the same as the high resolution advection scheme. This setting is more accurate but less robust. Values between 0.0 and 1.0 blend first and second order differencing, with increased accuracy and reduced robustness as you approach 1.0. At the higher values, overshoots and undershoots can appear. At lower values, excessive diffusivity can occur.

Important:

If you are using `High Resolution` or a high blend factor value, it will be necessary to use a smaller timestep. It is recommended that you use a timestep of approximately 1/4 to 1/3 the physical timestep. For details, see Time Scale Control (p. 563).

You cannot use this advection scheme for the volume fraction equation in a multiphase simulation.

16.5.4. Central Difference

This option is available when using the large eddy simulation turbulence model and is recommended for these cases.

16.5.5. Advection Scheme for Turbulence Equations

Irrespective of the **Advection Scheme** setting, the advection scheme for the turbulence model equations depends on the **Turbulence Numerics** option:

- `First Order`

  The Upwind advection scheme is used.

- `High Resolution`

  The High Resolution advection scheme is used.

Note that **Turbulence Numerics** option also specifies the transient scheme for the turbulence model equations (see Turbulence Numerics in the *CFX-Pre User's Guide*). This can be overridden in CFX-Pre in the **Solver Control** details view on the **Equation Class Settings** tab.

16.5.6. Advection Schemes for Multiphase Volume Fractions

If you set the global advection scheme (that is, if you do not specify equation-class advection schemes) to upwind or high resolution, then the volume fraction equation will also be solved using upwind or

high resolution. This is differs from Ansys CFX-5.5.1 where upwind was always used for the volume fraction equations. However, if you set the global advection scheme to a specified blend, then the volume fraction equations will actually use the high resolution advection scheme but the maximum blend factor will be limited to the specified value instead of 1.

16.5.7. Comparisons to CFX-TASCflow

For users of CFX-TASCflow, the following settings approximate the common advection schemes used in CFX-TASCflow:

- The **Upwind** scheme is equivalent to `ISKEW=1` with `LPAC=F` in CFX-TASCflow.
- A **Specified Blend Factor** value of 0.75 is approximately equivalent to `ISKEW=3` with `LPAC=T` in CFX-TASCflow.
- A **Specified Blend Factor** value of 1.0 is approximately equivalent to `ISKEW=4` with `LPAC=T` in CFX-TASCflow.
- The **High Resolution** scheme is usually close to `ISKEW=3` with `LPAC=T`, but may be slightly better in some cases. It is always better for all equations other than the hydrodynamic system of equations. It is always smoother than `ISKEW=4` with `LPAC=T`, but never as accurate.

16.6. Advanced Options: Dynamic Model Control

Dynamic Model Control is designed to improve robustness during the first few iterations or timesteps of a simulation. Often when the CFX-Solver fails, it is during the first few iterations or timesteps. This can be caused by difficult initial guesses, a mismatch between the initial guess values and boundary condition specifications, and so on.

16.6.1. Global Dynamic Model Control

This option should generally be left selected. By itself, the only change made during the first few iterations is to turn off a product limiter for combusting/reacting flows. The product limiter is disabled for the first five iterations, after which it is turned back on. This behavior can be overridden by adding the following lines to the CCL in your CFX-Solver input file (which must first be converted to an editable text file). For details, see Editing the Command Language (CCL) File in the *CFX-Solver Manager User's Guide*.

```
DYNAMIC MODEL CONTROL:
 COMBUSTION CONTROL:
  Transition Iteration = 0
 END
END # DYNAMIC MODEL CONTROL
```

This would cause the product limiter to be on all the time.

Specific controls to the turbulence and hydrodynamic equations are described next; however, the **Global Dynamic Model Control** toggle must be selected for these to have any effect.

Note:

For each of the turbulence, combustion and hydro controls, the transition iteration is the iteration at which the change from the starting model to the model chosen for the simulation is made. The fifth iteration is often a reasonable choice. The default transition iteration is zero; that is, the starting model is not used unless you set a transition iteration.

16.6.2. Turbulence Control

By selecting the Turbulence Control option you cause the zero equation turbulence model to be used from the first iteration through to the **Transition Iteration** for eddy-viscosity-based turbulence models. The idea is to solve a simpler and more robust turbulence model for the first few iterations, so try a **Transition Iteration** value of 5.

You should use the Turbulence Control option when startup with the desired turbulence model is not possible.

16.6.3. Combustion Control

Combustion Control is described in Advanced Combustion Controls (p. 462).

16.6.4. Hydro Control

These settings become active from the first iteration through to the specified transition iteration. The default transition iteration is zero; that is, the settings are not used unless you set a transition iteration.

Sum Continuity Coefficients is only applicable to compressible flows. This forces/enhances diagonal dominance in the solution matrices for more robust startup. It has no effect on the steady solution, but should be turned off using the transition iteration after a few iterations as it would begin to retard convergence.

Free Surface Harmonic Averaging has the same effect as the **Harmonic Body Force Averaging** option (see below), but enables you to use harmonic averaging (more robust) during startup and then revert back to the default body force treatment after the transition iteration is reached.

16.7. Pressure Level Information

Sometimes a simulation does not have a pressure specification set through the boundary conditions. In this case, the solver automatically sets a pressure level at a reference location. By default, the reference value is zero, and the reference location is at the first finite volume number. There are two methods to override this default behavior if necessary.

The first (and recommended) method is to set a pressure level using the **Pressure Level Information** option on the **Solver Control** form in CFX-Pre. The Cartesian coordinates option enables you to specify an X, Y, Z location for the reference pressure location. The nearest node to this location is used as the pressure reference location. A value for the reference pressure can also be entered.

The **Compressible Transient Option** determines whether the pressure field is shifted to accelerate mass conservation for compressible transient simulations that have no pressure boundaries.

16.8. Interpolation Scheme

The following topics will be discussed:

- Pressure Interpolation Type (p. 577)
- Velocity Interpolation Type (p. 577)
- Shape Function Option (p. 577)

16.8.1. Pressure Interpolation Type

Pressure Interpolation Type refers to the method of interpolating nodal pressures to integration points for the pressure gradient term of the momentum equation. Trilinear is more accurate, especially for large pressure gradients, but may have robustness problems on highly stretched meshes. Trilinear is the default for buoyant flows, otherwise the default is linear-linear.

16.8.2. Velocity Interpolation Type

Velocity Interpolation Type refers to the method of interpolating nodal velocities to integration points for the velocity divergence term in the continuity equation. The default is trilinear, and should not need to be changed.

16.8.3. Shape Function Option

For tets, wedges, and pyramids, using trilinear shape functions for value interpolation may not be linearly exact because the trilinear integration point location may not lie at the geometric centroid of the face on which the integration point is located. In such situations, interpolation accuracy may be improved by using geometric shape functions. The default behavior is to use geometric shape functions for buoyant problems and the standard parametric shape functions otherwise.

16.9. Temperature Damping

Temperature Damping provides advanced control for temperature relaxation and becomes available when a transport equation for heat transfer is solved (thermal or total energy).

16.9.1. Option

The available options are:

- `Automatic`

  Tells the solver to use the option believed to be most appropriate for each fluid, depending on the physical setup. This is the default option.

- `None`

Disables temperature damping.

- `Temperature Damping`

  Enables temperature relaxation. Two parameters are available for controlling the amount of relaxation applied: Temperature-Damping Limit (p. 578) and Under-Relaxation Factor (p. 578).

16.9.2. Temperature-Damping Limit

This option is available when **Option** is set to `Temperature Damping`. The default value for this option is 0 K.

Temperature damping is equivalent to classical relaxation when the temperature damping limit is set to zero.

16.9.3. Under-Relaxation Factor

This option is available when **Option** is set to `Temperature Damping`. The default value for this option is 0.2.

The **Under-Relaxation Factor** is applied only if the absolute change in temperature is larger than the value specified as the Temperature-Damping Limit (p. 578).

16.10. Monitoring and Obtaining Convergence

There are a variety of opinions on how to judge convergence. The level of convergence required depends on the purpose of the simulation (whether qualitative or quantitative results are required) and the details of model. In general, you need to consider residual size (both RMS and MAX) and overall flow balances (conservation). Some cases, such as those with transient convergence behavior, require other considerations.

When quantitative accuracy is required, it is highly recommended that you test sensitivity of relative quantities (derived quantities) to the convergence target. These can be monitored during the run in the CFX-Solver Manager using monitor points.

Important:

You should always check whether your run stopped because your convergence criteria were met or because the maximum number of iterations was reached.

The following topics will be discussed:

16.10.1. Residuals

16.10.2. Using Interrupt Control in Cases with Transient Convergence Behavior

16.10.3. Global Balances and Integrated Quantities

16.10.4. Convergence Rate

16.10.5. Problems with Convergence

16.10.1. Residuals

The residual is a measure of the local imbalance of each conservative control volume equation. It is the most important measure of convergence as it relates directly to whether the equations have been solved accurately. Additional information on how the residual is calculated is available. For details, see Linear Equation Solution in the *CFX-Solver Theory Guide*.

CFX uses normalized residuals to judge convergence. For details, see Residual Normalization Procedure in the *CFX-Solver Theory Guide*. By normalizing the residuals, you are presented with a relatively consistent means of judging convergence. The normalized residual is used to automatically stop the CFX-Solver run when the specified target level is reached.

In transient simulations, the same target residual is used to control the termination of the coefficient iterations, except that the residual now includes a contribution from the transient term. If the maximum number of coefficient iterations per time step has been set to a large value, such as 20, then when the coefficient iteration residuals are below the target residual, the coefficient iterations will stop. This method can be used to cause the CFX-Solver to run with many coefficient iterations early in the run (when residuals start relatively high) and fewer coefficient iterations later in the run.

16.10.1.1. Residual Type and Target Levels

To assess when convergence is reached, you can select a residual type, either MAX (maximum) or RMS (root mean square), and specify a target value.

Note that the residual level for turbulence transport equations (for example, k, ε, Reynolds stress components, and turbulent heat flux components) are not included by the solver in deciding when convergence is reached.

Note also that for multiphase simulations, the convergence criterion for the volume fraction equations is a factor of 10 higher than the specified target residual.

16.10.1.1.1. RMS Residual Level

Although the required convergence level depends on the model and on your requirements, the following guidelines regarding RMS residual levels may be helpful.

- Values larger than 1e-4 may be sufficient to obtain a qualitative understanding of the flow field.
- 1e-4 is relatively loose convergence, but may be sufficient for many engineering applications. The default target RMS residual value is 1e-4.
- 1e-5 is good convergence, and usually sufficient for most engineering applications.
- 1e-6 or lower is very tight convergence, and occasionally required for geometrically sensitive problems. It is often not possible to achieve this level of convergence, particularly when using a single precision solver.

16.10.1.1.2. MAX Residual Level

MAX residuals are typically 10 times larger than the RMS residual; The above guidelines for RMS residuals can also be applied to MAX residuals, with the targets increased by a factor of 10.

Sometimes, however, the MAX residuals are much larger (for example, a factor of 100) than the RMS residuals. In this situation, it is very likely that the region of high MAX residuals is isolated to a very small area of the flow, typically where some unstable flow situation exists (for example, a separation or re-attachment point, and so on). It may be the case that this small area of unstable flow / lack of tight convergence of the MAX residuals do not affect the overall prediction. To verify that the solution is acceptable, you should verify that the variation of relevant quantities (for example, lift, drag forces, efficiency of the device, and so on) is small.

16.10.2. Using Interrupt Control in Cases with Transient Convergence Behavior

Some steady-state or transient cases will not converge to the target residual levels recommended in Residual Type and Target Levels (p. 579). In some cases, the lack of convergence can be attributed to inherent unsteadiness in the solution. Such cases are considered to exhibit transient convergence behavior.

If you expect your case to exhibit transient convergence behavior, especially periodic behavior, you should consider setting interrupt control conditions. For such cases, you can stop the run based on trends in monitored quantities. For example, you could make an interrupt control condition that stops a run when all of the following criteria are met:

- At least 100 iterations have passed.
- The arithmetic averages of a monitor of efficiency over two intervals of different size, 25 and 50 iterations, are within a certain relative tolerance, 1.0e-04.
- The coefficient of variation (the ratio of the standard deviation to the mean) of a monitor of efficiency over an interval of 50 iterations is below a certain value, 1.0e-04.

Note:

An interrupt control condition cannot be computed and has no effect until all of the monitor statistics that are involved in its definition can be computed. If you set a minimum number of iterations (100 in this case), then that number should be at least the size of the interval required for all statistics to be computed (50 iterations in this case).

In order to set up this interrupt control condition in CFX-Pre, you could follow this procedure:

1. Create an expression for efficiency named `Eff`.

   For details, see Creating an Expression in the *CFX-Pre User's Guide*.

2. Create a monitor named `MP Eff 25`.

   For details, see Setting up Monitors in the *CFX-Pre User's Guide*.

3. Configure the following setting(s):

Setting	Value
MP Eff 25 > Option	Expression
MP Eff 25 > Expression Value	Eff

Setting	Value
MP Eff 25 > Monitor Statistics	(Selected) [a]
MP Eff 25 > Monitor Statistics > Statistics List	Arithmetic Average
MP Eff 25 > Monitor Statistics > Interval Definition > Number of Iterations	25

a. Monitor statistics enable you to evaluate statistical quantities for an expression value over a moving interval. For details, see [Monitor Name]: Monitor Statistics in the *CFX-Pre User's Guide*.

4. Create a monitor named `MP Eff 50`.

5. Configure the following setting(s):

Setting	Value
MP Eff 50 > Option	Expression
MP Eff 50 > Expression Value	Eff
MP Eff 50 > Monitor Statistics	(Selected)
MP Eff 50 > Monitor Statistics > Statistics List	Arithmetic Average, Standard Deviation
MP Eff 50 > Monitor Statistics > Interval Definition > Number of Iterations	50

6. Create an expression named `Avg Eff 25` with the following definition:

```
probe(Expression Value.Arithmetic Average)@MP Eff 25
```

This expression calculates the arithmetic average for `MP Eff 25` over the current interval (25 iterations). For details, see Using Monitors in CEL in the *CFX-Pre User's Guide*.

7. Create an expression named `Avg Eff 50` with the following definition:

```
probe(Expression Value.Arithmetic Average)@MP Eff 50
```

8. Create an expression named `StdDev Eff 50` with the following definition:

```
probe(Expression Value.Standard Deviation)@MP Eff 50
```

9. Create the interrupt control condition by accepting the default name and using the following logical expression:

```
aitern>100 && abs((Avg Eff 25-Avg Eff 50)/Avg Eff 25)<0.0001 && abs(StdDev Eff 50/Avg Eff 50)<0.0001
```

The interrupt control settings are described in Interrupt Control in the *CFX-Pre User's Guide*.

16.10.3. Global Balances and Integrated Quantities

At the end of each run, a summary of global balances is reported in the CFX-Solver Output file. For details, see Global Conservation Statistics in the *CFX-Solver Manager User's Guide*. You can also monitor Flows, Forces and general CEL expressions during the run. In most cases, when the residuals have converged sufficiently, the global balances will be met. However, if there is a process whose time scale is very large relative to other physical processes, it is possible to have converged residuals before the global balances are met. An example might be the dissipation of heat from a cup of coffee into a large room.

Flow balances are evaluated while assembling the conservation equations, and are therefore based on the solution from the previous timestep. For transient runs, if convergence within a timestep is achieved in one coefficient iteration, this can lead to unexpected diagnostics for the accumulation term. By default, the initial guess within a timestep is taken from the end of the previous timestep. For the First Order Backward Euler transient scheme, this leads to an accumulation term of zero, and for the Second Order Backward Euler scheme, the accumulation term has the opposite sign from expectation. However, if the Second Order Backward Euler scheme is used with extrapolation as the initialization option, then the accumulation term is reasonable after one coefficient. For details about the initialization setting, see Transient Scheme (p. 571).

16.10.3.1. Positive and Negative Domain Source Totals

A quantity may be produced in one region of a domain, and destroyed in another region of a domain. For example, an intermediate combustion species, may have a small overall source resulting from the difference of significant production and destruction. In order to handle this case appropriately, the positive and negative sources are accumulated separately.

In case a transport equation has production and destruction terms that contribute simultaneously at the same location, details of the implementation determine whether the positive and negative contributions are balanced separately, or whether just the local net source is balanced. In the latter case, the effective local source values are spread into the positive and negative domain source totals.

16.10.3.2. Conservation Target

To ensure the global balances are met, you may additionally apply a target imbalance for the conservation equations (that is, a global balance criterion). In such a case, the solver will only stop before the maximum number of iterations if the residual criteria are met and global balances are also met. The value you specify is the fractional imbalance, where the default value is 0.01 (that is, 1%). The global imbalances for the hydrodynamic equations should at least be less than 1% before you can consider a solution converged.

16.10.4. Convergence Rate

A convergence rate can be defined by:

$$\text{convergence rate} = \frac{R_n}{R_{n-1}} \tag{16.3}$$

where R_n is the Normalized Residual at iteration n, and R_{n-1} is the Normalized Residual at an earlier iteration. With two exceptions, the convergence rate is evaluated using the current and previous iter-

ations. The first exception corresponds to evaluating the residuals during the first coefficient iteration for a timestep in transient simulations. In this case, the rate is evaluated using the current residuals and the residuals from the first iteration from the previous timestep. The second exception corresponds to evaluating the residuals of the mesh displacement equations for both steady and transient simulations. In this case, the rate is evaluated using current residuals and the residuals from the first iteration from the previous solution of the displacement equations. In each of these cases, residual reduction values that are less than unity indicate convergence towards a steady state.

A residual reduction rate of 0.95 or smaller is considered typical for most situations, while a rate of 0.85 or smaller is considered to be very good. If your convergence behavior is slower than this (that is, rates are larger than 0.95), but is smooth, you should try increasing the timestep. Once the residuals are dropping monotonically, the timestep can often be safely increased to the characteristic time scale or greater in order to maximize the convergence rate. If there is no improvement, then the source of the convergence problem is probably not the timestep specification. For details, see Initial Condition Modeling (p. 161).

16.10.5. Problems with Convergence

This section describes how to solve two general classes of convergence problems:

16.10.5.1. Start-up Problems

16.10.5.2. Later Problems

16.10.5.1. Start-up Problems

If you are having problems getting your solution started, it will often be due to a poor initial guess. For details, see the following:

- Initialization Advice (p. 173)
- Recommended Configurations of Boundary Conditions (p. 114)

For some cases, in particular for compressible flow cases, you will need to first solve the case using simpler and more robust models and then use this solution as the initial guess for the more complex simulation. Below is a list of simulation attributes:

- Liquids and constant property materials are more robust than gases (in other words, incompressible flow is more robust than compressible flow).
- Laminar flow is more robust than turbulent flow with the Zero Equation Model, which in turn is more robust than the $k-\varepsilon$ and other two-equation turbulence models, which in turn are more robust than Reynolds stress models. The zero equation model should not be used to obtain final results.
- Pressure specified Openings may be more robust than Static Pressure Outlets.
- The **1st Order** advection scheme or a **Specified Blend = 0** is more robust than the **High Resolution** scheme, which is more robust than a **Specified Blend = 1**. Smaller values of the Blend Factor are more robust than larger ones. Final results should use a Specified Blend of at least 0.75 or the High Resolution scheme. It is also possible that using a Blend Factor of 1.0 will not produce a converged solution no matter what you do. In this case, gradually increasing values

of the Blend Factor could be used to get a more accurate solution than the 1st order option. If the flow includes shocks, the High Resolution scheme should be used.

- For transonic and supersonic flows, the most difficult flow regime is around Mach 1. If this is your flow regime of interest, you should first try a flow velocity lower or higher than Mach 1 and see how that converges. If this is successful, you can then modify your problem specification to your requirements and use the previous result as your initial value field.
- Smaller physical timesteps are more robust than larger ones.
- An Isothermal simulation is more robust than modeling heat transfer. The Thermal Energy model is more robust than the Total Energy Model.
- Velocity or mass specified boundary conditions are more robust than pressure specified boundary conditions. A Static pressure boundary is more robust than a total pressure boundary.

16.10.5.2. Later Problems

If you have problems with convergence, you should find the source of the problem rather than taking the results as they are. There are many factors that may lead to poor convergence, including poor mesh quality, improper boundary condition selection and timestep selection to name a few. If you are unable to diagnose the source of your convergence difficulties, contact your technical support representative for advice.

When you are having problems converging, try to determine whether the problem is local or global. Compare the RMS and MAX residuals of the equations having difficulty. If the MAX residual is more than one order of magnitude larger than your RMS residual, it usually indicates that the problem is concentrated to a local region.

If it is a locally high residual, identifying the location of the MAX residual will help in diagnosing the problem. Typically the location of the MAX residual of the momentum equations is the most useful to identify. This may be done by reading the location of the MAX residual from the solver summary at the end of your CFX-Solver Output file, or by setting the Expert Parameter **output eq residuals = true**. For details, see CFX-Solver Expert Control Parameters (p. 603).

If the MAX residual is far downstream of your region of interest and far from an outlet boundary, you may determine that it has no effect on your solution. If you write the equation residuals out to your RES file, then it may be useful to create an isovolume equal to your MAX residual criteria and to verify the residuals are low enough in your region of interest.

Once you have determined the location of your high residuals, you will want to determine the source of the local problem and fix it. Because CFD simulations are varied, a comprehensive list is not possible, but the following short list of common problems and solutions should be helpful. If you still cannot resolve your problem, contact your Ansys CFX technical support representative for advice.

Table 16.1: Convergence Problems Due to Local Effects

Local Problem	Description and Solution
Poor grid quality	Small angles or high aspect ratios can lead to round-off errors in the solver. Try to improve grid quality in the problem area.

Local Problem	Description and Solution

Free shear layer or wake	These often cause the convergence to stall due to transient effects. In most cases, the transients should be captured by the turbulence model and reflected as a mean velocity and turbulence intensity. A small timestep will resolve the transients, therefore causing the convergence problems. A larger timestep should take care of the problem. In some situations, it may also help to apply additional under-relaxation to the advection scheme gradients and blend factors. This can be done by adding the following text to the ADVECTION SCHEME CCL:
	`ADVECTION SCHEME:`
	`Option = High Resolution`
	`Gradient Relaxation = 0.1`
	`Blend Factor Relaxation = 0.1`
	`END`
	`Blend Factor Relaxation` is applicable only for the high resolution scheme, while `Gradient Relaxation` is applicable for both the specified blend and the high resolution schemes. The default values of these parameters are 0.25 for steady-state simulations and 0.5 for transient simulations

MAX residual adjacent to a shock	The solution field can "bounce around" near discontinuities like shocks. Shocks can also be susceptible to a transverse instability called the carbuncle effect, particularly if the mesh is finer in the transverse direction than the flow direction. To resolve these issues, activate **High Speed Numerics** on the **Advanced** panel of the **Solver Control** tab.
MAX residual adjacent to a Stage Domain Interface	A large timestep can cause convergence problems at a Stage Interface. Reduce your timestep by a factor of 2 to 10. For closely coupled components, the **Constant Total Pressure** option should be considered. Increasing the pressure profile decay above its default of 0.05 may help robustness by adding stiffness to the downstream pressure profile, but can also reduce the natural circumferential pressure variation.
MAX residual adjacent to a flow boundary	Check that the boundary condition is sensible. Try different options for the boundary specification. If you can specify a profile (by equation or data interpolation), it may help. If

Local Problem	Description and Solution
	you are using a profile, check to see that it is correct and sensible.

Table 16.2: Convergence Problems Due to Global Effects

Global Problem	Description and Solution
Large time scale effect	If the characteristic time scale is not simply the advection time of the problem, there may be transient effects holding up convergence. Heat transfer or combustion processes may take a long time to convect through the domain or settle out. There may also be vortices caused by the initial guess, which take longer to move through the entire solution domain. In these cases, a larger timestep may be needed to push things through initially, followed by a smaller timestep to ensure convergence on the small time scale physics. If the large timestep results in solver instability, then a small time scale should be used and more iterations may be required.
Turbulence levels	Sometimes the levels of turbulence in the domain can affect convergence. If the level of turbulence is non-physically too low, then the flow might be "too thin" and transient flow effects may be dominating. Conversely if the level of turbulence is non-physically too high then the flow might be "too thick" and cause unrealistic pressure changes in the domain. It is wise to look at the Eddy Viscosity and compare it to the dynamic (molecular) viscosity. Typically the Eddy Viscosity is of the order of 1000 times the dynamic viscosity, for a fully turbulent flow.
Turbulence model selection	When choosing a turbulence model, care must be taken to use a mesh appropriate for the given model. Running a given turbulence model on an inappropriate mesh can cause convergence to stall.
Advection scheme	The 2nd Order High Resolution advection scheme has the desirable property of giving 2nd order accurate gradient resolution while keeping solution variables physically bounded. However, may cause convergence problems for some cases due to the nonlinearity of the Beta value. If you are running High Res and are having convergence difficulty, try reducing your timestep. If you still have problems converging, try switching to a Specified Blend Factor of 0.75 and gradually increasing the Blend Factor to as close to 1.0 as possible.

16.11. Solver Issues

The following topics will be discussed:

- Robustness and Accuracy (p. 587)
- Linear Solver Failure (p. 587)

16.11.1. Robustness and Accuracy

Some comment has already been made about the relative merits of robustness and accuracy, and the need to solve more complex problems by gradually increasing the complexity of the model, but they are worth reiterating here. Generally speaking, higher order discretizations are more accurate but less robust than lower order ones. Increasing the order of discretization by increasing the Blend Factor should be approached in the same way as increasing the complexity of the physical model, that is, gradually, and by starting from a more robust solution.

16.11.2. Linear Solver Failure

This is the most catastrophic thing that can go wrong during the solution phase. A fatal failure of the linear solver could look like this:

```
================================================================================
 TIME STEP =   6   SIMULATION TIME = 3.00E+11   CPU SECONDS = 1.05E+02
 ------------------------------------------------------------------------
 | Equation | Rate | RMS Res | Max Res | Location |  Linear Solution  |
 +----------+------+---------+---------+----------+-------------------+
 | U - Mom  | 1.36 | 3.7E-02 | 3.5E-01 |       1  |       NaN  *      |
 | V - Mom  | 1.37 | 3.8E-02 | 3.5E-01 |       1  |       NaN  *      |
 | W - Mom  | 0.75 | 1.2E-01 | 1.2E+00 |      26  |       NaN  *      |
 | P - Mass | 1.74 | 2.0E-02 | 6.9E-01 |    1248  |  8.3  Nan  *      |
 +----------+------+---------+---------+----------+-------------------
 +----------------------------------------------------------------------+
 | ERROR #004100018 has occurred in subroutine FINMES.                  |
 | Message:                                                             |
 | Fatal overflow in linear solver.                                     |
 +----------------------------------------------------------------------+
```

After which the CFX-Solver terminates. The "*" means that there has been a floating point exception in the linear solver, usually a floating point overflow.

Sometimes a linear solver failure can be less dramatic, such as

```
================================================================================
TIME STEP =   2   SIMULATION TIME = 1.00E+11   CPU SECONDS = 2.52E+01
------------------------------------------------------------------------
| Equation | Rate | RMS Res | Max Res | Location |  Linear Solution  |
+----------+------+---------+---------+----------+-------------------+
| U - Mom  | 0.00 | 3.7E-05 | 4.9E-04 |      48  |       2.6E+01  F  |
| V - Mom  | 0.00 | 3.7E-05 | 4.9E-04 |      38  |       2.6E+01  F  |
| W - Mom  | 0.00 | 2.8E-04 | 7.1E-03 |      37  |       8.4E+01  F  |
| P - Mass | 0.00 | 1.3E-07 | 3.8E-06 |      37  |  8.3  1.1E+02  F  |
+----------+------+---------+---------+----------+-------------------+
| H-Energy | 0.00 | 2.6E-02 | 1.9E-01 |     748  | 10.2  5.0E-04  OK |
+----------+------+---------+---------+----------+-------------------+
| K-TurbKE | 0.00 | 0.0E+00 | 5.9E-34 |    1200  |  5.6  0.0E+00  OK |
+----------+------+---------+---------+----------+-------------------+
| E-Diss.K | 0.00 | 0.0E+00 | 5.7E-34 |    1200  |  5.6  0.0E+00  OK |
+----------+------+---------+---------+----------+-------------------+
```

This time the CFX-Solver continues with mixed results (that is, sometimes it will recover and converge, sometimes not). The "F" in the last column means that the linear solver failed to reduce the residuals of the linearized equations (the solution was diverging).

When the linear solver fails, it can mean that non-physical boundary conditions have been applied, or that the initial values were inappropriately set. For details, see Boundary Condition Modeling (p. 111). A good set of initial conditions can be generated by solving a simpler flow. For instance, assume that the flow you want to model is turbulent and compressible. You want to model it using the $k-\varepsilon$ model, with the most accurate discretization, with several pressure specified opening boundaries and a complex, curved geometry. There is no good initial values guess, and the linear solver might fail no matter what you try. However, if you instead modeled laminar flow, with the most robust discretization, and changed the boundary conditions to be velocity specified inflows with only one pressure-specified opening, the solution will converge with almost any initial value you choose. You could then use this converged result as the much better initial value for the real problem of interest.

16.12. How CEL Interacts with the CFX-Solver

At various points in the flow solver operation, the calculation requires information such as the rate of flow at an inlet boundary, or the viscosity in some part of the fluid. This dependence can be specified in a simple manner, but more complex dependencies can be developed by using CEL.

Consider the case of flow through a pipe. You may want to set a velocity profile at the inlet, which will typically be greatest at the center. The profile can be expressed mathematically in terms of a function of x, y, and z, which are Ansys CFX system variables known to the CFX-Solver. The mathematical definition of the profile is called an expression, and the value of this expression is a term that is input to the flow solver. You can define many terms in Ansys CFX by way of expressions: fluid properties, subdomain sources, boundary conditions, and initial values are all Ansys CFX terms that can be described using expressions.

A key feature of CEL is that it is used dynamically by the CFX-Solver. For example, you can set viscosity to be dependent on temperature using CEL, and the current heat transfer CFD solution will be computed for this relationship.

16.13. Best Practice Guides

The following best practice guides are available:

- CFX Best Practices Guide for Numerical Accuracy
- CFX Best Practices Guide for Cavitation
- CFX Best Practices Guide for Combustion
- CFX Best Practices Guide for HVAC
- CFX Best Practices Guide for Multiphase
- CFX Best Practices Guide for Turbomachinery

Chapter 17: Using the Solver in Parallel

The parallel implementation of the CFX-Solver is based on the *Single-Program-Multiple-Data* (SPMD) model. This model runs identical versions of the code on one or more processors.

The overall parallel run procedure is divided into two steps:

1. A *partitioning* step, where the mesh is divided into a number of different segments, or partitions.

2. A *running* step, where the mesh partitions are solved using separate processes (a *leader* and one or more *follower* processes), often on different machines, with each process working on its own partition.

The CFX-Solver is designed so that all of the numerically intensive tasks can be performed in parallel. Administrative tasks, such as simulation control and user interaction, as well as the input/output phases of a parallel run, are performed serially by the leader process. This approach guarantees good parallel performance and scalability of the parallel code, as well as ensuring that the input/output files are like those of a sequential run.

Communication between processes during a parallel run is performed using the MPI message-passing libraries. Platform-specific versions of MPI are available. In all cases, the processes of a parallel run are distributed among the processors in the specified pool of hosts.

The following limitation exists for combining hardware in a parallel run:

- Intel MPI can only be used on a homogeneous network of the supported platforms.

This chapter describes the ideas behind parallel processing in Ansys CFX and provides some advice on using it effectively:

17.1. Partitioning

17.2. Setup for Parallel Runs

17.3. Message Passing Interface (MPI) for Parallel

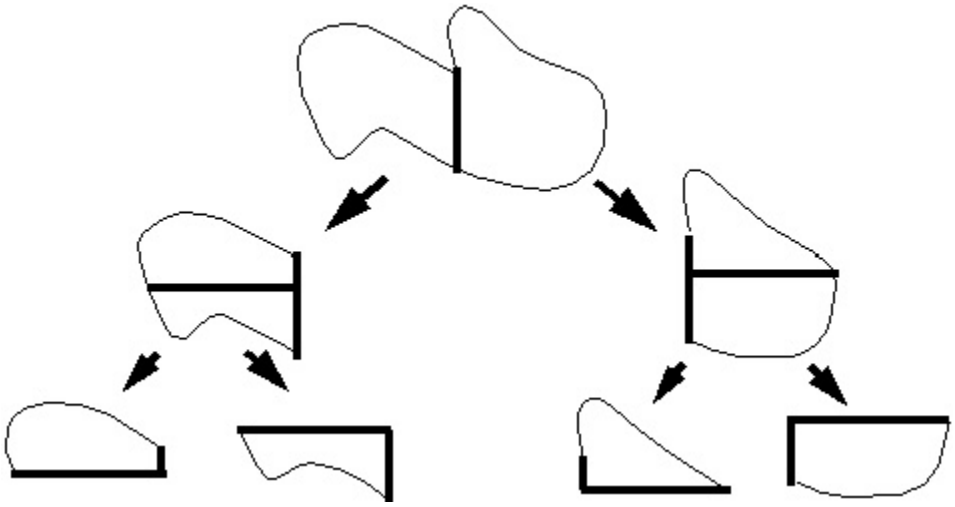
17.4. Advice on Using Ansys CFX in Parallel

More detailed instructions on how to set up a parallel run are available in Parallel Run in the *CFX-Solver Manager User's Guide*.

17.1. Partitioning

Partitioning is the process of dividing the mesh into a number of 'partitions' each of which may be solved on a separate processor. Several partitioning methods have been developed, but most are based on recursive bisection and differ only in the bisection step. The original mesh is first decomposed into two meshes of approximately equal size. The decomposition is then repeated recursively until the required number of partitions is obtained. The following diagram shows this process.

Figure 17.1: The Partitioning Process



For information about setting up partitioning, see Partitioner Tab in the *CFX-Solver Manager User's Guide*.

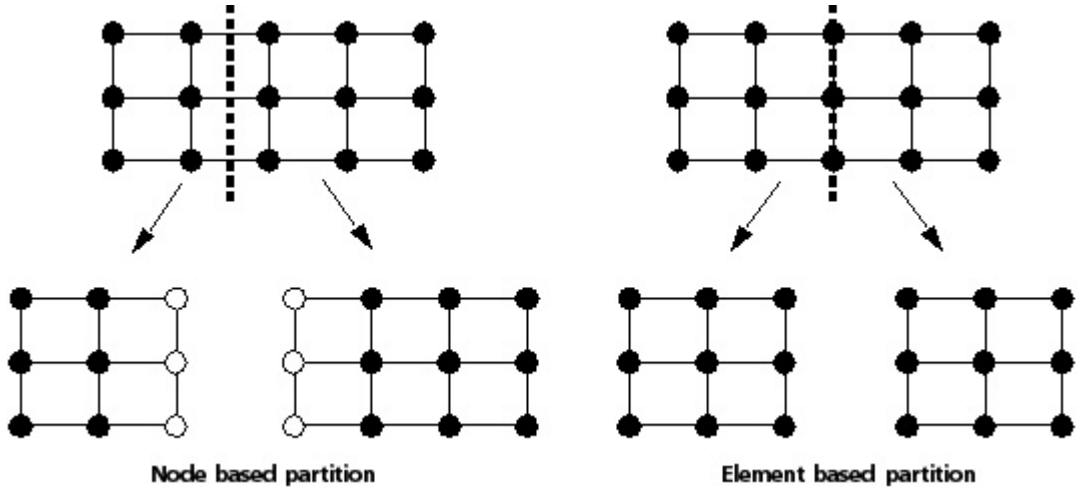
For information about viewing the mesh partitions after running the CFX-Solver in Partition Only mode, see CFX Partition File in the *CFX-Solver Manager User's Guide*.

For information about optimizing mesh partitioning, see Optimizing Mesh Partitioning (p. 595).

17.1.1. Node-based and Element-based Partitioning

Any three-dimensional mesh can be partitioned using either node-based or element-based partitioning. *Node-based partitioning* divides the mesh across element faces; that is, between nodal locations. *Element-based partitioning* divides the mesh along element faces without dividing elements themselves; that is, at nodal locations.

Figure 17.2: Node-based and Element-based Partitioning



CFX uses node-based partitioning because this is consistent with the node-based linear solver. The following partitioners are available:

- MeTiS

- Recursive Coordinate Bisection
- Optimized Recursive Coordinate Bisection
- Simple Assignment
- User Defined Direction
- Directional Recursive Coordinate Bisection
- Junction Box
- Radial
- Circumferential.

Not all nodes in a given case require the same amount of computation effort. For example, a node on a GGI boundary requires a larger assembly effort within the solver than a node that is not on a GGI boundary. To take account of this, the CFX-Solver does not weight all nodes equally during partitioning—a partition with a lot of nodes on GGI interfaces will generally be assigned fewer nodes in total than one that does not have many nodes on GGI interfaces. There is an expert control parameter, `ggi vertex weighting value`, that controls the weighting applied to GGI vertices during partitioning. For more information, see Expert Control Parameters (p. 603).

17.1.2. Multilevel Graph Partitioning Software - MeTiS

The public domain partitioning package MeTiS (Karypis and Kumar, 1996) uses the Multilevel Graph Partitioning Algorithm. The basic idea behind this algorithm is as follows: a graph is built containing the topology information of the mesh to be partitioned. This graph is first coarsened down to a few hundred vertices. A bisection of the resulting much coarser graph is calculated, and then the resulting partitions are projected back onto the original graph, by consecutively refining the partitions.

The MeTiS partitioner uses a fast algorithm that creates very efficient partitions. The whole process is fully automatic. However, it is unable to take advantage of coordinate direction alignment, and a substantial amount of dynamically allocated memory is required to run it (approximately 250 MB for a mesh containing 1.0E+06 nodes). If memory is a limitation, comparable results can be obtained with Optimized Recursive Coordinate Bisection. For details, see Optimized Recursive Coordinate Bisection (p. 592).

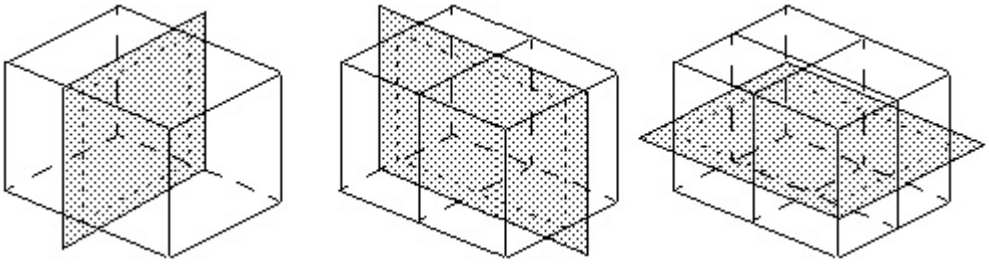
MeTiS is the default partitioner used by CFX.

The Multidomain Option setting can be set to partition the mesh for each domain separately (Independent Partitioning) or to partition the mesh ignoring domains (Coupled Partitioning). Note that only domains of the same physical type (fluid/porous and solid) are coupled together for coupled partitioning.

17.1.3. Recursive Coordinate Bisection

The Recursive Coordinate Bisection partitioning algorithm is very efficient with respect to CPU time and additional memory. The partitioning is based on the global coordinates of the mesh. Each step of the recursive bisection is performed in the coordinate direction with the largest dimension.

Figure 17.3: Recursive Coordinate Bisection



Like MeTiS, the Recursive Coordinate Bisection partitioner uses a fast, fully automatic algorithm, but it has only a small additional memory overhead. A particular disadvantage is that larger overlap regions can exist compared with the MeTiS partitioner, and each partition may contain separate parts.

17.1.4. Optimized Recursive Coordinate Bisection

This partitioning method is similar to Recursive Coordinate Bisection, but allows arbitrary directions for each partitioning step. The partitioning quality is comparable to the MeTiS algorithm. The method requires more CPU time to run, but is efficient with respect to memory. You may want to use this algorithm if some very large cases require too much memory for the MeTiS algorithm to function.

17.1.5. Simple Assignment

The Simple Assignment provides the fastest possible partitioning method that uses the smallest amount of memory. The vertices are assigned to the partitions simply corresponding to their vertex numbers. Assume a mesh with 1000 vertices that has to be partitioned into 10 partitions. The Simple Assignment produces the following partitions:

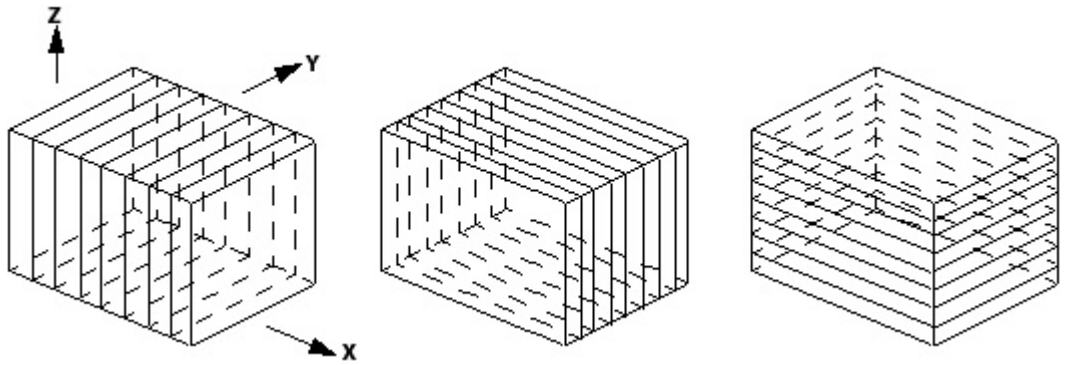
```
Partition 1:        Vertices 1-100
Partition 2:        Vertices 101-200
Partition 3:        Vertices 201-300
...
```

This is reasonable for hex meshes because their vertices are generally in topological order. However, this partitioning method can be arbitrarily bad for unstructured meshes and can result in huge overlapping regions.

17.1.6. User Specified Direction

The simplest partitioning algorithm is coordinate based in a direction whose vector components can be specified directly. This partitioning method uses the most efficient algorithm, and also has small additional memory overhead.

Figure 17.4: User Specified Direction



The partitioning process itself is partially controllable through the direction specification. However, in general, larger overlap regions occur compared with both the MeTiS and recursive coordinate bisection partitioners. This application is generally recommended only for special geometries.

17.1.7. Directional Recursive Coordinate Bisection

The Recursive Coordinate Bisection (p. 591) method recursively splits the domain into two parts using the x-, y-, or the z-coordinate direction (the choice depends on the biggest extension). This will geometrically lead to "cubic" partitions, but does not take into account the different mesh resolution in the three possible coordinate directions. Only the vertex coordinates are required for the partitioning. The graph (vertex/element connectivity) is not required, which makes this method very efficient.

In order to further improve the Recursive Coordinate Bisection method, there is the Optimized Recursive Coordinate Bisection (p. 592) method. Unlike the Recursive Bisection method, this method uses a graph to decide which coordinate direction has to be used for the bisection at each recursion level. The Directional Recursive Coordinate Bisection method is similar to the Optimized Recursive Coordinate Bisection, but is not limited to the three coordinate directions.

With respect to efficiency and partition quality (overlap region minimization), the methods can be ordered as follows, showing decreasing efficiency (increased CPU time and memory requirements) but better partitioning quality:

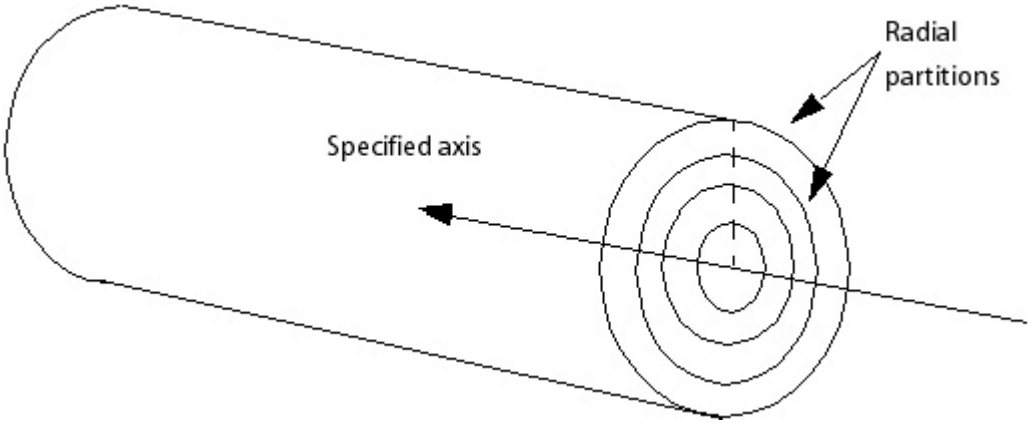
1. Recursive Coordinate Bisection
2. Optimized Recursive Coordinate Bisection
3. Directional Recursive Coordinate Bisection.

The best (but most expensive) partitioning method (Directional Recursive Coordinate Bisection) is still cheaper than Multilevel Graph Partitioning Software - MeTiS (p. 591), but generates partitions of similar quality.

17.1.8. Radial

Radial partitioning requires the specification of a rotation axis and partitions the mesh in the radial direction, according to the following diagram.

Figure 17.5: Radial Partitioning

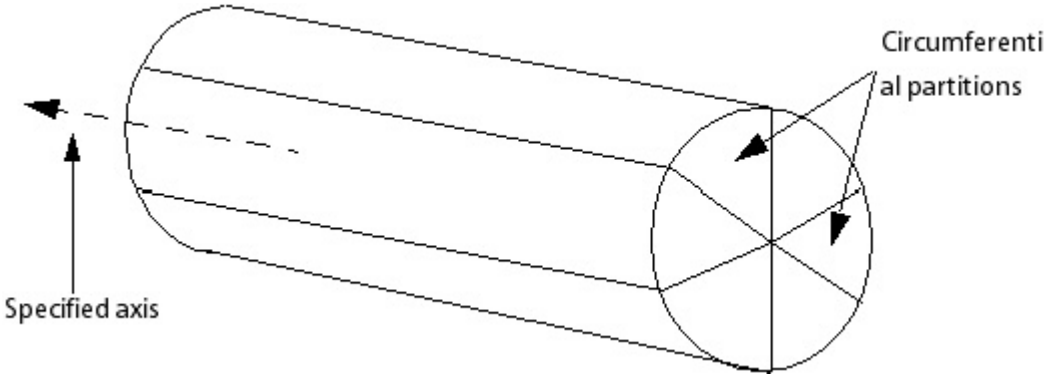


The same considerations apply for Radial partitioning as for User Specified direction partitioning.

17.1.9. Circumferential

Circumferential partitioning uses the same axis definition as the radial partitioning method and divides the problem in the circumferential direction, according to the diagram.

Figure 17.6: Circumferential Partitioning



The same considerations apply for circumferential partitioning as for user specified direction partitioning.

17.2. Setup for Parallel Runs

For details, see Configuring Ansys CFX Parallel in the *Ansys, Inc. Installation Guides*.

17.3. Message Passing Interface (MPI) for Parallel

MPI is available for Windows and UNIX systems as a parallel communication library.

Important:

MPI run modes cannot be used on a mixture of Windows and UNIX machines.

17.4. Advice on Using Ansys CFX in Parallel

The following topics will be discussed:

- Optimizing Mesh Partitioning (p. 595)
- Optimizing the Parallel Run (p. 596)
- Error Handling (p. 597)
- Measuring Parallel Performance (p. 601)

17.4.1. Optimizing Mesh Partitioning

Partitioning in Ansys CFX Parallel is a pre-processing step. In CFX-Solver Manager, you can choose one of several partitioning methods (see Partitioner Tab in the *CFX-Solver Manager User's Guide*), and you can specify the number of partitions (up to a maximum of 16384).

In general, you should try to follow these guidelines wherever possible:

Do not run small jobs in parallel

For tetrahedral meshes, you may want to use a minimum of 30,000 nodes per partition. For partitions smaller than this, you are unlikely to see any significant performance increase and may even see parallel slow down.

For hexahedral meshes, good parallel performance improvements are usually not seen until a minimum of 75,000 nodes per partition is reached.

These numbers are machine dependent and can be higher or lower. Dual CPU PCs usually give poorer performance due to lack of bandwidth in the bus. Essentially the two CPUs can demand more memory access than the memory bus can provide.

Always try to use the MeTiS partitioning method

MeTiS generates the best partitions with respect to the size of the overlap regions between the partitions, and is therefore the favored, and the default, partitioning method in CFX. Use one of the alternative partitioning methods only if MeTiS fails, or if the memory overheads are unacceptably high.

Use a sensible number of partitions

The partitioning of a mesh leads to the creation of overlap regions at the partition interfaces. These regions are responsible for communication and memory overhead during a parallel run. During partitioning, CFX prints partitioning diagnostic information to the CFX-Solver Manager text window and CFX-Solver Output file about partition overlaps. The percentage of overlap nodes to the total number of mesh nodes should ideally be less than 10% for efficient partitioning. Values greater than 20% will impair performance and are not recommended.

Store the partition file

As mentioned above, partitioning is a preprocessing step. By default, Ansys CFX performs mesh partitioning automatically in combination with a parallel run. The creation of a partition file is

therefore treated as an intermediate step, and is deleted at the end of a parallel run. However, if the machine that runs the leader process does not have enough memory to run the partitioner, or several parallel runs are performed with the same number of partitions, it is advisable to first store the partition file permanently with a partition-only run of CFX-Solver.

Select an appropriate partitioning mode

For multi-domain cases, you can set **Multidomain Option** on the **Partitioner** tab by selecting `Automatic`, `Independent Partitioning`, or `Coupled Partitioning`. For details, see Partitioner Tab in the *CFX-Solver Manager User's Guide*.

If the case does not involve particle transport, the `Automatic` option is the same as the `Coupled Partitioning` option; otherwise it is the same as the `Independent Partitioning` option.

`Coupled Partitioning` provides better partitions with respect to the size of overlap regions between partitions and thus exhibits improved parallel performance. However, this requires more memory for the partitioning run. The partitioning memory scales with the size of the largest domain for `Independent Partitioning` and with the total size of all domains for `Coupled Partitioning`. Furthermore, performance of particle transport calculations may be made worse when using `Coupled Partitioning`.

Perform partition smoothing

Partition smoothing attempts to minimize the surface area of partition boundaries by swapping vertices between partitions. Partition smoothing reduces communication overhead and improves solver robustness.

During partitioning, "Iso-Partition Connection Statistics" are written to the CFX-Solver Output file. These include:

- the number of smoothing sweeps performed
- the number of times that any vertices are swapped between partitions
- the number of vertices that have a certain percentage of their connections in the same partition

Vertices with a low percentage of neighbors in the same partition tend to reduce numerical stability. The smoothing algorithm performs successive sweeps until either the number of these poorly connected vertices has been minimized, or the maximum number of sweeps has been performed.

By default, the smoothing algorithm usually performs a sufficient number of sweeps to optimally smooth your partitions. If necessary, you can increase the maximum number of sweeps by increasing the value of **Partition Smoothing** > **Max. Smooth. Sweeps**. Decreasing the maximum number of sweeps is not recommended.

17.4.2. Optimizing the Parallel Run

The following is a list of tips to help you get the most from a parallel run using the CFX-Solver:

- For a single parallel run, try to use machines that have a similar level of performance.

- Avoid using swap space. Try to use machines that can perform their assigned tasks using their own physical memory.
- If you are using a network of workstations, use high speed Ethernet connections (1 Gbit is preferred to 100 Mbit).
- High-speed interconnects are supported by Intel MPI.
- Before starting a parallel job, check to see if your selected machines or processors can be used exclusively. If you have to share their use with others, this may increase the CPU static load.
- Performance may be limited by memory bandwidth rather than CPU speed on some hardware.
- Avoid assigning more than one process per core.
- Use a sensible number of partitions. There should always be a balance between the individual partition size and the number of partitions.

17.4.3. Error Handling

This section gives suggestions on what to do if you encounter problems during a parallel run of the CFX-Solver. Most failures occur during the initial phase of the run as a result of configuration problems.

17.4.3.1. Problems with Intel MPI

17.4.3.1.1. Semaphores and Shared-memory Segments

Semaphores are constructs used by Intel MPI to enable interprocess communication locally on a shared memory multiprocessor.

Each parallel process uses its own semaphore. Shared-memory access via semaphores is managed by the operating system kernel. Semaphores are created and used by a user-id, but managed by the operating system.

When an Ansys CFX Intel MPI parallel job is finished, these semaphores are, by default, deleted. If the job terminated in some abnormal fashion, for example one of the processes was killed, then the semaphores in use are not deleted and remain in an idle state. This situation consumes free semaphores on your system. There is a maximum limit on the number of semaphores that the operating system will permit. Once this limit is reached, no new semaphores can be created. Software usage may become unstable when the semaphore limit is reached.

17.4.3.1.2. Typical Problems When You Run Out of Semaphores

- You cannot start a new MPI run. Typical error message to console:

```
p0_97244:  p4_error: semget failed for setnum=%d: 0
        An error has occurred in cfx5solve:
```

- A new MPI job starts up, but kills an already running MPI job.
- When two MPI jobs are running at the same time, for the same user-id, stopping one MPI job kills the second MPI job.

- The solver hangs when starting a new job

17.4.3.1.3. Checking How Many Semaphores Are in Use, and by Whom

There is a built-in command for UNIX: "`ipcs`", or "`ipcs -a`". Use this command to see what semaphores are owned by your user-id. Typical output is as follows:

```
ipcs
Message Queues:
T      ID        KEY    MODE         OWNER    GROUP
q       0 0x416d02d8 --rw-------      root   system
Shared Memory:
T      ID        KEY    MODE         OWNER    GROUP
m     768    0x7a08e --rw-------      joe    all
m    1409    0x57463 --rw-------      moe    all
Semaphores:
T      ID        KEY    MODE         OWNER    GROUP
s       0 0x416d02d8 --ra-------      root   system
s       1 0x416d029e --ra-------      root   system
s      82    0x7a08e --ra-------      joe    all
s      83    0x7a08f --ra-------      joe    all
s      84    0x7a090 --ra-------      joe    all
s      85    0x7a091 --ra-------      joe    all
s     214    0x57463 --ra-------      moe    all
s     183    0x57464 --ra-------      moe    all
s     184    0x57465 --ra-------      moe    all
s     185    0x57466 --ra-------      moe    all
```

The user `joe` has one shared memory segment and 4 semaphores left over from an abnormally aborted previous CFX computation. These need to be deleted.

17.4.3.1.4. Deleting the Semaphores You Are Using

Once the system limit is reached for semaphores, new Intel MPI jobs are not possible. Because MPI does not delete these semaphores if a simulation terminates abnormally, it is up to each user to manually delete all such semaphores. It is also a good idea to also delete any remaining shared memory segments owned by your account at the end of an aborted run. This leaves the system free for other users to use for their MPI runs. Continuing from the above example, the user "`joe`" would issue the UNIX command "`ipcrm -m xxx -s yyy`" where `xxx` is shared memory ID number and `yyy` is a semaphore ID number. For example:

```
ipcrm -s 82 -s 83 -s 84 -s 85 -m 768
```

This will delete the semaphores and the shared memory segment, leaving the system free for the next user. On some systems, the exact syntax of the `ipcrm` command may differ from that shown above.

17.4.3.1.5. Shared-memory Segment Size Problems

In some cases, it is possible to encounter the following message:

```
p0_2406: (140.296294) xx_shmalloc: returning NULL; requested 65576 bytes
p0_2406: (140.296337) p4_shmalloc returning NULL; request = 65576 bytes
You can increase the amount of memory by setting the environment
variable P4_GLOBMEMSIZE (in bytes)
p0_2406:  p4_error: alloc_p4_msg failed: 0
```

This may occur if Ansys CFX attempts to allocate a shared-memory segment with a size larger than what is allowed by default by Intel MPI (the default is 4 MBytes). The default size can be

adjusted, as mentioned in the error message, by setting the environment variable P4_GLOBMEM-SIZE. For example, to set this variable to 16 MBytes running a C-Shell:

```
setenv P4_GLOBMEMSIZE 16777216
```

You must be careful when you increase the value of this parameter that you do not exceed the maximum size allowed for a shared memory segment. This limit is set by the operating system kernel rather than Intel MPI.

17.4.3.1.6. Checking Semaphore ID and Shared Memory Segment Limits

In some cases, it may be necessary to increase the maximum number of semaphores IDs, or the maximum size of a shared memory segment on your system, so that several users/jobs can each run in parallel, simultaneously. Once the total number of semaphores are used, then more parallel MPI jobs cannot be started. You can find out the maximum number of semaphores on your system with the following commands:

17.4.3.1.6.1. Linux

Semaphores: `/sbin/sysctl -a | grep -i sem`

Shared-memory segments: `/sbin/sysctl -a | grep -i shm`

17.4.3.1.7. Increasing the Maximum Number of Semaphores for Your System

In general, this is not a simple task, and it varies greatly between different computer vendors and operating systems. It may involve changing a system resource file and rebooting the computer, or it may involve making changes to the system kernel and recompilation of the system kernel. Contact your system administrator if you need to increase the number of semaphores on your system.

17.4.3.1.8. Max Locked Memory

On Linux systems, when using the Intel MPI 4.1.3 library in CFX, Intel MPI requires 32 MB of lockable memory for interprocess communication on all parallel hosts. When running distributed parallel with Intel MPI, the solver checks the 'max locked memory' limit and issues a warning if it is insufficient. The default hard limit is usually less than 32 MB, requiring you to change the limit in the system configuration file. Users may raise or lower the current limit up to the hard limit, but only the super-user may raise the hard limit. To check the soft and hard limits, use the following commands:

Shell	Command
bourne shell	ulimit —a
	ulimit —Ha
C shell	limit
	limit —h

The soft and hard limits may be set to a value (in KB) or to 'unlimited' by adding the following lines to `/etc/security/limits.conf`:

- `* soft memlock unlimited`

- `* hard memlock unlimited`

Alternatively, this issue may be overcome by limiting the communication method to TCP between hosts using the environment variable setting `I_MPI_FABRICS=shm:tcp`.

Note:

This setting will prevent the use of any faster interconnects available on the system. The preferred solution is to increase the 'max locked memory' limit described above.

17.4.3.2. Problems with the Ansys CFX Executables

- Check the Hosts File (`hostinfo.ccl`) for syntax errors.
- Check that the specified executables for each host in the Hosts File exist, and that the files have execute permission set.
- Check that the specified executables are correct for the specified hosts (operating system, 64-bit executables, and so on).

17.4.3.3. Problems with Ansys CFX Licenses

- Check that you have the correct parallel licenses to run a parallel job on the assigned hosts.
- Check that you have enough licenses for the specified number of partitions.

17.4.3.4. Windows Problems

- If you are having trouble with distributed parallel MPI, check that you have installed the MPI daemon.
- Inconsistent host name resolution with multiple network adaptors:

  If there are problems with host name resolution, distributed parallel runs might fail. This might occur when using systems that have multiple network adaptors and IP addresses (for example, systems that are connected to both private and enterprise networks). In such cases, the system configuration should be reviewed. A remedy for this problem is to use IP addresses instead of host names.

17.4.3.5. Linux Problems

- CFX Distributed Parallel runs fail

  On some SLES machines (typically ones with more than one network card), the default configuration of `/etc/hosts` will cause CFX distributed parallel runs to fail. In such cases, the problem might be solved by editing the `/etc/hosts` file to remove all lines that contain redundant loopback addresses. Do not remove the line with the first loopback address, which is typically 127.0.0.1.

17.4.3.6. Convergence Problems

Occasionally you may find that jobs submitted in serial will converge while those in parallel fail. This can be due to the different internal structure of the multigrid solver. The partitioned mesh leads to different coarse mesh blocking than the serial mesh, and if you have selected a timestep size that is close to the critical convergence limit, this can cause convergence problems.

Usually a reduction in the timestep size alleviates this problem.

17.4.4. Measuring Parallel Performance

In general, there are two reasons why you might want to run an Ansys CFX job in parallel:

- Getting results faster by combining the processing power of two or more processors. The performance of this set up is measured by the system speedup.
- Performing simulations that require more memory than can be provided by a single machine. The performance of this feature is best measured using the memory efficiency.

In a real-world computer environment, it is not always easy to measure parallel performance. Measured execution times depend on relative speed of processors, which in turn is limited by the processor architecture, as well as its load. For heterogeneous networks, the parallel performance is directly determined by the speed of the slowest processor.

17.4.4.1. Wall Clock Performance

In order to measure wall clock performance, two parameters can be defined:

$$\text{Speedup}, S = \frac{T_s}{T_p} \tag{17.1}$$

$$\text{Efficiency}, E = \frac{T_s}{N\,T_p} = \frac{S}{N} \tag{17.2}$$

where the subscripts s and p refer to the sequential and parallel wall clock execution times respectively, and N is the number of partitions. The best wall clock performance increase that can be expected is a linear speed up ($S=N$), and this corresponds to an efficiency of 100%.

The performance determination can sometimes not be straightforward due to the fact that only part of the CFX-Solver run is actually performed in parallel. The reading and distributing of the CFX-Solver input file data, and the collecting and writing of results file data are highly I/O dependent and not parallelized. These stages therefore depend on high disk speeds and fast network communication for fast operation, and should be neglected in any parallel performance evaluation. The number of seconds taken for the parallelized calculation stage of CFX-Solver can be found on the following line of the CFX-Solver Output file:

```
CFD Solver wall clock seconds: 2.5113E+04
```

This line is found at the end of the outer loop iterations (steady-state run) or timesteps (transient run). This number is accurate only if no backup or transient results have been written during this stage of the calculation because file i/o is a serial process.

17.4.4.2. Memory Efficiency

The memory efficiency:

$$E_m = \frac{M_s}{M_p} \tag{17.3}$$

where

$$M_p = \sum_{i=1}^{N} M_{p,i} \tag{17.4}$$

can be used to compare the total memory required by all processes of a parallel run, M_p, with the memory required by that of the equivalent serial run, M_s. The memory efficiency can be used to compare the total memory required by all processes of a parallel run, with the memory required by that of the equivalent serial run. The memory efficiency for Ansys CFX will always be smaller than 100%, but should generally be in the range of 80-95% for meshes with reasonable partitions. With respect to workstation clusters, it is important to know that all processes (operating on both leader and follower machines) use a nearly identical amount of memory.

Detailed information about the memory requirements of a parallel run are written to the CFX-Solver Output file.

17.4.4.3. Visualizing Mesh Partitions

Partitioning information is appended to the results file during a parallel run of the CFX-Solver. You can view the partitions in CFD-Post by loading the results file and viewing the variable **Real partition number** on a locator to display the partitions of the mesh.

Chapter 18: Expert Control Parameters

This chapter describes:

18.1. When to Use Expert Control Parameters

18.2. Modifying Expert Control Parameters

18.3. CFX-Solver Expert Control Parameters

Additional information on expert control parameters is available. For details, see Expert Control Parameters in the *CFX-Pre User's Guide*.

18.1. When to Use Expert Control Parameters

For a small number of cases, convergence difficulties can arise that cannot be resolved in the normal manner of time step control or judicious setting of initial values. Under these circumstances, modifying control parameters can help when all else fails to bring about a converged solution.

It should be emphasized that the most frequent requirement for access to expert control parameters is to take control of internal CFX-Solver defaults in order to improve or achieve convergence on a difficult case. Before you attempt to change expert control parameter settings, it is important that you have read through the advice section and have taken all the normal steps to address any convergence difficulties. Note that expert control parameters are only intended for use by customers who are experienced in using CFX, or who have been instructed to use expert control parameters by Ansys customer support. Use of these parameters is not fully supported, and may cause unexpected or unintended consequences for both the performance of the CFX-Solver and the quality of the results.

Advice on flow modeling is available so that you can ensure you have done all that you can to obtain a converged solution before proceeding to modify expert control parameters. For details, see Advice on Flow Modeling (p. 557).

18.2. Modifying Expert Control Parameters

You can set and edit expert control parameters in CFX-Pre from the Expert Parameters panel. For details, see Expert Control Parameters in the *CFX-Pre User's Guide*. The **Command File** editor can also be used. For details, see Command File Editor Overview in the *CFX-Solver Manager User's Guide*.

18.3. CFX-Solver Expert Control Parameters

In the following sections, the name and function of the available expert control parameters for the CFX-Solver is given. Each one is given either a logical, integer or real value. All of the parameters can be inserted using the **Command File** editor and "written" back into the CFX-Solver input file before executing the CFX-Solver.

Note that changing some of the CFX-Solver Control Parameters may affect the output generated by the CFX-Solver.

18.3.1. Discretization Parameters

Diffusion Scheme	
`bounded bnd diffusion tets`	
Type	Logical
Default Value	f
Description	Controls whether or not a bounded diffusion scheme is used at boundaries for tetrahedral elements. The diffusion scheme for other element types is bounded by default through the default setting of 5 for the expert parameters 'cht diffusion scheme' and 'scalar diffusion scheme'. Tetrahedral elements are controlled separately because the bounded scheme may lead to less accurate results.
`cht diffusion scheme`	
Type	Integer
Default Value	5
Description	Specifies the diffusion scheme for CHT solids: 1 = Central (interior), central (boundary) 2 = Positive definite coefficients (interior), central (boundary) 3 = Positive definite values (interior), positive definite values (boundary) 4 = Blended scheme (interior), central (boundary) 5 = Positive definite coefficients (interior), positive definite values (boundary) 6 = Blended scheme (interior), positive definite values (boundary)
`diffusion at inlets`	
Type	Logical
Default Value	f
Description	Indicates whether to include diffusion terms at inlets, openings, and outlets for scalar equations (for example, Additional Variables, species, turbulence, and so on).
`meshdisp diffusion scheme`	
Type	Integer
Default Value	2
Description	Specifies the diffusion scheme for mesh displacement equations: 1 = Central (interior), central (boundary) 2 = Positive definite coefficients (interior), central (boundary) 3 = Positive definite values (interior), positive definite values (boundary)

	4 = Blended scheme (interior), central (boundary) A value of 3 may improve the mesh quality is some cases, particularly when mesh folding occurs at sharp corners. For details, see Mesh Displacement Diffusion Scheme.
`pressure diffusion scheme`	
Type	Integer
Default Value	2 (tetrahedral) 1 (hexagonal)
Description	Specifies whether the standard central scheme (default) or the positive definite scheme is applied to the continuity equation. This parameter may be of use in obtaining convergence with poor quality meshes. A value of 2 sets the pressure diffusion scheme to be positive definite.
`scalar diffusion scheme`	
Type	Integer
Default Value	5
Description	Specifies the diffusion scheme for scalars: 1 = Central (interior), central (boundary) 2 = Positive definite coefficients (interior), central (boundary) 3 = Positive definite values (interior), positive definite values (boundary) 4 = Blended scheme (interior), central (boundary) 5 = Positive definite coefficients (interior), positive definite values (boundary) 6 = Blended scheme (interior), positive definite values (boundary)
`stress diffusion scheme`	
Type	Integer
Default Value	2 (tetrahedral) 1 (hexagonal)
Description	Specifies the diffusion scheme for scalars: 1 = Central (interior), central (boundary) 2 = Positive definite coefficients (interior), central (boundary) 3 = Positive definite values (interior), positive definite values (boundary) 4 = Blended scheme (interior), central (boundary) 5 = Positive definite coefficients (interior), positive definite values (boundary)

	6 = Blended scheme (interior), positive definite values (boundary)
`wallscale diffusion scheme`	
Type	Integer
Default Value	5
Description	Wallscale diffusion differencing scheme: 1 = Central (interior), central (boundary) 2 = Positive definite coefficients (interior), central (boundary) 3 = Positive definite values (interior), positive definite values (boundary) 4 = Blended scheme (interior), central (boundary) 5 = Positive definite coefficients (interior), positive definite values (boundary) 6 = Blended scheme (interior), positive definite values (boundary)
Tolerances	
`degeneracy check tolerance`	
Type	Real
Default Value	1e-4
Description	A face set is considered degenerate if the dimensionless area of the face set is less than this tolerance. The dimensionless area is calculated as the ratio of the area of the domain boundary (summed up over all integration point faces on the domain boundary) to a geometry length scale based on the volume of the elements adjacent to the domain boundary. If the dimensionless area is smaller that the specified tolerance, the whole domain boundary is flagged as degenerate. Note that no-slip wall boundaries that are flagged as degenerate are not allowed and the solver will stop with an error message. Also, when symmetry conditions are applied to a degenerate domain boundary, vectors and tensors on the domain boundary are modified to satisfy the required conditions. For example, all components of vectors that are orthogonal to the degenerate domain boundary are set to zero.
`tangential vector tolerance`	
Type	Real
Default Value	10.0
Description	The solver will trigger an error message when the angle between the specified direction at a total pressure inlet and the inlet face is smaller than this angle.
`vector parallel tolerance`	
Type	Real
Default Value	1.0
Description	The value of this parameter is the number of degrees tolerated by the solver in determining the maximum deviation of any element face normal from the calculated average element face normal in a Symmetry Plane

	boundary condition. This error may occur when element inflation is used on surfaces adjacent to the Symmetry Plane boundary. Increase this value to relax the tolerance.
Miscellaneous	
`bcp arithmetic aver sum option`	
Type	Integer
Default Value	0
Description	Controls the interpolation of variables when calculating arithmetic averaging and sum quantitative CEL functions on boundary patches, as well as RMS averaging quantitative CEL functions. The results calculated by these quantitative CEL functions can differ between CFX-Solver and CFD-Post. A value of 0 indicates that variables at face centers are used for the calculation, while a value of 1 indicates that variables at boundary vertices are used. The latter option leads to results that are more consistent between CFX-Solver and CFD-Post; however, the results can still differ due to edge and corner effects. This option can also lead to changes in the convergence of solutions computed by CFX-Solver, because sometimes the results of quantitative CEL functions of these types are used in the solver control area. For details on how CFD-Post and CFX-Solver compute values differently, see ave in the *CFX Reference Guide*, rmsAve in the *CFX Reference Guide*, and sum in the *CFX Reference Guide*.
`build artificial wall`	
Type	Logical
Default Value	t
Description	Artificial walls are usually built during the solution if the solver detects inflow at an outlet boundary, or outflow at an Inlet boundary. Set this parameter to `'f'` to prevent artificial walls from being built. Setting this to `'f'` also enables you to set up a static-pressure to static-pressure flow problem, by enabling flow to be 'sucked' into the domain through an outlet boundary. Very small timestep sizes are usually required (an L/V scale where L approaches the mesh spacing at the inflow), and this method in general is susceptible to severe convergence difficulties.
`fsdamp efficiency scale`	
Type	Real
Default Value	0.01
Description	Scaling factor for the free stream damping used in the efficiency calculation. For details, see Isentropic Efficiency and Total Enthalpy (p. 101).
`highres energy option`	
Type	Integer

Default Value	0
Description	This parameter controls how the high resolution blend factor is calculated for the energy equation. Note that the blend factor is always shared with the species equations for multicomponent flow, with the actual blend factor taken as the most conservative of all contributions from the energy and mass fraction group. The information below indicates which variable is used to calculate the energy equation blend factor for single-component flow and the energy equation's contribution to the shared blend factor for multicomponent flow: 0 = Enthalpy for single component flow, enthalpy for multicomponent flow. 1 = Enthalpy for single component flow, no contribution for multicomponent flow. 2 = Enthalpy for single component flow, temperature for multicomponent flow. 3 = Temperature for single component flow, temperature for multicomponent flow.
`ipmt area density clip`	
Type	Logical
Default Value	t
Description	Controls whether the interfacial area density is clipped to a non-zero value for interphase mass transfer models. The value at which clipping occurs is obtained by using the specified minimum volume fraction (see Minimum Volume Fraction in the *CFX-Solver Modeling Guide* (p. 302)) in the calculation of interfacial area density. Using a clipped interfacial area density can serve as a crude nucleation model for some kinds of mass transfer, but can affect mass conservation within the solution.
`laplacian stresses`	
Type	Logical
Default Value	f
Description	When set to `'t'`, causes a Laplacian form of the viscous stresses to be used instead of the strictly-correct stress tensor form. When there are strong shear layers in regions of high mesh aspect ratio, the full viscous stress tensor can cause convergence slow-down or convergence stalling at a moderate level (for example, maximum residuals at 1.0E-2). The Laplacian form converges better in such cases, but is formally less accurate.
`linearly exact numerics`	
Type	Logical

Default Value	f
Description	Controls whether numerics are linearly exact (that is, give a zero error in the presence of a linear solution field). This is a feature that has so far been found to be useful when there is quiescent flow in a hydrostatic pressure field and the mesh is a non-orthogonal hexahedral or mixed-element mesh.
`porous cs discretisation option`	
Type	Integer
Default Value	3
Description	Specifies how the pressure is treated at interfaces to a porous domain: 1 = Constant static pressure 2 = Constant total pressure 3 = Constant total pressure based on the normal component of velocity at the interface Option 1 is the least physically realistic setting. Option 2 is a more realistic setting. However, when using this setting the CFX-Solver may fail to converge in cases where sections of the porous interface have little or no flow normal to the interface. Option 3 is the most realistic setting and also converges better than option 2.
`tef numerics option`	
Type	Integer
Default Value	1
Description	The $k-\omega$ and SST turbulence models sometimes give convergence difficulties in areas of extremely high gradients in ω, for example, at a blunt trailing edge or sudden rearward facing surface, in conjunction with a highly refined boundary layer grid (generally resolved into the viscous sub-layer). The SST model is most susceptible to this problem. Symptoms are generally a lack of convergence with residuals stuck at regions of extremely high gradients of omega, sometimes leading to sudden, early total failure of the solver. In these cases, a flux limiting numerics implementation can be used for the ω equation, activated by setting this parameter to 1.
`transient startup option`	
Type	Integer
Default Value	0
Description	Controls the startup option for 2nd order transient discretization: 0 = 1st order scheme applied to first timestep (non-conservative). 1 = Modified 2nd order applied to first timestep (conservative).

	2 = Standard 2nd order applied to first timestep, but with $d\varphi/dt = 0$ assumed at start.
`vfr gradient force option`	
Type	Integer
Default Value	2
Description	Used to switch between old and new discretization schemes for force terms proportional to volume fraction gradients, namely, turbulent dispersion force and solid pressure. The old scheme adds the force explicitly as a source term. The new formulation discretizes the term implicitly as a gradient term, hence allows the term to be taken into account in a coupled manner when used in conjunction with the coupled volume fraction algorithm. Possible values are: 0 = Original formulation, added to source terms 1 = New formulation for turbulent dispersion forces, but excluded from Rhie-Chow 2 = New formulation for turbulent dispersion forces, including Rhie and Chow 3 = Include solids pressure as well
`virtual mass force option`	
Type	Integer
Default Value	1
Description	Used to switch between old and new discretization schemes for virtual mass forces. The old scheme adds the force explicitly as a source term, with velocity gradients discretized using central differences. The new formulation discretizes the term implicitly, and uses an upwind scheme to discretize velocity gradients. Possible values are: 0 = Original source term formulation 1 = New implementation upwind, ignoring mass imbalance 2 = New implementation upwind, including mass imbalance
`viscous work at ip`	
Type	Logical
Default Value	f
Description	Discretizes the viscous work term in the energy equation at integration points (ip).
`volfrc sumapp`	
Type	Logical
Default Value	t
Description	For multiphase calculations, the linear equations for phasic continuity (that is, the volume fraction equations) are not always diagonally dominant. Setting this parameter to true will force the linear equations

	to be diagonally dominant. This may improve robustness in some situations, but may also considerably slow down the convergence rate.

18.3.2. Linear Solver Parameters

`mg solver option`	
Type	Integer
Default Value	5 (automatic)
Description	For hydrodynamic equations (u, v, w, p), determines the coarsening algorithm for algebraic multigrid of the fluid coupled equations. A value of 5 (automatic) has the following behavior: • If the simulation involves multiphase flow with the free surface model activated, or with coupled volume fractions, option 2 is used. • Otherwise, option 4 is used. A value of 4 means that the anisotropic coarsening is designed to give robust convergence for the widest possible range of flow conditions. Alternative coarsening algorithms are also available for comparison purposes. A value of 3 means that coarsening is based on the velocity coefficient. A value of 2 means that the anisotropic coarsening is based on the pressure coefficient. This was the previous default value. A value of 1 sets an isotropic coarsening based on topology only (similar to conventional geometric multigrid).
`multigrid solver`	
Type	Logical
Default Value	t
Description	Control whether multigrid is used in CFX-Solver.
`overlap relaxation fluids`	
Type	Real
Default Value	1.0
Description	Linear solver under-relaxation of overlap equations in a parallel run for hydrodynamics equations.
`solver relaxation fluids`	
Type	Real
Default Value	0.9
Description	The under-relaxation value for the coupled U, V, W and P linear smoother within the multigrid solver.

	On a very poor mesh, coupled U, V, W and P linear solver failure can be helped by increasing this under-relaxation to, for example, 0.75.
`solver relaxation scalar`	
Type	Real
Default Value	0.9
Description	The under-relaxation value for the scalar linear smoother within the multigrid solver. On a very poor mesh, scalar equation linear solver failure can be helped by increasing this under-relaxation to, for example, 0.75.
`solver target reduction fluids`	
Type	Real
Default Value	0.1
Description	The linear solver for the coupled mass-momentum system iterates until the final RMS residual for the continuity equation is below this number times the initial RMS continuity residual. It is occasionally useful, for testing purposes for instance, to drive the linear solver to a tighter tolerance (for example, to machine round-off).
`solver target reduction scalar`	
Type	Real
Default Value	0.1
Description	The linear solver for the coupled mass-momentum system iterates until the final RMS residual for the continuity equation is below this number multiplied by the initial RMS continuity residual. It is occasionally useful, for testing purposes for instance, to drive the linear solver to a tighter tolerance (for example, to machine round-off).

18.3.3. I/O Control Parameters

General I/O Control	
`include pref in forces`	
Type	Logical
Default Value	f
Description	If set to `'t'`, the solver will account for the specified reference pressure in the force and moment calculation.
`mcf concentrations output option`	
Type	Integer
Default Value	2
Description	Controls the output of mass concentrations, molar concentrations and molar fractions to the results file. 0 = Determine what data to write from the VARIABLES file (or CCL) 1 = Do not write this data

	2 = Do not write this data for mixture fraction based models, and do write this data for other combustion models or no combustion model 3 = Always write this data To achieve full backwards compatibility with version 10.0, set this parameter to 3. Note that, in the case of a large number of components (>20), the computational cost of writing the data may be unacceptably high.
`read boundary flows`	
Type	Logical
Default Value	f
Description	When set to `'t'`, the solver reads in boundary flows from the initial values when restarting from a previous run. This helps with smooth restart of cases with user function CEL expressions involving forces or boundary flows.
Backup Results	
`backup file at start`	
Type	Logical
Default Value	f
Description	Controls whether a backup file is written at the start of every run (that is, showing the initial values field for the run). Mass flow information written to a backup file at the start of the run is not accurate. To calculate accurate mass flow from a backup file, use information from the backup file written after the first timestep/iteration of the run.
`backup file at zero`	
Type	Logical
Default Value	f
Description	Controls whether a backup file is written after zero timesteps (that is, showing the initial guess field for the run). Mass flow information written to a backup file at time zero is not accurate. To calculate accurate mass flow from a backup file, use information from the backup file written after the first timestep/iteration of the run.
Transient Results	
`mfloip to trn files`	
Type	Logical
Default Value	f
Description	When set to `'t'`, integration point mass flows are written to transient files (`*.trn`). This will ensure numerically consistent mass flow evaluations when postprocessing transient files.
Output File and Monitor Data	

always output post processing	
Type	Logical
Default Value	f
Description	When this parameter is set to 't', flow summaries, variable ranges, forces, moments and scales are written to the CFX-Solver Output file on intermediate adaption steps.
fmu logging	
Type	Logical
Default Value	f
Description	This expert parameter controls whether logging information provided by an FMU is written to the standard output of a CFX-Solver run. The amount of logging information is controlled by the FMU.
monitor digits	
Type	Integer
Default Value	8
Description	Controls the number of digits used in the MON file. Increase the value to increase the precision of the results reported.
monitor ftrans	
Type	Logical
Default Value	f
Description	When this parameter is set to 't', false transient information (steady-state only) is written to the CFX-Solver Output file for each coefficient loop.
monitor fmu	
Type	Logical
Default Value	f
Description	This expert parameter controls whether the values of all FMU input and output variables and parameters are written to the CFX-Solver Output file of a CFX-Solver run whenever the FMU is called.
monitor ranges	
Type	Logical
Default Value	f
Description	When set to 't', outputs variable ranges for each coefficient loop in a steady-state simulation or for each timestep in a transient simulation.
monitor scales	
Type	Logical
Default Value	f
Description	When set to 't', outputs variable scales for each coefficient loop in a steady-state simulation or for each timestep in a transient simulation.
wall clock time	
Type	Logical
Default Value	f

Description	Output wall clock time at the start of each outer loop in a steady-state run or timestep loop in a transient run and in the summary at the end of the run.

18.3.4. Convergence Control Parameters

Relaxation Parameters	
`model coefficient relaxation`	
Type	Real
Default Value	1.0
Description	Sometimes poor convergence is observed with the $k-\varepsilon$ turbulence model, characterized by bouncy k and ε residual graphs. Changing this under-relaxation parameter can help, but do not use values lower than 0.1.
`outer loop relaxations default`	
Type	Real
Default Value	0.75
Description	Parameter `outer loop relaxations default`, which defaults to 0.75, is designed to provide a default value for multiple relaxation parameters that operate at the outer loop level. Currently, parameter `outer loop relaxations default` provides a default value for only one relaxation parameter, `relax mass`, and only for steady state cases. Bouncy convergence, even with a small timestep, can sometimes be reduced by lowering this value to 0.5 or less. A value below 0.25 is not recommended because that would drastically slow down convergence.
`relax mass`	
Type	Real
Default Value	0.75
Description	Changing the value of this parameter can aid convergence for problems that show residual oscillations in separation and reattachment regions. Because it relaxes the mass flow calculation procedure, it should not be used for transient problems (or should be set to 1). For transient cases, parameter `relax mass` defaults to 1. For steady state cases, parameter `relax mass` defaults to the value of parameter `outer loop relaxations default`.
`wallscale relaxation factor`	
Type	Real
Default Value	0.75
Description	This is a relaxation factor for the wallscale equation, which is solved for all $k-\omega$ and SST turbulence models.

	The wall scale equation is a pure diffusion equation and is therefore very sensitive to non-orthogonal meshes (such as a mesh that is highly skewed at a corner). If convergence problems for a wallscale equation are observed, try reducing the value of the `wallscale relaxation factor`.
Convergence and Runtime Control	
`allow cbck for initcon`	
Type	Logical
Default Value	f
Description	If set to `'t'`, expressions involving callbacks (such as integrated values) can be used to set initial conditions. By default it is not allowed because allowing callbacks introduces the risk of hanging in parallel.
`auto turnoff solve eq flag`	
Type	Logical
Default Value	f
Description	This flag determines whether to stop solving individual equations when residuals have fallen below their convergence criteria. With this flag set to true then, for example, the solution for each of the U, V, W, and P-Mass equations is stopped when the corresponding equation residual falls below its target by more than 10%. The run is completed when convergence has been achieved and when the Additional Variable equations have been solved satisfactorily. Note that turbulence equations are ignored by the solver when determining the level of convergence, and so are not affected by this flag. This is only functional for steady-state models, and when the model contains one or more Additional Variables. It is automatically set to `'t'` if the Auto Timescale feature is invoked.
`check isolated regions`	
Type	Logical
Default Value	t
Description	For serial runs, the solver checks if any fluid domain contains volumetric regions that are isolated pockets. This check cannot be performed for parallel solver runs.
`ignore solve flag on restart`	
Type	Logical
Default Value	f
Description	The solution of the wallscale equation stops after it has converged, even if the other equations have not yet converged. If you perform a restart and reduce the convergence criterion, it will not start solving again because the flag indicating it has converged is still set. To start solving the wallscale equation again on a restart, set this parameter to `'t'`.
`imbalance normalisation type`	
Type	Integer
Default Value	2

Description	Specifies how imbalances are normalized: 0 - no normalization 1 - normalize by maximum contribution in domain 2 - normalize by maximum contribution in all connected domains
`require coefloop convergence`	
Type	Logical
Default Value	f
Description	If set to `'t'` then the solver will stop in a transient run if convergence is not achieved within the coefficient loop
Memory Control	
`topology estimate factor`	
Type	Real
Default Value	1.0
Description	**Note:** Do not use this parameter unless informed to in a run-time error message. Adjusts an internal factor used to help when the solver is estimating the total memory required to store the control volume equation matrix. Sometimes the internal factor is not sufficient and the solver will stop indicating that the "topology estimate factor" parameter must be increased. If this is the case, supply a number for this parameter that is larger than 1; for example, 1.2 will increase the internal memory estimate by 20%, which is often sufficient.
`topology estimate factor zif`	
Type	Real
Default Value	1.0
Description	**Note:** Do not use this parameter unless informed to in a run-time error message. Adjusts an internal factor used to help when the solver is estimating the total memory required to store the matrix coefficients at GGI domain interfaces. Sometimes the internal factor is not sufficient and the solver will stop indicating that the "topology estimate factor" parameter must be increased. If this is the case, supply a number for this parameter that is larger than 1; for example, 1.2 will increase the internal memory estimate by 20%, which is often sufficient.
High Speed Models	
`max continuity loops`	

Type	Integer
Default Value	1
Description	Sets the maximum number of continuity loops to perform within a timestep. The continuity loop iterates on the density*velocity nonlinearity in the continuity equation. The default value of 1 is usually appropriate. For high-speed supersonic flows (Mach numbers above 2), increasing the value to 2 may help convergence.

18.3.5. Physical Model Parameters

Material Properties	
`realeos liquid prop`	
Type	Integer
Default Value	1
Description	This parameter controls whether liquid saturation properties are extracted using the Rackett equation, which is a function of temperature (realeos liquid prop = 1), or using the full equation of state, which is a function of both temperature and pressure (realeos liquid prop = 2) — the latter is more accurate. For details, see Real Gas Liquid Properties in the *CFX-Solver Theory Guide*.
Combustion Models	
`coupled scalars`	
Type	Logical
Default Value	f
Description	All combustion models that solve for the mixture fractions of the reactants and products (single or multiple step reaction models), are solved using a coupled multigrid linear solver. This means that all of the mixture fraction equations are solved simultaneously, with the mixture mass transfer source terms linearized in each equation. Under certain conditions, this coupling procedure can fail, leading to oscillatory convergence or divergence. In this case, you can disable the coupled solver and switch to a segregated approach to solving the mixture fraction equations. This is achieved by setting this parameter to `'f'`.
`use kolmogorov ts for extinction`	
Type	Logical
Default Value	f
Description	This parameter is provided for backwards compatibility with the first release of the extinction model. It is not recommended for use (leave at default setting of `'f'`, which uses the turbulence time scale for the flame extinction model). When set to `'t'`, the Kolmogorov time scale is used in the extinction model (not recommended).
Turbulence Models	
`apply ic fluctuations for les`	
Type	Logical

Default Value	f
Description	If set to `'t'`, to force the application of velocity fluctuations when restarting an LES. This is most useful when using a RANS solution as this initial guess. For details, see LES Initialization (p. 216).
General Grid Interface	
`force intersection`	
Type	Logical
Default Value	f
Description	On all GGI interfaces (fluid-fluid attachments, periodicity and frame change interfaces), the solver performs an intersection procedure to connect the two sides of the interface together. This procedure is CPU intensive, so the result of the intersection is stored in the outgoing results file, for future use (for example, upon restart of the simulation). You can force a re-intersection of the GGI interface if desired, upon restart, by setting this parameter to `'t'`. It is rare that this parameter should be needed, as it generally only results in wasted CPU time upon a restarted simulation.
`ggi vertex weighting value`	
Type	Integer
Default Value	2
Description	Weighting applied to GGI vertices during partitioning. A larger value assigns a larger weight to vertices on GGI boundaries to account for the fact that the assembly effort is higher at GGI interfaces. For more information, see Node-based and Element-based Partitioning (p. 590).
`mpf ggi force flow balance`	
Type	Logical
Default Value	f
Description	Usually flow balance is not enforced at GGI interfaces for multiphase flows. By setting the value of this parameter to 't', the code will enforce flow balance. This can help with the convergence of many inhomogeneous multiphase flows that include such interfaces.
`stage energy closure option`	
Type	Integer
Default Value	2
Description	With option = 2, conservation of rothalpy is enforced; with option = 1, conservation of rothalpy is not enforced.
`stage velocity factor option`	
Type	Integer
Default Value	1
Description	This parameter is applicable only to the mixing plane with the `Constant Total Pressure` downstream velocity constraint option. The default value of 1 provides the standard treatment of the mixing plane. Use a value of 2 if the momentum conservation across the mixing plane (stage) interface is found to be unsatisfactory.

Mesh Displacement

`meshdisp phase angle convention`

Type: Integer

Default Value: 1

Description: Expert Parameter `meshdisp phase angle convention` selects a convention for traveling wave direction:

- For a value of 0, no convention is enforced.
- For a value of 1 (default):
  - For a rotating component: a forward traveling wave, which corresponds to a positive phase angle or IBPA, moves in the direction of machine rotation.
  - For a stationary component (for example, a model of a stand-alone stator): a forward traveling wave moves in the direction of increasing sector tag number (increasing angle variable).

  A forward traveling wave that uses this convention is illustrated in Case 3: Blade Flutter (p. 277).
- For a value of 2:
  - For a rotating component: a forward traveling wave, which corresponds to a positive phase angle or IBPA, moves opposite to the direction of machine rotation.
  - For a stationary component (for example, a model of a stand-alone stator): a forward traveling wave moves in the direction of increasing sector tag number (increasing angle variable). Note that this is the same as for a value of 1.

Miscellaneous

`asm momentum drift flux`

Type: Logical

Default Value: f

Description: This parameter adds the ASM drift fluxes into the momentum equation.

`mpf ptot option`

Type: Integer

Default Value: 0

Description: When `mpf ptot option = 0` (the default), only the compressible phase is considered when computing the total pressure; the dispersed phase contributions are not added.

When `mpf ptot option = 1`:

	• For total pressure boundaries in cases where the continuous phase is compressible, the dispersed phase contributions are added to the total pressure. • For porous interfaces, an incompressible fluid approximation is made for all phases. • For stage interfaces, an incompressible fluid approximation is made for all phases. The approximations that are made when `mpf ptot option = 1` are accurate in each of the following limits: • The dispersed phases are dilute by mass. • The continuous compressible phase has a low Mach number. • The continuous phase is dilute by mass fraction. Note that, for droplets in gas, the dilute droplet mass fraction limit can imply a very small droplet volume fraction.
`tbulk for htc`	
Type	Real
Default Value	300 K
Description	When the Heat Transfer Coefficient (HTC) is computed for a temperature specified wall, by default a near-wall fluid temperature is used for a temperature scale. However, for consistency with traditional 1D analyses, you may want to enter a reference bulk temperature to compute the HTC. This parameter is that reference value. Thus the HTC computed when this parameter is provided is equal to the local heat flux calculated by the solver divided by the difference of the specified wall temperature and this specified bulk temperature.
`topology simplification`	
Type	Logical
Default Value	t
Description	This parameter controls internal mesh topology simplification. It can improve performance for models with a large number of 2D primitives. While the simplification will not change the output regions (in CFD-Post), it may lead to small differences in solution results due to rounding errors, especially for single precision solutions. Note that topology simplification does not take effect for models involving System Coupling or radiation. Also note that topology simplification cannot be used if there are region changes introduced by re-reading CCL during a solver run (for example, using the **Edit Run In Progress** command).
`transient initialisation override`	
Type	Logical
Default Value	f

Description	When set to `'t'`, enables solver default values to be used for initialization in a transient simulation.

18.3.6. Particle Tracking Parameters

`pt distribute trans particles`	
Type	Logical
Default Value	t
Description	Uniformly distribute transient particles over the time step in a transient run. If this parameter is set to `'f'`, all particles are injected at the start of the time step.
`pt extrap vel for integ`	
Type	Logical
Default Value	f
Description	Use extrapolation of the particle velocity during particle integration. This may be helpful for cases with streamline curvature.
`pt fluid var interpolation option`	
Type	Integer
Default Value	1 (for cases that involve particle reactions) 2 (for cases that do not involve particle reactions)
Description	This parameter controls whether fluid variables are interpolated to the current particle position. Values: 0 = No interpolation. All fluid variables are assumed to be constant within a control volume. 1 = Interpolate all fluid variables except those that represent species concentrations. 2 = Interpolate all fluid variables.
`pt force export massflow`	
Type	Logical
Default Value	f
Description	When this parameter is set to `'t'`, particle mass flows are always written to the results files. Otherwise they will only be written if they are used in CEL expressions or monitors.
`pt linear src coef multiplier`	
Type	Real
Default Value	1.0
Description	Defines a factor which multiplies the linearization coefficient of the particle sources.
`pt maximum particle temperature`	

Type	Real
Default Value	3000.0
Description	Defines upper bound for the particle temperature in units of Kelvin [K].
`pt minimum diameter`	
Type	Real
Default Value	1.E-8
Description	Defines the minimum allowed particle diameter. Tracking of a particles is stopped if its diameter falls below this value.
`pt minimum eddy life time`	
Type	Real
Default Value	0.0
Description	Defines the minimum eddy life time for turbulent dispersion of particles.
`pt time interpolation`	
Type	Integer
Default Value	1
Description	Used in a transient run in order to decide how fluid quantities have to be interpolated to the current particle time. Possible values are: 0 = Use newest time level 1 = Linear interpolation between current and previous time levels 2 = Quadratic interpolation between current and two previous time levels
`pt tracks in absolute frame`	
Type	Logical
Default Value	f
Description	Output tracks in absolute frame of reference. Important for postprocessing of non-transient RFR cases. Not required for transient cases.
`pt zone specific tracks`	
Type	Logical
Default Value	t
Description	This parameter controls how particle tracking information is written to the results file. By default, this expert parameter is set to `'t'` and particle tracks will contain domain information. This information enables CFD-Post, for example, to display particle tracks on a user specified domain only. In a multi-configuration run, where the number of domains may vary across simulations, it can happen that the interpolator can no longer map the source domain to a target domain. In this case, the source particle tracks are not copied to the target domain and are therefore lost. To avoid this problem, it is suggested that you set the `pt zone specific tracks` parameter to `'f'` for multi-configuration runs.

	This enables the interpolator to always copy particle tracks to the target file, regardless of any changes in the number of source to target domains. **Note:** When the pt zone specific tracks parameter is set to `'f'`, no domain filtering will be possible in CFD-Post.
`solve particles`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the particle transport equations

18.3.7. Transient Blade Row Model Parameters

`ps max coeff loops`	
Type	Integer
Default Value	3
Description	This parameter controls the maximum number of coefficient loops in the initialization and phase shifted startup cycles of the Fourier Transform model. This parameter applies only to cases that do not model any transient rotor stator interfaces using the Fourier Transform model.
`ps max coeff loops trs`	
Type	Integer
Default Value	3
Description	This parameter controls the maximum number of coefficient loops in the initialization and phase shifted startup cycles of the Fourier Transform model. This parameter applies only to cases that model at least one transient rotor stator interface using the Fourier Transform model.
`ps number periodic cycles`	
Type	Integer
Default Value	2
Description	Defines the number of periodic cycles of initialization of the Fourier Transform model. The purpose of the cycles is to initialize the Fourier coefficients before the start of phase shifting. The length of these cycles is the period of the disturbance being modeled. This parameter applies only to cases that do not model any transient rotor stator interfaces using the Fourier Transform model.
`ps number periodic cycles trs`	
Type	Integer
Default Value	2
Description	Defines the number of periodic cycles of initialization of the Fourier Transform model. The purpose of the cycles is to initialize the Fourier

	coefficients before the start of phase shifting. The length of these cycles is the period of the disturbance being modeled. This parameter applies only to cases that model at least one transient rotor stator interface using the Fourier Transform model.
`ps number startup cycles`	
Type	Integer
Default Value	1
Description	This parameter controls the number of phase shifted startup cycles of the Fourier Transform model. The purpose of the cycles is to limit the maximum number of coefficient loops at the start of phase shifting. The length of these cycles is the period of the disturbance being modeled. This parameter is applicable to inlet disturbance or flutter cases.
`ps reconstr mflow from sfc`	
Type	Logical
Default Value	t
Description	This is a flag that controls the reconstruction of the mass flow at the pitchwise periodic interfaces from the accumulated Fourier coefficients for the Fourier Transform model. This parameter applies only to cases that do not model any transient rotor stator interfaces using the Fourier Transform model.
`ps reconstr mflow from sfc trs`	
Type	Logical
Default Value	t
Description	This is a flag that controls the reconstruction of the mass flow at the pitchwise periodic interfaces from the accumulated Fourier coefficients for the Fourier Transform model. This parameter applies only to cases that model at least one transient rotor stator interface using the Fourier Transform model.
`ps relax sngl fourier coeff value`	
Type	Real
Default Value	0.9
Description	This is a relaxation parameter applied to the accumulation of Fourier coefficients for the Fourier Transform model. This parameter applies only to cases that do not model any transient rotor stator interfaces using the Fourier Transform model.
`ps relax sngl fourier coeff value trs`	
Type	Real
Default Value	0.9
Description	This is a relaxation parameter applied to the accumulation of Fourier coefficients for the Fourier Transform model. This parameter applies only to cases that model at least one transient rotor stator interface using the Fourier Transform model.
`ha optimization level`	
Type	Integer

Default Value	3
Description	This parameter selects between different built-in algorithms for improving the rate of solution convergence. This parameter has no effect on the results obtained for a converged solution. Set this parameter to 0 to revert to the behavior of Release 2019 R3 and earlier. Other valid values are: 1, 2, 3. If your case fails to converge, changing the value of this parameter might help.

18.3.8. Run Control Parameters

Parallel and Partitioner Control	
`parallel optimization level`	
Type	Integer
Default Value	0
Description	This parameter applies pre-set optimizations to improve parallel performance. Increasing the value of this parameter will increase the expected performance gain, but it may also reduce robustness in some cases. Other expert parameters are available to control individual performance optimizations.
`part adjacency output`	
Type	Logical
Default Value	f
Description	This parameter controls whether the partition adjacency data is stored in the `.par` file.
`part cvs weighting`	
Type	Logical
Default Value	f
Description	This parameter can be used to enable a more complex, experimental algorithm for parallel partitioning. This method attempts to assign different weights to each node/vertex in the domain, depending on the complexity of the elements surrounding a vertex. It attempts to make a more balanced partitioning for mixed element grids.
`part mmesh intersect option`	
Type	Integer
Default Value	0
Description	This parameter can be used to enable a modified algorithm for partitioning moving mesh cases. The modified algorithm recognizes moving domain interfaces. A value of `1` selects the newer algorithm. A value of `0` selects the original algorithm. The modified algorithm aims to keep both sides of a GGI interface within the same partition, potentially improving convergence behavior in parallel and reducing communication requirements between partitions.

	This expert parameter affects parallel runs only; it has no effect on serial runs.
`rad data in par file`	
Type	Logical
Default Value	f
Description	This parameter controls whether the radiation mesh is stored in the `.par` file.
Licensing	
`reserve solver licenses`	
Type	Logical
Default Value	T
Description	This parameter causes preliminary tools, such as the partitioner, to reserve the required solver licenses for any following solver runs, helping to avoid a license availability issue after having invested time in running a preliminary tool. Note that this parameter has no effect when running the solver in Partition Only mode because there is no subsequent solver step that would require a license. Note also that the parameter (when set to "T") attempts to provide, but does not guarantee, license availability; it is unlikely but possible that an unrelated solver process will use one of the required licenses in the small window of time after the preliminary tools have run but before the solver has started.

18.3.9. Model Override Parameters

`solve emag`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the solver solves the electromagnetic equations.
`solve energy`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the energy equation.
`solve fluids`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the hydrodynamics equations. Use if you have obtained convergence and you need to continue solving other equation groups.

`solve fsd`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the solver solves the flame surface density equation.
`solve masfrc`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves for mass fraction.
`solve mixture fraction`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves for mixture fraction mean and variance equations.
`solve postproc masfrc`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the mass fraction equations in chemistry postprocessing. This flag affects only postprocessed components. The equations for regular components of the mixture are solved in the main iteration and may be controlled using the solve `masfrc parameter`.
`solve postproc tvariance`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the temperature variance equation in chemistry postprocessing. This flag applies only if the temperature variance equation is solved as part of chemistry postprocessing. When solved as part of the main iteration, the corresponding solve temperature variance flag applies.
`solve radiation`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the radiation equations
`solve reaction progress`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves for reaction progress equations. For details, see Other Parameters (p. 441).
`solve scalar`	
Type	Logical

Default Value	t (if running with Additional Variables)
Description	Flag for controlling whether or not the controller solves equations for Additional Variables.
`solve soot`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves equations for soot equations.
`solve temperature variance`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves for the temperature variance equation.
`solve turbulence`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the turbulence model equations
`solve volfrc`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the volume fraction equations
`solve wallscale`	
Type	Logical
Default Value	t
Description	Flag for controlling whether or not the controller solves the wall scale and boundary scale equations. Note that these equations must be solved to access the Wall Distance and Boundary Distance variables. Detailed information on the wall scale equation is available at Wall and Boundary Distance Formulation in the *CFX-Solver Theory Guide*.

Chapter 19: User Fortran

This chapter discusses:

19.1. User CEL Functions and Routines

19.2. User Junction Box Routines

19.3. Shared Libraries

19.4. User Parameters

19.5. Utility Routines for User Functions

19.6. CFX Memory Management System (MMS)

19.7. User Fortran in Ansys Workbench

19.8. User CEL Examples

19.9. User Junction Box Examples

19.10. Using CFX-4 Routines in CFX

19.11. State Point Evaluation

19.12. Calculating the Particle Drag Coefficient

To add additional features and physical models to CFX, you can write your own subroutines in Fortran and have the CFX-Solver call these routines through a source code interface. You may also want to implement customized physical models that would never be available in CFX due to confidentiality considerations.

CFX supports user subroutines written in Fortran 77 or Fortran 90. A list of supported compilers for each platform is available. Using Fortran 77 whenever possible is recommended.

Two different kinds of user routines are available in CFX:

- User defined CEL (CFX Expression Language) functions can be used within a CEL expression, following the standard CEL rules. For details, see User CEL Functions and Routines (p. 632).
- Junction box routines can be used at several places in the CFX-Solver to run user code. For details, see User Junction Box Routines (p. 635).

The following tasks can be accomplished with user subroutines in CFX:

- Input of user data (for example, data required for profile boundary conditions or externally generated sources).
- User-specified boundary conditions (for example, profile boundary conditions).
- User-specified initial conditions (for example, externally generated flow fields, random distribution or disturbance of existing solutions).

- User-specified source terms (for example, externally generated body forces or general additional source terms used to implement new physical models).
- Junction box routines called every timestep, which acts as a general interface between the CFX-Solver and other software (for example, structure mechanic codes). Junction boxes also offer an interface for advanced monitoring and solution output.
- User particle routines are used to specify sources of momentum, heat and mass transfer, and can also be used to specify injection regions for particles. The structure of particle user routines is the same. An example of this functionality is available. For details, see Structure of User CEL Functions (p. 633), Particle User Sources (p. 374), and User Defined Injection Regions (p. 397).

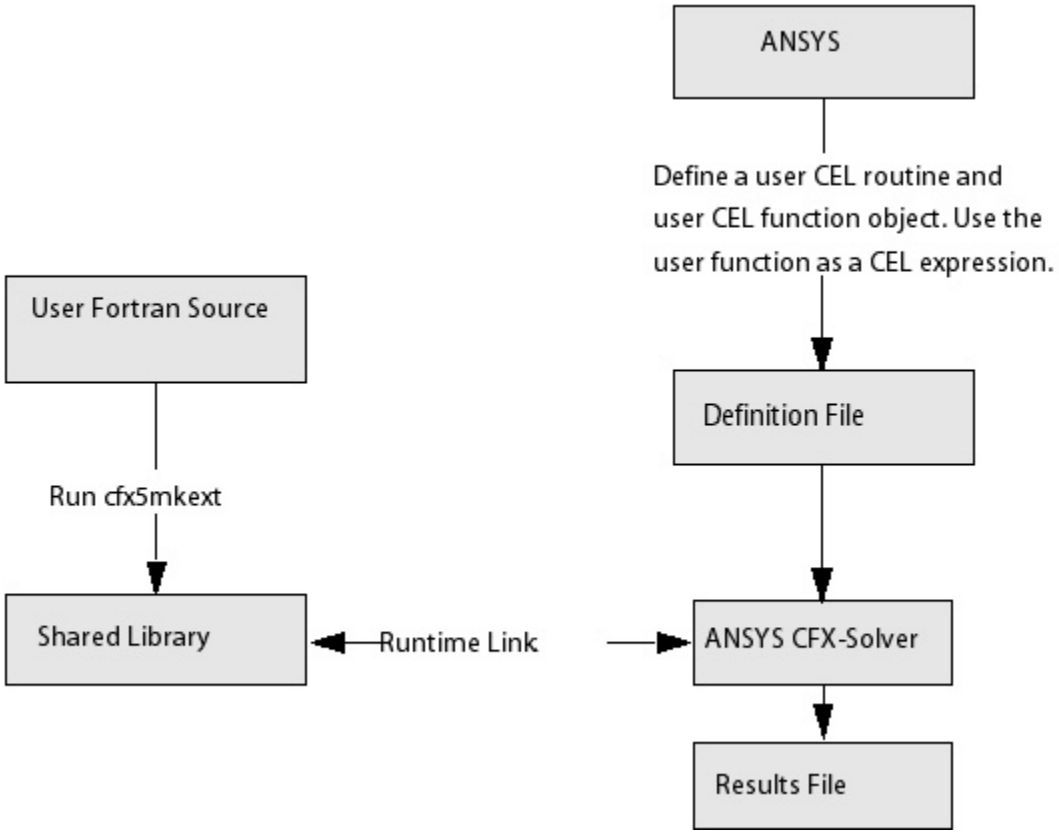
Note that CFX includes features such as advanced monitoring of solution variables or global values and extended CEL functionality, which may reduce your need for user subroutines.

To use junction box routines, you will need to be familiar with the CFX MMS (Memory Management System) to set up and pass around user data for access in any subroutine. For details, see CFX Memory Management System (MMS) (p. 678).

Shared libraries enable subroutines to be reused without recompilation for successive CFX-Solver runs or even for different applications. The location of a shared library is specified in CFX-Pre. During execution of the CFX-Solver, the user subroutines are loaded from the specified shared libraries. For details, see Shared Libraries (p. 640).

19.1. User CEL Functions and Routines

User CEL functions enable you to create your own functions in addition to the predefined CEL functions (such as sin, cos, step, and so on). You can then use these functions in any expression where a CEL function can be used. A user CEL function passes an argument list to a subroutine that you have written, and then uses the returned values from the subroutine to set values for the quantity of interest. The figure below demonstrates the concept.



All variables that are available for use in standard CEL expressions are also available for use in User CEL Expressions. A list of these variables is available in Variables and Predefined Expressions Available in CEL Expressions in the *CFX Reference Guide*.

Details on creating user CEL functions in CFX-Pre and defining quantities via an expression with an argument list are available. For details, see:

- User Routine Details View in the *CFX-Pre User's Guide*
- User Functions in the *CFX-Pre User's Guide*.

Details on creating shared libraries and compiling subroutines are available. For details, see Shared Libraries (p. 640).

Examples of using user CEL functions are available. For details, see User CEL Examples (p. 697).

19.1.1. Structure of User CEL Functions

A User Fortran file may contain several user routines that can be called from the CFX-Solver, as well as any secondary routines that are called only from other routines in this file.

In addition to any comments and declarations that you may want to add, the basic structure of a user CEL function is:

```
#include "cfx5ext.h"
dllexport(<callingname>)
      SUBROUTINE <callingname>(
      &  NLOC, NRET, NARG, RET, ARGS, CRESLT, CZ,DZ,IZ,LZ,RZ )
C
```

```
      INTEGER NLOC,NARG,NRET
      CHARACTER CRESLT*(*)
      REAL ARGS(NLOC,NARG), RET(NLOC,NRET)
C
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
      .... executable statements
      END
```

The `dllexport()` macro is used to ensure that a calling name is known externally on those platforms that require it for successful run-time linking. The macro is defined in an include file so `#include "cfx5ext.h"` should be the first line of the Fortran file.

One `dllexport()` should be specified for every routine that the CFX-Solver can call in the Fortran file, and each `dllexport()` macro must precede the SUBROUTINE statement that it refers to and must start in column 1. The argument of the `dllexport` macro should be the name of the subroutine in lower case and should not contain spaces.

User CEL functions have a fixed argument list that contains the following data fields:

- NLOC: Number of locations in space over which the calculations have to be performed.
- NARG: Number of arguments passed to the function.
- ARGS(1:NLOC,1:NARG): Arguments passed to the function (at each point in space).
- NRET: Number of return variables. This is always 1 in CFX, but is included to enable future extensions.
- RET(1:NLOC,1:NRET): Return variables (at each point in space).
- CZ(*), DZ(*), IZ(*), LZ(*), RZ(*): CHARACTER, DOUBLE PRECISION, INTEGER, LOGICAL and REAL stacks.

The length (NLOC) of the arguments (ARGS) and the return value (RET) of user CEL functions is determined by the locale for which the routine is called. For example, for a boundary element group, NLOC is the number of faces in the group, and for vertices, NLOC is the number of vertices in the current zone.

Note that, in general, your user CEL function will be called several times during each iteration and the value of NLOC will be different for each call. This is because the CFX-Solver will split the specified region (for example, a boundary condition region) into a number of smaller 'pieces' and call your function for each piece. Your user subroutine should be coded to deal with this.

The stacks are required if the information specified on the right side of the CEL expression (for example, `B*C` and `D` in `A = UR(B*C, D)`) is not sufficient to calculate A. It might be necessary to pick up additional data (such as user input data, data at other locales, gradients, and so on). For details, see Utility Routines for User Functions (p. 647). This data is accessed from the CFX Memory Management System and requires the global stacks; therefore, the global stacks are added to the argument list. For details, see CFX Memory Management System (MMS) (p. 678).

A template user CEL function Fortran file named `ucf_template.F` can be found in `<CFXROOT>/examples/`.

Note that all strings used in User Fortran are case-sensitive.

19.1.2. User CEL Function Units

On entry into a user CEL function routine, the arguments are automatically converted into the units specified in the **Argument List** list in the **User Function Editor** (labeled `Argument List` in the definition for the function in the CCL file `LIBRARY` section).

On exit, the results are automatically converted from the **Result Units** into the solution units used by the CFX-Solver.

This ability to choose the working units for the routine with automatic conversion may be useful for creating interfaces between the CFX-Solver and third-party data or applications.

19.2. User Junction Box Routines

In addition to defining your own CEL functions, you can call user subroutines at several points during execution of the CFX-Solver; such subroutines are called junction boxes. Junction boxes can be used to accomplish a number of different tasks:

- Input / Output of user data.
- Advanced solution monitoring tasks.
- Coupling of CFX with other software packages.
- Program control (for example, control of the termination criteria of coefficient loops in a transient run by some other means than maximum number of iterations or residual targets).

User defined junction box routines are passed only to the CFX-Solver workspace stacks as arguments. Unlike user CEL functions, there are no user arguments for these routines. However, by using the CFX MMS (Memory Management System) utilities, advanced users have full access to the internal data structures of the CFX-Solver. For details, see CFX Memory Management System (MMS) (p. 678). Some simplified routines have been implemented to enable access to commonly used data in the MMS. For details, see Utility Routines for User Functions (p. 647).

CEL variables are not available for use in junction box routines.

You can call more than one junction box during a run.

19.2.1. Junction Box Routine Options and Parameters

A junction box routine is created in CFX-Pre so that the CFX-Solver knows where to find the Fortran subroutine and when it should be called. The parameters that must be specified are described below. For details, see Junction Box Routines in the *CFX-Pre User's Guide*.

The parameters that must be set for the `USER ROUTINE` CCL object are similar to the parameters used for User CEL Functions; however, an additional `Junction Box Location` CCL parameter specifies at which point the junction box routine is called during execution of the CFX-Solver.

The following parameters have to be set for the `USER ROUTINE` CCL object to fully declare a junction box routine.

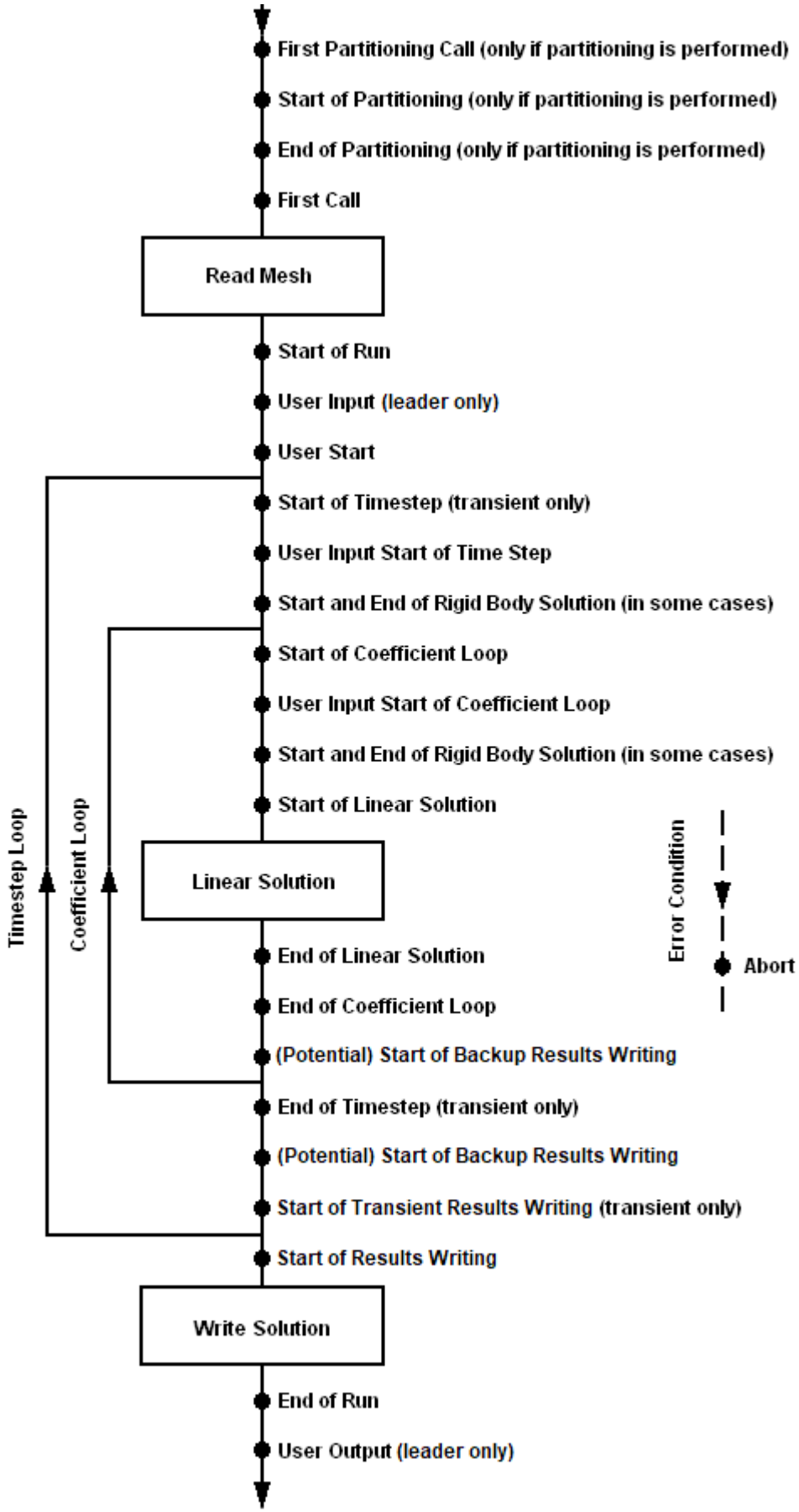
- Calling Name: Is the name you use internally in your subroutine (for example, `mysub` for SUB `ROUTINE MYSUB(...)`). Note that this should be in lower case even when it is in upper case in the Fortran file.
- Library Name: Is the shared library name (your Fortran filename by default). For details, see Shared Libraries (p. 640).
- Library Path: Is the path to the directory that contains the final shared library in subdirectories for each platform. For details, see Shared Libraries (p. 640).
- Junction Box Location: Is a list of locations in the simulation where a junction box routine has to be called. The locations listed below are possible and are illustrated in the diagram that follows:
  - **First Partitioning Call**: Only applicable if partitioning is performed. Called just before the mesh is read and before partitioning is performed.
  - **Start of Partitioning**: Only applicable if partitioning is performed. Called after reading the mesh but before the partitioning process.
  - **End of Partitioning**: Only applicable if partitioning is performed. Called just after the partitioning has been completed and the `.par` file is written.
  - **First Call**: Called just before the mesh is read and the physics and boundary condition initialization. Called after partitioning has been completed if applicable.
  - **Start of Run**: Called after the problem setup is completed and the mesh has been read in.
  - **End of Run**: Called just after the results file has been written.
  - **User Input**: Called after **Start of Run**. This is the recommended location for your Fortran input code that works in parallel as well as serial runs. For details, see Reading Data with the User Input Option (p. 639).
  - **User Start**: Called after **User Input** on all partitions.
  - **User Output**: Called after **End of Run**. This is the recommended location for your Fortran output code that works in parallel as well as serial runs. For details, see Writing Data with the User Output Option (p. 639).
  - **Start of Timestep**: Called at the start of each timestep. This applies only for transient runs and relates to the outer loop.
  - **User Input Start of Timestep:** Called at the start of each timestep. Works identically to the User Input location but is called at the beginning of each timestep in a transient run.
  - **End of Timestep**: Called at the end of each timestep. This is applicable only for a transient run and relates to the outer loop.
  - **Start of Coefficient Loop**: Called at the start of each iteration loop. For a transient run, this relates to the inner loop.
  - **User Input Start of Coefficient Loop**: Called at the start of each coefficient loop. Works identically to the User Input location but is called at the beginning of each coefficient loop in a transient run and steady-state run.

- **End of Coefficient Loop**: Called at the end of each iteration loop. For a transient run, this relates to the inner loop.
- **Start of Linear Solution**: Called before the linear solver for a particular set of equations.
- **End of Linear Solution**: Called after the linear solver for a particular set of equations.
- **Abort**: Called if the Solver stops prematurely for some reason.
- **Start of Rigid Body Solution**: Called at the start of the rigid body solution. Only applicable for cases with a rigid body enabled. The rigid body solver is called in accordance with the rigid body solver control settings. For details, see Rigid Body Control Tab in the *CFX-Pre User's Guide*.
- **End of Rigid Body Solution**: Called at the end of the rigid body solution.
- **Start of Backup Results Writing**: Called just before writing a backup results (`.bak`) file. Such files are written at the end of a coefficient loop or at the end of a timestep loop, depending on the output control settings.

  This junction box location might be useful for writing restart information needed by other User Fortran routines.
- **Start of Transient Results Writing**: Called just before writing a transient results (`.trn`) file.

  This junction box location might be useful for writing restart information needed by other User Fortran routines.
- **Start of Results Writing**: Called just before writing the CFX-Solver results (`.res`) file.

  This junction box location might be useful for writing restart information needed by other User Fortran routines.



19.2.2. Calling Junction Box Routines

Many junction box routines may be defined, but only those selected on the **Solver Control** panel will be called during execution of the CFX-Solver. For details, see Basic Settings Tab in the *CFX-Pre User's Guide*.

19.2.3. Reading Data with the User Input Option

The **User Input** option for Junction Box Location enables you to read in your own data stored in files using arbitrary data formats. An example of data you may want to read in is any kind of data table (for example, profile boundary condition) when the data is read during a CFX-Solver run, a special MMS data area called /USER_DATA is created before the user input call. In a parallel run, this call is made only on the leader process. After the call, any data found in /USER_DATA will be automatically copied from the leader to the follower processes, so it becomes available to all subroutines on all processes. Use of this MMS data area requires calls to basic CFX MMS routines. For details, see CFX Memory Management System (MMS) (p. 678).

In a parallel run, **Start of Run** calls are made on every partition, while **User Input** calls are made only by the leader process.

19.2.4. Writing Data with the User Output Option

The **User Output** option for junction box location enables you to write out data at the end of a run (advanced solution monitoring in CFX can be used in many cases to achieve tasks where user output would have previously been needed). For parallel runs, user output becomes more complicated. Unlike the automatic communication on the **User Input** call, the **User Output** location has no built-in facility for collecting data from follower processes, because it is difficult to anticipate which data will be output. With the exception of integrated quantities (which can be written out with the advanced monitoring facility), all data is stored locally on each partition.

19.2.5. Other Junction Box Location Options

Junction box routines called at the beginning or end of timesteps or coefficient loops are expected to be mainly used for custom solution monitoring, program steering or coupling with other applications. A good example application is the coupling of CFX with a structure analysis code to calculate the fluid-structure interaction.

The **Abort** option for Junction Box Location is a special call that occurs just before the CFX-Solver terminates abnormally (for example, if an MMS error occurs). It is designed to enable you to shut down programs that might be running in conjunction with the CFX-Solver or other system level operations required as the CFX-Solver process ends. The CFX-Solver exit status is provided in the common block **/CFXSTATUS/ STATUS**. **STATUS** is an integer.

19.2.6. Structure of the User Junction Box Routines

Junction box routines have a fixed argument list that contains only the global stacks; no further arguments can be passed to them. Therefore, the argument and unit information that is required for user CEL functions is not required for junction box routines. Junction box routines always work in base units, as specified in the FLOW > SOLUTION UNITS section of a CCL file or on the **Solution Units** form in CFX-Pre. For details, see Setting the Solution Units in the *CFX-Pre User's Guide*.

MMS (Memory Management System) calls can be used in junction box routines to access CFX-Solver data. For details, see CFX Memory Management System (MMS) (p. 678).

A template junction box routine, `jcb_template.F`, is in `<CFXROOT>/examples/`.

19.2.7. Which Call

Because the same junction box routine can be called from several locations in the same CFX-Solver run, it may be useful for the routine to know which location currently applies. This can be determined by looking up the variable `Which Call` as if this were a user parameter. For example, a junction box routine may contain:

```
      CHARACTER*256 Which Call
      CHARACTER*4 CRESLT
      CHARACTER*1 CZ(*)
      ....
      ....
      CALL USER_PEEKCS( 'Which Call', Which Call,
      &   'STOP', CRESLT, CZ )
      IF( Which Call .EQ. 'USERINPUT' ) THEN
      ....
      END IF
```

Note that the value returned by the enquiry is in upper case even though the junction box locations were specified in lower case. For details, see User Parameters (p. 642).

19.3. Shared Libraries

The shared library concept enables you to link to user subroutines during a CFX-Solver run and avoids the need to create custom binaries each time user subroutines are used to extend the functionality of CFX. They also have advantage that:

- Different applications can reuse the same common library or set of libraries.
- One application can use several libraries or alternative libraries.

As has been shown in the previous sections, both user CEL functions and junction box routines declared in the CCL file must specify a library name, a unique calling name and a path to the shared library.

19.3.1. Creating the Shared Libraries

A script called `cfx5mkext` is used to create shared libraries. You can run `cfx5mkext -help` for information on the arguments accepted by the script. To create the shared libraries, run at the command line:

```
cfx5mkext myroutine1.F myroutine2.F
```

This creates an architecture-specific directory in the current working directory. This directory contains the shared object library `myroutine1.dll` (Windows). A single library is created for all files passed

to `cfx5mkext`. By default, the library is named using the first file specified in the `cfx5mkext` command.

Important:

Use a capital F as the file extension for Fortran source to be run through the preprocessor.

If you are using one of the double precision CFX-Solver executables, you should use the `-double` option with `cfx5mkext` to compile your subroutines. For details, see Double-Precision Executables in the *CFX-Solver Manager User's Guide*.

If you want to create shared libraries for "large problem" executables, use the `-large` option. For details, see Large Problem Executables in the *CFX-Solver Manager User's Guide*.

You can specify a different name for the shared object library using the `-name` option. The `-dest` option can be used to set the location of the library. For example,

```
cfx5mkext -name UserRoutine -dest /home/user/SharedLibraries myroutine.F
```

creates `UserRoutine.dll` (for Windows) in an architecture-specific subdirectory in `/home/user/SharedLibraries`.

When specifying the library name in CFX-Pre (or in a CCL file), any prefix or suffix (`lib`, `.so`, `.sl`, `.dll`) added to the name should not be included.

When specifying the library path in CFX-Pre (or in a CCL file), you should specify an absolute path to the directory that contains the architecture-specific subdirectories (for example, `/home/user/SharedLibraries` in the previous example. On Windows systems, this might be `F:\user\SharedLibraries`). Library path lists are supported to enable distributed parallel runs across different architectures. For details, see Library Name and Library Path in the *CFX-Pre User's Guide*.

Note:

The double-precision solver uses 'auto promotion' of REAL data to DOUBLE. Specifically, when you include the `-double` option in your `cfx5mkext` command line, this activates auto-promotion and gives you double-precision User Fortran.

19.3.2. Default Fortran Compilers and Compiler Options

The `cfx5mkext` command requires a Fortran compiler to be in your path. The commands given in the following table are the defaults used to execute the compiler on each platform.

Platform	Command
winnt-amd64	ifort
linux-amd64	ifort

If the commands shown do not execute the Fortran compiler on your platform, then you will have to manually edit a copy of the *<CFXROOT>*`/etc/cfx5mkext.ccl` file and place it in the working

directory. The `cfx5mkext` command looks for a copy of `cfx5mkext.ccl` in the current working directory first. Further instructions are provided at the top of the file `cfx5mkext.ccl` file.

Compiler options (flags) are specified in `cfx5mkext.ccl`.

For all compilers (ifort, gnufort and pgfort) there is a compiler flag that specifies that no implicit typing is allowed, which is the equivalent of having the FORTRAN statement "IMPLICIT NONE" at the start of every procedure.

If you are using the Intel FORTRAN compiler (ifort), and you do not want to declare all local variables, you can enable implicit typing by editing `cfx5mkext.ccl`, removing the following compiler options:

- (winnt-amd64)

  ```
  -warn:declarations -Qdiag-error:6717
  ```

- (linux-amd64)

  ```
  -warn declaration -diag-error 6717
  ```

The required compiler versions are given in Compiler Requirements for All Windows Versions in the *Ansys, Inc. Installation Guides* and Compiler Requirements for All Linux Versions in the *Ansys, Inc. Installation Guides*.

19.4. User Parameters

User Fortran using shared libraries is more versatile if its key parameters can be changed between runs without the need for recompilation or maintenance of different versions of the routines.

For example, the same User Fortran source code may be run for different applications or for parametric studies, but each run may need to open a different filename for reference data, or use a different value for some model option or a model parameter. Such parameters may be passed to the User Fortran through the command language in the CFX-Solver input file or the results file. In a parallel run, each partition gets a copy of this data. A demonstration on the use of some user parameters is available in Junction Box Example 1: Profile Boundary Conditions (p. 706).

19.4.1. Adding and Modifying User Parameters

There is a `USER:` object in command language for automatically passing user parameters to the solver (for access by User Fortran routines). Any parameter may be defined inside this object by simply adding statements of the form `name = value` within the `USER:` object.

The `USER:` object and parameters can be added to a CFX-Solver input file using `cfx5cmds`. For details, see Editing the Command Language (CCL) File in the *CFX-Solver Manager User's Guide*. Once user parameters have been entered into the problem definition, they can also be seen and modified in the **Command File** editor.

19.4.2. Parameter Names

The name of a user parameter can be arbitrary, but subject to certain rules.

- Names in command language should start with an alphabetic character followed by alphanumeric characters.
- They are allowed to contain embedded spaces.
- Underscores are reserved for future syntax and, as such, are not allowed.
- The name "Workbench Path Parameters" is a reserved name. For details, see User Fortran in Ansys Workbench (p. 697).

However, to enable access to the parameter on the solver side by using the same name, you should ensure that the names of user parameters contain only single spaces and are unique in the first 20 characters.

19.4.3. Parameter Values

The value part of the parameter statement is a string that conforms to command language rules (leading and trailing spaces are ignored; a backslash character at the end of a line may be used for continuation) and which can be interpreted by the user routine.

Standard utilities are provided for interpreting a parameter value as a simple Fortran-style INTEGER, REAL, DOUBLE PRECISION, or LOGICAL value, or as a CHAR string, or as a list of any of these types. These utilities expect list items to be separated by commas.

Note that these standard utilities do not support the processing of units.

19.4.4. Example of CCL File User Parameters

```
USER:
 #
 # location of external inlet profiles.
 InputFile = /home/cfxuser/u_profile.dat
 #
 # printing control switch.
 User Printing = .FALSE.
 #
 # model option.
 ModelOption = 1
 #
 # model data.
 InletTemperature = 300
 #
 RadiusList = 0, 0.2, 0.4, 0.6, 0.8, 1
 VelocityList = 1, 0.96, 0.84, 0.64, 0.36, 0
END
```

19.4.5. Looking up a String Value

A User Fortran routine can pick up any user parameter value as a simple string, by calling the utility:

```
      CALL USER_PEEKCS( CDANAM, VALUE, CERACT, CRESLT, CZ )
```

Input Arguments:

- `CHARACTER*(*) CDANAM` - solver name of parameter.

- CHARACTER*4 CERACT - error action:
  - STOP to stop on error.
  - SKIP to continue silently.
  - WARN to continue after warning.
- CHARACTER*1(*) CZ - solver stack for CHARACTER data.

Output Arguments:

- CHARACTER*(*) VALUE - value of parameter.
- CHARACTER*4 CRESLT - result code:
  - GOOD if successful.
  - BIG if data string is longer that length of VALUE variable.

19.4.6. String Value Example

```
      CHARACTER*(80) InputFile
      CHARACTER*4    CRESLT
      CHARACTER*1    CZ(*)
C
C Look up command file parameter InputFile
C The run will stop if the data has not been provided.
C
      CALL USER_PEEKCS(
     & 'InputFile', InputFile, 'STOP', CRESLT, CZ )
```

19.4.7. Looking Up List Values

The USER: data arriving at the CFX-Solver may be interpreted either as a single value, or as a list of values separated by commas. If there is only a single value, the following utilities regard this as a list that is one item long.

Each individual value of a list may be looked-up as data of type CHARACTER, DOUBLE PRECISION, LOGICAL, INTEGER or REAL by calling the appropriate utility and specifying the position of the item in the list.

```
      CALL USER_PEEKCA(
     & CDANAM, JADRES, CVALUE, CERACT, CRESLT, CZ )
      CALL USER_PEEKD(
     & CDANAM, JADRES, DVALUE, CERACT, CRESLT, CZ )
      CALL USER_PEEKL(
     & CDANAM, JADRES, DVALUE, CERACT, CRESLT, CZ )
      CALL USER_PEEKI(
     & CDANAM, JADRES, IVALUE, CERACT, CRESLT, CZ )
      CALL USER_PEEKR(
     & CDANAM, JADRES, RVALUE, CERACT, CRESLT, CZ )
```

Input arguments:

- CHARACTER*(*) CDANAM - name of parameter.

- INTEGER JADRES - parameter list index, for example, 1 for a simple value or for the first value in the list.
- CHARACTER*4 CERACT - error action.
  - STOP to stop on error.
  - SKIP to continue silently.
  - WARN to continue after warning.
- CHARACTER*1(*) CZ - solver stack for CHARACTER data.

Output arguments:

- CHARACTER*(*) CVALUE - value of string parameter.
- DOUBLE PRECISION DVALUE - value of double precision parameter.
- LOGICAL LVALUE - value of logical parameter.
- INTEGER IVALUE - value of integer parameter.
- REAL RVALUE - value of real parameter.
- CHARACTER*4 CRESLT - result code.
  - GOOD if successful.
  - ADRS if JADRES is out of bounds.

19.4.8. Real Value Example

```
      REAL InletTemperature
      CHARACTER*4 CRESLT
      CHARACTER*1 CZ(*)
C
C Set a default value
      InletTemperature = 298.0
C
C Look up command file parameter InletTemperature
C but continue with default value if none provided.
C Only one value is expected, so specify JADRES=1.
      CALL USER_PEEKR(
     & 'InletTemperature', 1, InletTemperature, 'SKIP', CRESLT, CZ )
```

19.4.9. Looking up Sizes of Lists and Strings

If the length of items or number of items in a list could be variable, then these sizes can be determined by the following utility.

```
      CALL USER_PEEK_SIZE(CDANAM, NSIZE, LENVEC, CERACT, CRESLT, CZ )
```

Input arguments:

- CHARACTER*(*) CDANAM - solver name of parameter.

- CHARACTER*4 CERACT - error action.
  - STOP to stop on error.
  - SKIP to continue silently.
  - WARN to continue after warning.
- CHARACTER*4 CRESLT - result code.
  - GOOD if successful.
- CHARACTER*1(*) CZ - solver stack for CHARACTER data.

Output arguments:

- INTEGER NSIZE - number of items in list.
- INTEGER LENVEC - number of characters in the longest item, ignoring any leading spaces

19.4.10. Real List Example

```
      REAL RadiusList(100), VelocityList(100)
      INTEGER NSIZE, LENVEC
      CHARACTER*4 CRESLT
      CHARACTER*1 CZ(*)
      INTEGER ITEM
C
C Initialization
      DO ITEM = 1, NSIZE
         RadiusList(ITEM) = 0.0
         VelocityList(ITEM) = 0.0
      END DO
C
C Assume VelocityList and RadiusList have same,
C but arbitrary, number of items, up to a limit of 100.
C
C Determine number of items, NSIZE
      CALL USER_PEEK_SIZE(
     & 'RadiusList', NSIZE, LENVEC, 'STOP', CRESLT, CZ )
C
C Lookup VelocityList(1:NSIZE) and RadiusList(1:NSIZE)
      DO ITEM = 1, NSIZE
         CALL USER_PEEKR( 'RadiusList',
     &    ITEM, RadiusList(ITEM), 'SKIP', CRESLT, CZ )
         CALL USER_PEEKR( 'VelocityList',
     &    ITEM, VelocityList(ITEM), 'SKIP', CRESLT, CZ )
      END DO
```

19.4.11. Printing Parameters

User Fortran code that needs to output results to file can use normal Fortran output statements, provided that these are coded in junction box routines called from the user output location. This enables the same code to use in serial and parallel.

However, it can also be useful for other user routines to be able to print out key parameters elsewhere during a run. Typical uses might be for checking and recording control settings or for observing run progress.

Some simple printing utilities are provided that are valid for both serial and parallel runs. These send text via the leader process to the output file and at the same time to the console.

The following utilities print messages of the form `name = value`:

```
      CALL USER_PRINT_CHAR( NAME, VALUE )
      CALL USER_PRINT_DBLE( NAME, VALUE )
      CALL USER_PRINT_INTR( NAME, VALUE )
      CALL USER_PRINT_LOGL( NAME, VALUE )
      CALL USER_PRINT_REAL( NAME, VALUE )
```

There are two input arguments.

- `NAME` is a label of type `CHARACTER*(*)`.
- `VALUE` is a Fortran constant or variable of type `CHARACTER*(*)`, `DOUBLE PRECISION`, `INTEGER`, `LOGICAL` or `REAL` as indicated by the utility name.

The above utilities make use of the following CFX-Solver routine for outputting a general string

```
      CALL MESAGE( 'WRITE-ASIS', STRING )
```

where `STRING` is data of type `CHARACTER*(*)`.

This `MESAGE` routine can be used for constructing other forms of message from User Fortran that will also be valid in both parallel and serials runs.

19.5. Utility Routines for User Functions

The following topics will be discussed:

- Introduction to Utility Routines for User Functions (p. 647)
- Data Acquisition Routines (p. 648)
- Character Handling (p. 674)
- Output Routines (p. 677)

19.5.1. Introduction to Utility Routines for User Functions

The main memory stacks are passed to both user CEL Functions and junction box routines providing you with full access to the CFX MMS, along with some useful MMS utility routines. For details, see CFX Memory Management System (MMS) (p. 678).

To enable easy access to some of the more frequently used data in the MMS, a set of simplified routines that can be called from within user CEL function and/or user junction box subroutines have been provided.

A routine called USER_GETVAR provides easy access to gradients on the given locale of a user CEL function. This is documented below.

Locale information can be accessed from within a user CEL function routine using the subroutine. For details, see USER_CALC_INFO (p. 657).

Scalar mesh information (for example, total numbers of vertices and elements) can be accessed using the USER_GET_MESH_INFO routine. For details, see USER_GET_MESH_INFO (p. 659). Arrays describing mesh data, such as vertex coordinates and element volumes, are accessed using the USER_GET_MESHDATA routine. For details, see USER_GET_MESHDATA (p. 662).

Physical data can be accessed using the USER_GET_PHYS_INFO routine. For details, see USER_GET_PHYS_INFO (p. 664). This includes, for example, information on numbers and types of fluids and solids, and details on physical models, such as turbulence models. This can be used from within both user CEL function routines and user junction box routines.

Equation assembly information can be accessed using the USER_ASSEMBLE_INFO routine. For details, see USER_ASSEMBLE_INFO (p. 668).

Parallel information can be accessed using the GET_PARALLEL_INFO routine. For details, see GET_PARALLEL_INFO (p. 669). This can be used from within both user CEL function routines and user junction box routines.

Throughout, arguments to subroutines are listed, for example as follows

```
CI CZONE        : Solver Zone Name   (blank for global information only)
CV CDIR         : Subdirectory name of /USER into which to put info.
CO CRESLT       : Result = ‘GOOD’, ‘ERR’ or ‘DIR’.
```

The abbreviations CI, CV, CO refer to input data, modified data, and output data respectively.

19.5.2. Data Acquisition Routines

19.5.2.1. USER_GETVAR

The main application for USER_GETVAR is to provide derived variables, which are not accessible with CEL or operations on primitive and derived variables, and to provide material properties data. USER_GETVAR can be called from within user CEL functions to obtain a pointer to the latest solution fields for the locations on which the subroutine is currently operating. Note that USER_GETVAR obtains or computes its own copy of the data, and the returned pointer cannot be used to change the value of the solution fields. Note also that USER_GETVAR is not relevant to junction box routines.

Calls to USER_GETVAR have the following form:

```
      CALL USER_GETVAR('<VARIABLE_DESCRIPTION>',CRESLT,
     &                  <pVAR>,CZ,DZ,IZ,LZ,RZ)
```

where the arguments in the argument list are as follows:

- <VARIABLE_DESCRIPTION>: name of the variable for which USER_GETVAR is called, following the naming convention:

  [<Field A>].[<Field B>].<Field C>.[<Field D>]

  where:

Field	Description
A	A valid name of one of the following objects: FLUID, SOLID, FLUID PAIR, or MUSIG FLUID
B	A valid name of one of the following objects: COMPONENT, SIZE GROUP, REACTION
C	A valid name of one of the following objects: VARIABLE, ADDITIONAL VARIABLE
D	One of the following variable operators: Magnitude, Gradient, Curl, Laplacian, Time Derivative, Transient Minimum, Transient Maximum, Transient Std Deviation, Transient RMS, Transient Average, Boundcon, magnitude, grad, curl, laplacian, dt, Trnmin, Trnmax, Trnsdv, Trnrms, Trnavg, hybrid

Note:

The Laplacian operator does not work in USER_GETVAR in some situation.

You should always add the fluid name even though, in some cases such as for the variable Pressure, it is not strictly necessary. The fluid name is required when specifying a phase-specific variable (in a multiphase case) and when specifying an algebraic Additional Variable (even for a single-phase case).

The names are case sensitive. You can use long variable names (for example, **Pressure**) or short names as they appear in CEL (for example, **p**).

You can also request whole vector and tensor entities (for example, Air at STP.Velocity) as well as components of these variables (for example, Air at STP.Velocity u).

Examples:

– Pressure: is the fluid pressure. Notice that no fluid tag is required.

– Pressure.Gradient: is the pressure gradient in the fluid.

– Water at RTP.Density: is the density of fluid Water at RTP

– Air.Nitrogen.Mass Fraction: is the mass fraction of component Nitrogen in the multi-component fluid Air

– Water.Pollutant.Mass Fraction.Gradient: is the gradient of the variable Water.Pollutant.Mass Fraction

– MyMusigFluid.Group1.Size Fraction: is the size fraction of size group Group1 in the MUSIG fluid MyMusigFluid

– Mixture.CO Oxidation.Molar Reaction Rate: is the molar reaction rate of the reaction CO Oxidation in the reacting fluid Mixture.

- CRESLT: returned result key word containing the status of the USER_GETVAR call. After a successful call to USER_GETVAR, this variable will be CRESLT='GOOD'. The possible return codes are:

CRESLT	Description
BADL	Variable cannot be computed on the current LOCALE
CIRC	Variable has a circular reference
CVAR	Variable solver name is not valid
ENTY	Variable cannot be compute on the current ENTITY
GOOD	Variable was successfully computed
NAME	Variable user name is not valid
RECP	Variable is valid but CFX-Solver could not find a formula to evaluate the variable
IGET	Variable is valid and a formula exists, but the recipe could not be computed

This variable MUST be checked after every call to USER_GETVAR and verified that it is equal to GOOD; otherwise, there has been a failure and the caller should continue or stop at their discretion.

- <pVAR>: stack pointer to the variable if the call was successful. Otherwise, its value is undefined and will result in a segmentation violation if used.
- `CZ`, `DZ`, `IZ`, `LZ`, RZ: stacks.

For example:

```
      CALL USER_GETVAR('Water at RTP.Pressure.Gradient',CRESLT,
     &                 pPGRADIENT,CZ,DZ,IZ,LZ,RZ)
      IF (CRESLT.NE. 'GOOD') STOP
```

Depending on the locale, the gradients are provided as vertex gradients (for example, for initial conditions `LOCALE='VERTICES'`), element averaged gradients (for example, for source terms `LOCALE='IELGn'`) or face averaged gradients (for example, for boundary conditions `LOCALE='BELGn'`). For `'IELGn'` and `'BELGn'`, USER_GETVAR gathers the specified variable to the vertices of the `IELG` or `BELG` and then use the shape functions to calculate the gradients.

The following restrictions apply to the use of gradients or curls:

- Gradients (or Curl) on `'VERTICES'` are approximate at wall boundaries. This is because wall values are taken from the interior.
- The normal component of gradients on `'VERTICES'` is always zero on symmetry planes.

19.5.2.1.1. Boundcon Operator

The "Boundcon" operator enables you to pick up boundary condition data for transport variables on boundary locales when applicable. By default, USER_GETVAR modifies the entity on a boundary locale to ensure that data can be obtained from the solution field, or a recipe if relevant. The use of this operator is targeted at experts as the precise effect is boundary condition, variable and equation dependent.

19.5.2.1.2. Variable Shape and Dimensions

The shape of the returned variable depends on the variable type and specified operation. The following syntax is used for non-gradients:

```
VAR(NCOMPT,NLOC)
```

with

- NCOMPT: number of tensor components of the variable. This is 1 for a scalar, 3 for a vector and 6 for a symmetric rank two tensor.
- NLOC: dimension of the locale.

For gradients and curls, the gradient/curl components appear first, with the tensor components appearing at the end, as follows:

```
VAR(NGRAD,NLOC,NCOMPT)
```

with

- NGRAD = 3: number of gradient components in the result.
- NLOC: dimension of the locale.
- NCOMPT: number of tensor components of the variable. This is 1 for a scalar, 3 for a vector and 6 for a symmetric rank two tensor.

This leads to the following definitions:

- Scalar variable: `RZ(pVAR)` points to a variable `VAR(NLOC)`.
- Vector variable: `RZ(pVAR)` points to a variable `VAR(3,NLOC)`.
- Gradient of a scalar variable: `RZ(pVAR)` points to a variable `VAR(3,NLOC)`.
- Gradient of a vector variable: `RZ(pVAR)` points to a variable `VAR(3,NLOC,3)`.

The locale is always the locale for which the User CEL routine is called. It is not possible to call USER_GETVAR for non-local information (for example, using the face gradients on `BOUNDARY:Inlet` for the boundary condition specification of a variable on `BOUNDARY:Outlet`).

19.5.2.2. USER_GET_GVAR

This utility enables the integrated global quantities, normally used in CEL, to be used from User Fortran. For example, one could obtain the mean pressure (over all partitions in the case of a parallel run).

Note:

This utility may fail in parallel, if the utility routine is not called on every partition.

The integrated function notation used is similar to that in CEL. USER_GET_GVAR may be called from user CEL or junction box routines. However, note the following differences in its operation in these two contexts:

- When called from a junction box, the requested integrated quantity is recalculated.
- When called from a User CEL routine, the requested integrated quantity is recalculated only if it has not previously been calculated and saved. If it has previously been calculated and saved, then the previously calculated value is returned.

These differences in operation are due to parallel processing issues. It is safe to recalculate on junction boxes, as all parallel processes are synchronized in that case. However, user CEL routines are used during local assembly, where it is not possible to recompute global quantities in parallel. If such a quantity is requested from a user CEL routine during a parallel run, the code may hang. For this reason, it is recommended that you call USER_GET_GVAR only from junction box routines. For user CEL routines, it is better practice to obtain the required global quantities from call back routines in CEL, and pass these as arguments to the user CEL routine. An alternative workaround is also to call the required global quantity from a junction box that is called once per coefficient loop. It is then safe to request the quantity from user CEL.

Examples of the use of USER_GET_GVAR are available in User Junction Box Examples (p. 706).

```
      SUBROUTINE USER_GET_GVAR (USER_VAR, USER_LOC, USER_OPER,
     &                       CRESLT, VAR, CZ,DZ,IZ,LZ,RZ)
C
CI USER_VAR:  User Name of variable or phase
CI USER_LOC:  User Location.
CI USER_OPER: Operation. e.g. 'maxVal'
C
CO CRESLT: Result, e.g. ‘GOOD’, ‘ERR’
```

Depending on the operation requested, USER_VAR may be specified as follows:

- USER_VAR = user variable name, for example, ‘Water at RTP.Temperature’
- USER_VAR = ‘ ‘ (blank), for example, for USER_OPER = ‘area’.

Depending on the operation requested, USER_LOC may specified as follows:

- USER_LOC = domain name, for example, ‘duct’
- USER_LOC = subdomain name, for example, ‘heated region’
- USER_LOC = boundary condition name, for example, ‘inlet’.

Allowed operations are tabulated and described below, together with allowed variable and location types.

USER_OPER	Allowed USER_VAR	Allowed USER_LOC	Description
area	blank	Any 2D region (such as a boundary or interface)	Area of a 2D region or injection region.
area_x area_y	blank	Any 2D region (such as a boundary or interface)	The component of the normal area vector in the X, Y, or Z direction.

USER_OPER	Allowed USER_VAR	Allowed USER_LOC	Description
area_z			
areaAve	variable	Any 2D region (such as a boundary or interface)	Area-weighted average of the variable on a 2D region.
areaInt	variable	Any 2D region (such as a boundary or interface)	Area-weighted integral of the variable over a 2D region.
ave	variable	Any 3D region (such as a domain or subdomain)	Arithmetic average of the variable within a 3D region.
force	blank	Any 2D region (such as a wall)	The magnitude of the force vector on a boundary[1].
force_x force_y force_z	blank	Any 2D region (such as a wall)	The component of the force vector in the X, Y, or Z direction[1].
mass	blank	Any 3D region (such as a domain or subdomain)	The total mass within a 3D region. This is fluid dependent.
massAve	variable	Any 3D region (such as a domain or subdomain)	Mass-weighted average of the variable on a 3D region.
massInt	variable	Any 3D region (such as a domain or subdomain)	The mass-weighted integration of the variable within a 3D region.
massFlow	blank	Any fluid surfaces (such as inlets and outlets)	Mass flow through a fluid surface.
massFlowAve	variable	Any fluid surfaces (such as inlets and outlets)	Mass flow weighted average of the variable on a fluid surface.
massFlowAveAbs	variable	Any fluid surfaces (such as inlets and outlets)	Absolute mass flow weighted average of the variable on a fluid surface.
massFlowInt	variable	Any fluid surfaces (such as inlets and outlets)	Mass flow weighted integration of the variable on a fluid surface.
maxVal	variable	Any 3D region (such as a domain or subdomain)	Maximum value of the variable within a 3D region.
minVal	variable	Any 3D region (such as a domain or subdomain)	Minimum value of the variable within a 3D region.

USER_OPER	Allowed USER_VAR	Allowed USER_LOC	Description
rbstate	variable long name for a rigid body state variable The rigid body state variables are: • Position, Position X, Position Y, Position Z • Linear Velocity, Linear Velocity X, Linear Velocity Y, Linear Velocity Z • Linear Acceleration, Linear Acceleration X, Linear Acceleration Y, Linear Acceleration Z • Euler Angle X,	Any rigid body object or any immersed solid domain that is governed by a rigid body solution.	Returns the position/velocity/acceleration or orientation/angular velocity/angular acceleration (or axis components of these) of a rigid body object or immersed solid that is governed by a rigid body solution. These quantities are with respect to the global coordinate frame.

USER_OPER	Allowed USER_VAR	Allowed USER_LOC	Description
	Euler Angle Y, Euler Angle Z • Angular Velocity, Angular Velocity X, Angular Velocity Y, Angular Velocity Z • Angular Acceleration, Angular Acceleration X, Angular Acceleration Y, Angular Acceleration Z For variables that do not include an axis name, the magnitude of the vector is returned. For example, if the variable is Position, the distance from the origin is returned.		

USER_OPER	Allowed USER_VAR	Allowed USER_LOC	Description
rmsAve	variable	Any 2D region (such as a boundary or interface)	RMS average of the variable within a 2D region
sum	variable	Any 3D region (such as a domain or subdomain)	Sum of the variable over all 3D region vertices.
torque	blank	Any 2D region (such as a wall)	Magnitude of the torque vector on a 2D region.
torque_x torque_y torque_z	blank	Any 2D region (such as a wall)	Magnitude of the torque vector in the X, Y, or Z direction.
volume	blank	Any 3D region (such as a domain or subdomain)	The total volume of a 3D region.
volumeAve	variable	Any 3D region (such as a domain or subdomain)	Volume-weighted average of the variable on a domain.
volumeInt	variable	Any 3D region (such as a domain or subdomain)	Volume-weighted integration of the variable within a domain or subdomain.

1. When restarting from an initial value file, USER_GET_GVAR will return only the normal part of the force when called before or at the start of the first iteration (the first coefficient loop of the first timestep for a transient simulation). For all subsequent iterations the force returned will include both the normal and the tangential parts, that is, the solution converges to the correct result. The restart kink can be resolved by setting expert parameter "read boundary flows = t".

19.5.2.2.1. USER_GET_GVAR with Multiphase Flow

If the USER_OPER is fluid-specific, various behaviors are possible depending on the USER_OPER type:

- For `massFlow` and `massFlowAve`, if the fluid prefix is not specified for the USER_OPER, then the bulk mass flows will be used
- For other fluid-specific functions, if a fluid-specific variable is specified and no fluid is specified for the USER_OPER, then the fluid specified for the variable will be assumed for the USER_OPER as well.
- For a fluid-specific USER_OPER that is specified, with a variable that is fluid-specific, and no fluid is specified for the variable, then the fluid specified for the USER_OPER will be assumed for the variable as well.
- If either the USER_OPER or variable are fluid-specific, and a prefix is not given for either, the solver will stop with an error.

USER_GET_GVAR multiphase examples are shown in the following table:

CEL Function	USER_OPER	USER_VAR	USER_LOC	Behavior
massFlow()@inlet	massFlow	blank	inlet	Bulk mass flow rate through inlet
Air.massFlow()@inlet	Air.massFlow	blank	inlet	Air mass flow rate through inlet
massFlowAve(p)@inlet	massFlowAve	p	inlet	Bulk mass flow averaged pressure on inlet
Air.massFlowAve(p)@inlet	Air.massFlowAve	p	inlet	Air mass flow averaged pressure on inlet
massFlowAve(vf)@inlet	massFlowAve	vf	inlet	Error because no fluid specified
massFlowAve(Air.vf)@inlet	massFlowAve	Air.vf	inlet	Bulk mass flow averaged air volume fraction on inlet
Air.massFlowAve(Air.vf)@inlet	Air.massFlowAve	Air.vf	inlet	Air mass flow averaged air volume fraction on inlet
Air.massFlowAve(vf)@inlet	Air.massFlowAve	vf	inlet	Same as Air.massFlowAve(Air.vf) @ inlet

19.5.2.3. USER_CALC_INFO

A simple routine, USER_CALC_INFO is provided for obtaining mesh locale information from within any user CEL function (it is not available from within junction box routines).

```
      SUBROUTINE USER_CALC_INFO (WHO,ACTION,CVAR,LOCALE,ENTITY,WHEN,
     & CALIAS,CERACT,CRESLT,
     & CZONE,ILOCS,ILOCF,IENTS,IENTF,
     & CZ,DZ,IZ,LZ,RZ)
C
CD Controls setting and getting of user data area and locale info.
C
CO WHO          : Name of routine making original call to GETVAR.
CI ACTION       : 'SET', 'GET' & 'RELEASE'
CO CVAR         : Solver Name of variable to be calculated by CEL call.
CO LOCALE       : Solver Locale (vertices, elements, element set etc.)
CO ENTITY       : Solver Entity (for element locales: integration point, etc.)
CO WHEN         : The time & iteration for evaluation.
CO CALIAS          : External variable name. Usually user provided for mass
C                     :  fractions and additional variables.
CO CERACT       : Action to be taken on an error occurring.
CO CRESLT       : Result of attempt to calculate variable.
CO CZONE        : Zone (if known) as context for AV links.
CO ILOCS        : Number of first item of locale type (e.g. vertex or
C                 element).
CO ILOCF        : Number of last item of locale type.
CO IENTS        : Number of first item of entity type (e.g. integration pt.)
CO IENTF        : Number of last item of entity type.
C                 For vertices, or ALL elements, IENTS=IENTF=1
```

This routine should be called only with ACTION set to 'GET'. Each argument is then set with the data described above. The arguments can be declared as follows in the calling routine:

```
CHARACTER*(MXDNAM) WHO, ACTION, CVAR, LOCALE, ENTITY, WHEN, CZONE
CHARACTER*(120) CALIAS
CHARACTER*(4) CERACT, CRESLT
INTEGER ILOCS, ILOCF, IENTS, IENTF
```

Note that for ACTION='GET', all other arguments are returned with output values, even CERACT, which in all other situations would be expected to be an input argument. Hence, do not attempt to code constant values for any arguments, except for ACTION.

Allowed solver LOCALE names are listed and described in the table below. *m*, *n* denote arbitrary integers:

LOCALE	**Description**
ZN*m*	zone *m* = domain *m*
VL*m*	volume *m*
VP*m*	volume patch *m* = subdomain *m* = a union of elementary volumes
VERTICES	all vertices on a zone
ELEMENTS	all elements on a zone
ELS*m*	element set *m* = set of elements of a fixed logical type
IELG*m*	interior element group *m* (a contiguous subset of an element set)
BCP*m*	boundary condition patch (BCP) *m* (a set of faces belonging to a boundary condition)
FCS*m*	face set *m* (a subset of a BCP consisting of faces of fixed logical type)
BELG*m*	boundary element group *m* = (a subset of elements of fixed logical type, coincident with a face set)

Allowed Solver ENTITY names are listed and described in the table below. *m*, *n* denote arbitrary integers:

ENTITY	**Description**
' '	not specified, for example, if LOCALE = 'VERTICES'
CENTRE = CENTER	centroids, for example, of elements on an element group, faces on a face set.
VXEL	element vertices, for example, on an interior or boundary element group.
IP	integration points, for example, on an interior or boundary element group.

19.5.2.4. USER_GET_MESH_INFO

This subroutine stores useful geometric and mesh related scalar data under a directory named '/USER/''/CDIR. It does so by looking inside /FLOW/GEOMETRY/ and /FLOW/MESH/, and copying required data. It is available in user CEL and junction box routines.

```
------SUBROUTINE USER_GET_MESH_INFO (WHO,ACTION,CERACT,
     &                   WHEN,CZONE,LOCALE,CDIR,CRESLT,
     &                   CZ,DZ,IZ,LZ,RZ)
CI WHO          : Name of routine making original call.
CI ACTION       :'GET' & 'DELETE'
CI CERACT       : Error Action = 'STOP', 'WARN', 'SKIP'.
CI WHEN         ; Only 'LATEST' supported at present.
CI CZONE        : Solver Zone Name   (blank for global information only)
CI LOCALE       : Locale Name (blank for zonal  information only)
CV CDIR         : Subdirectory name of /USER into which to put info.
CO CRESLT       : Result = 'GOOD', 'ERR' or 'DIR'.
```

The arguments are of type:

```
      CHARACTER*(*) WHO,ACTION,CEARCT,WHEN,CZONE,LOCALE,CDIR
      CHARACTER*(4) CRESLT
```

There are two ways in which you can choose a name for a directory under which to place the data. Either:

- Create a directory named CDIR below /USER, and pass the name down as an argument to USER_GET_MESH_INFO. In this case, the requested information is placed in the specified directory. For more information, see CFX Memory Management System (MMS) (p. 678) and Directories (p. 682).
- Alternatively, if CDIR does not exist under /USER/, for example CDIR = ' ', when passed down to USER_GET_MESH_INFO, then the program generates a name for a directory in which to place the data. The generated name is of the form MESH_INFO/Czone/Locale/. This directory name will be passed back in the argument CDIR.

If you choose the second alternative, then CDIR must be declared to be sufficiently long to hold this directory name. If not, then an error is generated, and CRESLT = 'DIR' is returned.

Setting ACTION = 'GET' copies required data into the specified directory name. Setting ACTION = 'DELETE' deletes the directory. The data under CDIR can be subsequently obtained using the PEEK facilities. For details, see:

- CFX Memory Management System (MMS) (p. 678)
- Setting and Reading Individual Values (p. 691).

Examples of use of USER_GET_MESH_INFO are available. For details, see USER_GET_MESH_INFO (p. 659).

The data copied depends on your setting of CZONE and LOCALE as follows.

19.5.2.4.1. Global Mesh Information: CZONE = ' ', LOCALE = ' '

If you specify blank zone and locale names, it is assumed that you require global mesh data. The information stored under CDIR is tabulated below:

Variable Name	Variable Type	Description
NZN	INTEGER	total number of zones = domains
CZONE(NZN)	CHARACTER * (MXDNAM)	character array containing internal solver names of zones (These are of the form 'ZNm', m = integer.)
NVX	INTEGER	total number of vertices
NEL	INTEGER	total number of elements
NELS	INTEGER	total number of element sets
NIELG	INTEGER	total number of internal element groups
NBELG	INTEGER	total number of boundary element groups

19.5.2.4.2. Zonal Information: CZONE = 'ZNm', LOCALE = ' '

If you specify a valid solver zone name, and a blank locale name, then it is assumed that you require mesh data specific to that zone. The information stored under CDIR is tabulated below:

Variable Name	Variable Type	Description
PHYTYPE	CHARACTER * (MXDNAM)	zone physical type = 'FLUID' or 'SOLID'
IZN	INTEGER	zone number
NDIM	INTEGER	number of spatial dimensions
NVL	INTEGER	number of volumes
NVP	INTEGER	number of volume patches = subdomains.
NSF	INTEGER	number of surfaces
NSP	INTEGER	number of surface patches
NBCP	INTEGER	number of boundary condition patches
CBCP(NBCP)	CHARACTER * (MXDNAM)	character array containing internal solver names of boundary condition patches. (These are of the form 'BCPm', m = integer.)
CVP(NVP)	CHARACTER * (MXDNAM)	character array containing internal solver names of volume patches = subdomains. (These are of the form 'VPm', m = integer.)
NVX	INTEGER	total number of vertices on this zone
NEL	INTEGER	total number of elements on this zone
NELS	INTEGER	total number of element sets on this zone
NIELG	INTEGER	total number of internal element groups on this zone
NBELG	INTEGER	total number of boundary element groups on this zone

19.5.2.4.3. Boundary Condition Patch Information: CZONE = 'ZNm', LOCALE = 'BCPn'

If you specify a valid solver zone name, and a valid Boundary Condition Patch (BCP) name, then it is assumed that you require mesh data specific to that BCP. The information stored under CDIR is tabulated below:

Variable Name	Variable Type	Description
PHYTYPE	CHARACTER * (MXDNAM)	BCP physical type = 'INLET' or 'OUTLET', 'OPENING', 'WALL', 'SYMMETRY', 'PERIODIC', 'INTERFACE'
IBCP	INTEGER	boundary condition patch number

19.5.2.4.3.1. Face Set Information: CZONE = 'ZNm', LOCALE = 'FCSn'

If you specify a valid solver zone name and a valid face set name, then it is assumed that you require mesh data specific to that face set. The information stored under CDIR is tabulated below:

Variable Name	Variable Type	Description
CBCP	CHARACTER * (MXDNAM)	name of BCP to which the face set belongs
IBCP	INTEGER	number of BCP to which the face set belongs
ISFC_S	INTEGER	starting set face number for this face set
ISFC_F	INTEGER	finishing set face number for this face set

19.5.2.4.3.2. Element Set or Element Group Information: CZONE = 'ZNm', LOCALE = 'ELSn' or 'IELGn' or 'BELGn'

If you specify a valid solver zone name, and a valid element set name, or element group name, then it is assumed that you require mesh data specific to that element set or element group. The information stored under CDIR is tabulated below:

Variable Name	Variable Type	Description
IELS	INTEGER	element set number for an element set or interior element group = 0 for a boundary element group
IELG	INTEGER	interior element group number = 0 if not an interior element group
BELG	INTEGER	boundary element group number = 0 if not a boundary element group
NVXEL	INTEGER	number of vertices per element.

Variable Name	Variable Type	Description
NFCEL	INTEGER	number of faces per element
NIPEL	INTEGER	number of integration points per element
IP1	INTEGER	starting integration point number
IP2	INTEGER	finishing integration point number
ISEL_S	INTEGER	starting set element number for an element set or an interior element group
ISEL_F	INTEGER	finishing set element number for an element set or an interior element group
ISFC_S	INTEGER	starting set face number for a boundary element group
ISFC_F	INTEGER	finishing set face number for a boundary element group

19.5.2.5. USER_GET_MESHDATA

This routine is similar to USER_GETVAR, in that it obtains arrays containing mesh related data. The subroutine looks for the requested variable on the /FLOW/MESH stack, and returns a pointer to it, and its size.

```
      SUBROUTINE USER_GET_MESHDATA (WHAT,WHERE,CACTION,CERACT,CRESLT,
     &                              NVAR,pVAR, CZ,DZ,IZ,LZ,RZ)
CI WHAT     : User name of variable.
CI WHERE    : User name of locale.
CI CACTION : Action = 'RETURN' or 'RELEASE'
CI CERACT  : Error action = 'STOP', 'SKIP' or 'WARN'
CO CRESLT  : Result of attempt to get variable, e.g. 'GOOD' or 'BADL'
CO pVAR     : Address in stack for the result.
CO NVAR     : Number of words in stack used for the variable.
```

The arguments are of type:

```
      CHARACTER*(*) WHAT, WHERE, CACTION, CERACT, CRESLT
      INETEGER NVAR
      __stack_point__ pVAR
```

USER_GET_MESHDATA is available only in user CEL routines. Also, the range of obtainable arrays depends on the current solver locale, which may be determined by a prior call. For details, see USER_CALC_INFO (p. 657). If you try to access a variable on an invalid solver locale, then USER_GET_MESH_INFO returns CRESLT = 'BADL'.

The table below lists permitted user mesh variable names and locations, and the permitted solver locales on which they may be obtained. It also describes the name, type, and shape of the associated solver variable that is returned. A description of the integer variables defining the array shape is available in USER_GET_MESH_INFO (p. 659). The values of these variables may be obtained by a prior call to USER_GET_MESH_INFO.

Note that, unlike USER_GETVAR, this routine returns a pointer to the actual solver array, not a copy of it.

WHAT	WHERE	Permitted Solver Locales	Solver Array Shape	Variable Type	Description
Coordinates	Vertices	Any	Crdvx (NDIM, NVX)	REAL	Coordinates of mesh vertices
Coordinates	Integration Points	ELS*, IELG*, BELG*	CrdIp (ISEL_S:ISEL_F, NDIM, IP1:IP2)	REAL	Coordinates of integration points
Coordinates	Faces	BELG*	CrdFc (ISFC_S:ISFC_F, NDIM)	REAL	Coordinates of faces
Volume	Vertices	Any	ContVol (NVX)	REAL	Control cell volumes
Volume	Elements	ELS*, IELG*, BELG*	VolEl (ISEL_S:ISEL_F)	REAL	Element volumes
Volume	Sectors	ELS*, IELG*, BELG*	VolSec (ISEL_S:ISEL_F, NVxEl)	REAL	Control cell sector volumes
Area	Faces	FCS*, BELG*	ArFc (ISFC_S:ISFC_F)	REAL	Areas of faces
Area	Integration Points	BELG*	ArIp (ISFC_S:ISFC_F, IP1:IP2)	REAL	Areas of sectors surrounding boundary integration points
Normal Area Vector	Integration Points	ELS*, IELG*, BELG*	NarvIp (ISEL_S:ISEL_F, NDIM, IP1:IP2)	REAL	Inward-directed normal area vectors at integration points
Normal Area Vector	Faces	BELG*	NarvFc (ISFC_S:ISFC_F, NDIM)	REAL	Inward-directed normal area vectors at faces
Normal Unit Vector	Integration Points	IELS*, IELG*, BELG*	NunvIp (ISEL_S:ISEL_F, NDIM,IP1:IP2)	REAL	Inward-directed normal unit vector at integration points

WHAT	WHERE	Permitted Solver Locales	Solver Array Shape	Variable Type	Description
Normal Unit Vector	Faces	BELG*	NunvIp (ISFC_S:ISFC_F, NDIM)	REAL	Inward-directed normal unit vector at faces
Near Wall Distance	Elements	BELG*	NWDIST (ISFC_S:ISFC_F)	REAL	Distance from element centroid to wall
Element-to-vertex Connectivity	Elements	ELS*, IELG*, BELG*	KVxEl (ISEL_S:ISEL_F, NVxEl)	INTEGER	List of vertex numbers connected to each element

Note:

Before Release 17.0, a call to Fortran routine USER_GET_MESHDATA to get the near-wall distance would return the near-wall distance multiplied by a constant. The value of the constant depended on the choice of turbulence model. Starting with Release 17.0, a call to USER_GET_MESHDATA to get the near wall distance returns the (unmodified) near-wall distance. Note that the near-wall distance is evaluated as the distance from an element's centroid to a boundary face, measured normal to the boundary face.

19.5.2.6. USER_GET_PHYS_INFO

This routine gets physics information on zones and phases, and stores under the /USER data area. This subroutine stores useful physical model data under a directory named '/USER/'//CDIR. It does so by looking inside /FLOW/PHYSICS/, and copying required data. It is available in user CEL and junction box routines.

```
      SUBROUTINE USER_GET_PHYS_INFO (WHO,ACTION,CERACT,
     &                   WHEN,CZONE,CPHASE,CDIR,CRESLT,
     &                   CZ,DZ,IZ,LZ,RZ)
CI WHO          : Name of routine making original call.
CI ACTION       :'GET' & 'DELETE'
CI CERACT       : Error Action = 'STOP', 'WARN', 'SKIP'.
CI WHEN         ; Only 'LATEST' supported at present.
CI CZONE        : Zone Name
CI CPHASE       : Phase Name (blank for zonal information only)
CV CDIR         : Subdirectory name of /USER into which to put info.
CO CRESLT       : Result = 'GOOD', 'ERR' or 'DIR'.
```

The arguments are of type:

```
      CHARACTER*(*) WHO,ACTION,CERACT,WHEN,CZONE,CPHASE,CDIR
      CHARACTER*(4) CRESLT
```

There are two ways in which you can choose a name for a directory under which to place the data. Either:

- Create a directory named CDIR below /USER, and pass the name down as an argument to USER_GET_PHYS_INFO. In this case, the requested information is placed in the specified directory. For more information, see CFX Memory Management System (MMS) (p. 678) and Directories (p. 682).
- Alternatively, if CDIR does not exist under /USER/, for example CDIR = ' ', when passed down to USER_GET_PHYS_INFO, then the program generates a name for a directory in which to place the data. The generated name is of the form PHYS_INFO/Czone/Locale/. This directory name will be passed back in the argument CDIR.

If you choose the second alternative, then CDIR must be declared to be sufficiently long to hold this directory name. If not, then an error is generated, and CRESLT = 'DIR' is returned.

Setting ACTION = 'GET' copies required data into the specified directory name. Setting ACTION = 'DELETE' deletes the directory. The data under CDIR can be subsequently obtained using the PEEK facilities. For details, see:

- CFX Memory Management System (MMS) (p. 678)
- Setting and Reading Individual Values (p. 691).

Examples of use of USER_GET_PHYS_INFO are available in the Junction Box 2 example. For details, see Junction Box Example 2: Integrated Residence Time Distribution (p. 714).

The information copied depends on your setting of CZONE and CPHASE, as described below.

19.5.2.6.1. Zonal Information: CZONE = 'ZNm', CPHASE = ' '

If you specify a valid solver zone name, and a blank phase name, then it is assumed that you require physical data specific to that zone. The information stored under CDIR is tabulated below:

Variable Names	Variable Type	Description
PHYTYPE	CHARACTER * (MXDNAM)	zone physical type = 'FLUID' or 'SOLID'
IZN	INTEGER	zone number
NPHASE	INTEGER	number of phases defined on the zone
PREF	REAL	reference pressure
TREF	REAL	reference temperature
GX, GY, GZ	REAL	components of gravity vector
CPHASE (NPHASE)	CHARACTER * (MXDNAM)	character array containing internal solver names of phases; of the form 'FLm', or 'SLm', m = integer, for fluid and solid phases respectively
CPP(NPP)	CHARACTER * (MXDNAM)	character array containing internal solver names of phase pairs; of the form 'PPm', m = integer

19.5.2.6.2. Phase Information: CZONE = 'ZNm', CPHASE = 'FLm' or 'SLm'

If you specify a valid solver zone name, and a valid phase name, then it is assumed that you require physical data specific to that phase on that zone. The information stored under CDIR is tabulated below:

Variable Names	Variable Type	Description
IPHASE	INTEGER	phase number
CMAT	CHARACTER * (MXDNAM)	internal solver name of material associated with this phase; of the form 'MTm', m = integer
CTURBM	CHARACTER * (MXDNAM)	internal solver name of turbulence model associated with this phase
CHEATM	CHARACTER * (MXDNAM)	internal solver name of beat transfer model associated with this phase
CBUOYM	CHARACTER * (MXDNAM)	internal solver name of buoyancy model associated with this phase
CCOMBM	CHARACTER * (MXDNAM)	internal solver name of combustion model associated with this phase
MORPHOLOGY	CHARACTER * (MXDNAM)	internal solver name of morphology associated with this phase

The table below translates internal solver names for physical models:

Variable Names	Variable Value	Description
CTURBM	' '	Laminar
CTURBM	'EVM:ALG:CX'	Baldwin-Lomax
CTURBM	'EVM:ALG:DIS'	Dispersed Phase Zero Equation model
CTURBM	'EVM:ALG:LES'	LES (Smagorinsky)
CTURBM	'EVM:K-E'	Standard k-ε model
CTURBM	'EVM:K-E:RNG'	RNG $k-\varepsilon$ model
CTURBM	'EVM:K-O'	Wilcox $k-\omega$ model
CTURBM	'EVM:K-O:SST'	Menter Shear Stress Transport (SST) model
CTURBM	'EVM:K-O:SST:DES'	DES=Menter Shear Stress Transport (SST) model
CTURBM	'EVM:K-O:BSL'	Menter Baseline (BSL) model
CTURBM	'EVM:EVT'	Menter Eddy Viscosity Transport model
CTURBM	'RSM:TED'	Reynolds stress model (Launder, Reece and Rodi)
CTURBM	'RSM:TED:SSG'	SSG Reynolds stress model
CTURBM	'RSM:TED:QI'	QI Reynolds stress model
CTURBM	'RSM:TEF'	Reynolds Stress-Omega model
CTURBM	'RSM:TEF:BSL'	Reynolds Stress-Omega model with BSL Omega equation
CHEATM	'ISOTHERMAL'	Isothermal.

Variable Names	Variable Value	Description
CHEATM	'THERMAL_H'	Thermal Energy model
CHEATM	'TOTAL_H'	Total Energy model
CBUOYM	'DEN*G'	Buoyancy Force = $\rho \boldsymbol{g}$
CBUOYM	'DENDIF*G'	Buoyancy Force = $\left(\rho - \rho_0\right) \boldsymbol{g}$
CBUOYM	'BOUSSINESQ'	Boussinesq approximation

19.5.2.7. USER_GET_TRANS_INFO

The USER_GET_TRANS_INFO routine provides access to time variables during transient simulations. It stores information in the '/USER/'//CDIR data area. It is available in user CEL and junction box routines.

```
      SUBROUTINE USER_GET_TRANS_INFO(WHO,ACTION,CDIR,CZ,DZ,IZ,LZ,RZ)
C
```

The last five arguments are the stacks as usual. All the other arguments are character strings and are defined as follows:

```
CI WHO          : Name of routine making original call.
CI ACTION       : 'GET' & 'RELEASE'
CC
CC
CV CDIR         : Sub-directory name of /USER into which to put info.
CC                If blank on entry, this is set equal to
CC                TRANS_INFO
CC                If non-blank on entry, it is unchanged, and treated
CC                as a user supplied directory name.
```

Calling this routine with `ACTION = 'GET'` creates and stores the transient information in `'USER/'//CDIR` whereas `ACTION = 'RELEASE'` deletes it. If `CDIR` is blank, the directory `'TRANS_INFO'` will be used.

After calling `USER_GET_TRANS_INFO` the following variables will be available in `'/USER/'//CDIR`:

Variable Names	Description
ATSTEP	Accumulated time step counter
CTSTEP	Current time step counter
ACLOOP	Accumulated coefficient loop counter
NCLOOP	Current coefficient loop counter
DT	Current time step size
ATIME_END	Accumulated simulation time at the end of time step
CTIME_END	Current simulation time at the end of time step

Variable Names	Description
ATIME_MID	Accumulated simulation time at the middle of time step
CTIME_MID	Current simulation time at the middle of time step
ATIME	Accumulated simulation time
CTIME	Current simulation time

By default, ATIME is the same as ATIME_END and CTIME is the same as CTIME_END, but they can be changed to the 'middle of timestep' values using the expert parameter 'time value option=2'.

For an example involving USER_GET_TRANS_INFO, see Junction Box Example 3: Timestep Control.

19.5.2.8. USER_ASSEMBLE_INFO

This is a utility to obtain information about the equation being assembled when a call to a User Fortran routine is made during the computation of a boundary condition or a source. The call is made as follows:

```
      CALL USER_ASSEMBLE_INFO(ACTION,CGROUP,CEQN,CTERM,CPVAR,
     &                        CLVAR,CPATCH,CRESLT,
     &                        CZ,DZ,IZ,LZ,RZ)
```

The last five arguments are the stacks as usual. All the other arguments are character strings and are defined as follows:

```
C --------------------
C Arguments
C --------------------
C
CI ACTION       : 'GET' or 'GET_USER'
C
CO CRESLT       : Result of attempt to return information. See details.
CO CGROUP       : Equation group being assembled.
CO CEQN         : Equation being assembled.
CO CTERM        : Term or term type being assembled.
CO CPVAR        : Principal variable of equation being assembled.
CO CLVAR        : Linearization variable - see details.
CO CPATCH       : Patch name.
C
C --------------------
C Details
C --------------------
C
C ACTION:-
C  = 'GET'       => Get all the field names that exists returned in
C      internal solver form.
C  = 'GET_USER'  => As for 'GET' except that user aliases are returned.
C
C CRESLT:-
C  If complete information can be provided then CRESLT = 'GOOD'
C  If partial information is provided then CRESLT = 'SOME'
C  Otherwise CRESLT = 'NONE' if no information can be found.
C  CRESLT = 'BIG' if actual argument string lengths are too short to
C  hold the data required. This argument should be declared to be *(4).
C Other output arguments:-
```

```
C  These should be declared to be *(MXDNAM) for ACTION='GET' and they
C  must be long enough to hold the user's names, a safe declaration is
C  *(MXLEN_XALIAS).
```

CTERM is returned with the term TYPE if ACTION = 'GET_USER' rather than the term name. Possible term types are: FTRANS.VOL, SOURCE.VOL or TRANS.VOL for volumetric terms and ADVECT.SUR, DIFFUS.SUR, FLUX.SUR, GRAD.SUR, LAPLACE.SUR, MSFLOW.SUR and STRESS.SUR.

CLVAR is the variable on which the term operates, or for a source term, it is the one of the variables used to linearize the term. Generally this will be identical to CPVAR but for coupled systems of equations it might be the principal variable of any of the equations in the system. For a deferred term, for example, the dP/dt term in the energy equation, it might be ANY variable.

An example of the use of USER_ASSEMBLE_INFO is available in User CEL Example 2: Using Gradients for an Additional Variable Source (p. 700).

19.5.2.9. GET_PARALLEL_INFO

The GET_PARALLEL_INFO routine provides access to information about the parallel run mode and partitioning details from within any user CEL function or from within junction box routines. It can be called in the following ways to look up the variables RUNMOD, PARMOD, OWNPAR and NPARD:

```
      CALL GET_PARALLEL_INFO ('RUNMOD',RUN_MODE,CNAME,CRESLT)
      CALL GET_PARALLEL_INFO ('PARMOD',PAR_MODE,CNAME,CRESLT)
      CALL GET_PARALLEL_INFO ('OWNPAR',OWN_PAR,CNAME,CRESLT)
      CALL GET_PARALLEL_INFO ('NPARD',NUM_PAR,CNAME,CRESLT)
```

- RUNMOD describes the run mode as either serial or parallel. It is an integer variable where 0 corresponds to serial and 1 to parallel.
- PARMOD describes the parallel mode and can be leader or follower. It is an integer variable where 1 corresponds to the leader process and 2 to a follower process.
- OWNPAR is an integer variable that gives the partition number on the current process.
- NPARD is an integer variable that gives the total number of partitions.

The variables RUN_MODE, PAR_MODE, OWN_PAR and NUM_PAR are used to store the values locally and should be declared as integer variables in your subroutine. They can be named differently to the names used here, but to avoid conflicts they should not be named RUNMOD, PARMOD, OWNPAR or NPARD. CNAME is a placeholder character variable and should be declared as such.

19.5.2.10. CAL_MESHMAP

This utility is used to generate and validate a mapping between an array of user-specified coordinates and the mesh coordinates within each domain.

```
      SUBROUTINE CAL_MESHMAP(CACTION,CZONE,UVX,IUVX_S,IUVX_F,
     &                       KUVXVX,NKUVXVX,CRESLT,
     &                       CZ,DZ,IZ,LZ,RZ)
CC
CC --------------------
CC        Input
CC --------------------
CC
```

```
CI  CACTION: 'CREATE' to create the KUVXVX mapping
CI           'CHECK' to perform basic validation of the map
CI  CZONE:   Solver zone name for which mapping between mesh and UVX
CI           coordinates is wanted
CI  UVX:     User/new coordinates for which mapping to mesh coordinates
CI           is wanted. Array dimensioning is UVX(3,IUVX_S:IUVX_F)
CI  IUVX_S/F: Start/finish index of current block of user coordinates
CC
CC --------------------
CC      Modified
CC --------------------
CC
CM  KUVXVX: Mapping between current mesh nodes and those contained in
CM          in the user array, UVX. Array dimensioning is KUVXVX(NKUVXVX)
CM          where NKUVXVX is the number of nodes in the current zone.
CC
CC --------------------
CC        Output
CC --------------------
CC
CO  CRESLT: Status of work done
```

19.5.3. Name Conversions

19.5.3.1. CONVERT_NAME_U2S

Converts user names to solver names, for domains, subdomains, boundary conditions, materials, phases and variables. It is available in both user CEL and junction box routines.

```
      SUBROUTINE CONVERT_NAME_U2S (TYPE,USER_NAME,
     &                             SOLVER_ZONE,SOLVER_NAME,OPER,CRESLT,
     &                             CZ,DZ,IZ,LZ,RZ)
CC --------------------
CC        Input
CC --------------------
CI TYPE          : Data Type
CI USER_NAME     : User name
CI SOLVER_ZONE   : Zone Name (solver format), if CTYPE = 'Variable'
CC
CC --------------------
CC        Output
CC --------------------
CO SOLVER_ZONE   : Zone Name (solver format),
CC                 for all TYPEs except 'Variable'.
CC SOLVER_NAME   : Solver name of form Var_Mat_Phase.
CC OPER          : Operator, if applicable.
CC CRESLT        : ='GOOD' for a successful conversion.
```

The arguments are of type:

```
      CHARACTER*(*) TYPE,USER_NAME,
     & SOLVER_ZONE,SOLVER_NAME,OPER,CRESLT
```

The character string TYPE denotes the type of entity for which a name conversion is required. Permitted types are tabulated below, together with descriptions of input and output names.

TYPE	Description	Example User Name Format	Solver Name Format
'Domain'	domain name = zone name	'Pipe'	'ZN*n*'

TYPE	Description	Example User Name Format	Solver Name Format
'Subdomain'	subdomain name = volume patch name	'Heated Region'	'VP*n*'
'Boundary' or 'Bcp'	boundary condition patch name	'Inlet'	'BCP*n*'
'Material'	material name (library variable) SOLVER_ZONE may be blank	'O2'	'MT*m*'
'Monitor Point'	Monitor Point name	'MonitorPoint1'	'MP*n*'
'Source Point'	Source Point name	SourcePoint1	'SP*n*'
'Radiometer'	Radiometer name SOLVER_ZONE may be blank	Radiometer1	'RADMON*n*'
'Rigid Body'	Rigid Body name SOLVER_ZONE may be blank	RigidBody1	'RB*n*'
'Region'	Region name SOLVER_ZONE may be blank	Region1	'REG*n*'
'Additional Variable'	Additional Variable Name	Water.Phi1	for example 'QPUMASS_AVm_FLn'
'AV'	Additional Variable name (library variable) SOLVER_ZONE may be blank	'Phi'	'AV*m*'
'Phase'	phase name = fluid or solid name	'Water.' 'Copper'	for example, 'FL*n*' for example 'SL*n*'
'Variable'	variable name	'Water.Velocity' 'Water.O2.Mass Fraction' 'Water.Phi1' 'Copper.Temperature'	for example 'VEL_FL*n*' for example 'MASFRN_MT*m*_FL*n*' for example 'QPUMASS_AVm_FL*n*' for example 'TEMP_SL*m*'

Note:

- The SOLVER_ZONE name is output for all entity types, *except* for TYPE = 'Variable', in which case it is required as an input variable.
- OPER = 'GRADIENT' is returned if TYPE = 'Variable', and the USER_NAME is the name of a variable gradient. Otherwise, OPER = ' ' is returned.
- The format of the name returned using 'Additional Variable' (for example, 'QPUMASS_AV2_FL1') is the format accepted by many internal utility routines.

19.5.3.2. CONVERT_NAME_S2U

Converts solver names to user names, for domains, subdomains, boundary conditions, materials, phases and variables. It is available in both user CEL and junction box routines.

```
      SUBROUTINE CONVERT_NAME_S2U (TYPE,SOLVER_NAME,OPER,
     &                             USER_NAME,CRESLT,
     &                             CZ,DZ,IZ,LZ,RZ)
CC TYPE          : Data Type
CI SOLVER_NAME   : Solver Name.
CC OPER          : Operator, if applicable
CO USER_NAME     : User name
CO CRESLT        : ='GOOD' for a successful conversion.
```

The arguments are of type:

```
      CHARACTER*(*) CTYPE,SOLVER_NAME,OPER,USER_NAME,CRESLT
```

This performs the reverse operation. For details, see CONVERT_NAME_U2S (p. 670). See the table for TYPE name conventions, as well as for full descriptions of input and output names.

Examples of its use are provided in:

- User CEL Example 2: Using Gradients for an Additional Variable Source (p. 700)
- Junction Box Example 2: Integrated Residence Time Distribution (p. 714)
- Junction Box Example 4: Solver Control (p. 723).

19.5.3.3. VAR_ALIAS

This function returns the alias (user name) of the input variable CVAR (solver format).

```
      CHARACTER*(*) FUNCTION VAR_ALIAS(CVAR, CZ,DZ,IZ,LZ,RZ)
```

The arguments are of type:

```
      CHARACTER*(*) CVAR
```

Its functionality is superseded by CONVERT_NAME_S2U (p. 672), but it is retained, as it may be quicker and easier to use in certain circumstances. For details, see CONVERT_NAME_S2U (p. 672).

19.5.3.4. LOCALE_ALIAS

Returns the alias (user name) of the input locale LOCALE (solver format).

```
      CHARACTER*(*) FUNCTION LOCALE_ALIAS(LOCALE, CZ,DZ,IZ,LZ,RZ)
```

The arguments are of type:

```
      CHARACTER*(*) LOCALE
```

Its functionality is superseded by CONVERT_NAME_S2U (p. 672), but it is retained, as it may be quicker and easier to use in certain circumstances. For details, see CONVERT_NAME_S2U (p. 672).

19.5.3.5. CONVERT_USER_NAMES

Converts user variable names to solver variable names. The functionality of this routine is superseded, but it is included for back compatibility. For details, see CONVERT_NAME_U2S (p. 670).

```
      SUBROUTINE CONVERT_USER_NAMES(SOLVERR_ZONE,USER_NAME,SOLVER_NAME,OPER,
     &                              CRESLT,CZ,DZ,IZ,LZ,RZ)
CC --------------------
CC        Input
CC --------------------
CI SOLVER_ZONE   : Zone Name (solver format)
CI USER_NAME     : User name
CC --------------------
CC        Output
CC --------------------
CC SOLVER_NAME   : Solver name of form Var_Mat_Phase.
CC OPER          : Operator, if applicable.
CC CRESLT        : ='GOOD' for a successful conversion.
```

The arguments are of type:

```
      CHARACTER*(*) SOLVER_ZONE,USER_NAME,SOLVER_NAME,OPER,CRESLT
```

OPER = ‘GRADIENT’ is returned if the user variable name is a gradient. Otherwise, OPER = ‘ ‘ is returned.

19.5.3.6. GET_PHASE_FROM_VAR

Returns the phase name from the input variable name CVAR.

```
      SUBROUTINE GET_PHASE_FROM_VAR ( CVAR, PHASE )
CI CVAR Variable Name (Solver Format)
CO PHASE Fluid or solid name (Solver Format. Blank if not present).
The arguments are of type:
CHARACTER*(*) CVAR, PHASE
```

On exit, PHASE = ‘FL*n*’ for a fluid phase, or ‘SL*n*’ for a solid phase, where *n* is an integer.

Examples of its use are available. For details, see User CEL Example 2: Using Gradients for an Additional Variable Source (p. 700).

19.5.3.7. PARSMP

This subroutine provides the property, material and fluid components of the input variable CVAR.

```
      SUBROUTINE PARSMP(WHO,CVAR,CPROP,CMAT,CPHASE,LBULK,
     & LNOPHASE,CERACT,CRESLT,CZ,DZ,IZ,LZ,RZ)
CI CVAR         : Target variable name (Solver Format)
CO CPROP        : Material property (i.e. class) part of name
CO CMAT         : Material (i.e. component) part of name
CO CPHASE       : Fluid part of name
CO LBULK        : True if material name is missing from full name
CO LNOPHASE     : True if phase name is missing from full name
```

The arguments are of type:

```
      CHARACTER*(*) WHO,CVAR,CPROP,CMAT,CPHASE,CERACT,CRESLT
      LOGICAL LBULK,LNOPHASE
```

19.5.4. Character Handling

The following section outlines some useful character handling functions and subroutines that operate on Fortran character variables.

Functions are provided for carrying out internal formatting in a standard manner, without using Fortran format statements. Thus, you can convert numbers (double precision, integer or real) to and from their character representations. Using the string editing facilities it is possible to remove redundant spaces from a character string, find the *active length* of a character string, and split a character string into separate strings by treating spaces as separators.

19.5.4.1. CFROMD

```
      CHARACTER*(*) FUNCTION CFROMD (DVAL)
      DOUBLE PRECISION DVAL (input)
```

This function converts a double precision number DVAL into a character string, of the form ' -1.2345678D+09' (in this case NFIG=8). The number of figures contained in CFROMD is determined by NFIG. In the calling subroutine, CFROMD must be declared EXTERNAL and to be CHARACTER*n where n is at least of size NFIG+7.

Note that CFROMD produces a right-justified character string (that is, padded with leading blank spaces).

19.5.4.2. CFROMI

```
      CHARACTER*(*) FUNCTION CFROMI (IVAL)
      INTEGER IVAL (input)
```

This function converts an integer IVAL into a character string, of the form ' -123456'. The number of figures contained in CFROMI is chosen purely according to the value of IVAL. In the calling subroutine, CFROMI must be declared EXTERNAL and to be CHARACTER*n where n is at least of size m+2, where m is the maximum number of figures that will be needed to represent any integer IVAL.

Note that CFROMI produces a right-justified character string (that is, padded with leading blank spaces).

19.5.4.3. CFROMR

```
      CHARACTER*(*) FUNCTION CFROMR (RVAL)
      REAL RVAL (input)
```

This function converts a real number RVAL into a character string, of the form ' -1.2E+05' (in this case NFIG=2). The number of figures contained in CFROMR is determined by NFIG as described above. In the calling subroutine, CFROMR must be declared EXTERNAL and to be CHARACTER*n where n is at least of size NFIG+7.

Note that CFROMR produces a right-justified character string (that is, padded with leading blank spaces).

19.5.4.4. DFROMC

```
      DOUBLE PRECISION FUNCTION DFROMC (CVAL)
      CHARACTER*(*) CVAL (input)
```

This function converts a character string CVAL into a double precision number. If CVAL is not a valid string for producing a double precision number then the result is 0.

19.5.4.5. IFROMC

```
      INTEGER FUNCTION IFROMC (CVAL)
      CHARACTER*(*) CVAL (input)
```

This function converts a character string CVAL into an integer. If CVAL is not a valid string for producing an integer then the result is 0.

19.5.4.6. RFROMC

```
      REAL FUNCTION RFROMC (CVAL)
      CHARACTER*(*) CVAL (input)
```

This function converts a character string CVAL into a real number. If CVAL is not a valid string for producing a real number then the result is 0.

19.5.4.7. LENACT

```
      INTEGER FUNCTION LENACT (CSTR)
      CHARACTER*(*) CSTR (input)
```

This function gives the position of the last character in the string CSTR, which is *not* a space. Thus, it gives the *active length* of the string.

LENACT must be declared EXTERNAL in any calling subroutine.

19.5.4.8. EDIT

```
      SUBROUTINE EDIT (CSTR, IL, IU, COM, IAC)
      CHARACTER*(*) CSTR (input/output)
      INTEGER IL,IU (input)
      CHARACTER*(*) COM (input)
      INTEGER IAC (output)
```

This subroutine edits a substring of CSTR. The action taken depends on the value of the command string, COM. Four possible actions can be taken:

- COM='U'⇒convert to upper case;
- COM='L'⇒convert to lower case;
- COM='S'⇒remove all spaces;
- COM='C'⇒remove leading and trailing spaces and compress all other contiguous spaces to a single space.

The edit is performed on a substring of CSTR determined by the values of IL and IU. The edit starts at character, I1, and ends at character, I2, where these are determined as follows. I1 is the maximum of IL and 1; therefore, if IL is less than or equal to zero the first character in CSTR is the beginning character. I2 is the minimum of the string length of CSTR and IU. Additionally, if IU is less than or equal to zero, I2 is set to the length of CSTR. If I2 is less than I1, then no edit is performed.

IAC is returned with the length of the substring edited if COM is 'L' or 'U.' If COM is 'S' or 'C' it is returned with the active length of the substring edited, see LENACT above.

Note that if COM is not set to a valid character, then EDIT simply returns without taking any action.

Note that MAPCHA must be called before EDIT can be used.

19.5.4.9. IOPNAM

```
      SUBROUTINE IOPNAM (CNAME, LENSTR, NCHAR, NNAME, LFULL, MXNAME)
      CHARACTER*(*) CNAME (input)
      INTEGER LENSTR, MXNAME (input)
      INTEGER NCHAR(MXNAME), NNAME (output)
      LOGICAL LFULL (output)
```

This subroutine locates words between slashes inside a string, and is useful for parsing data area pathnames. It takes the pathname given in CNAME and returns a pointer to the start of each part of the pathname within the string CNAME, in the array NCHAR, starting at the left-hand end of the string.

MXNAME is used as the dimension of NCHAR and should be set to the maximum number of names anticipated in the pathname.

LENSTR is the active length of the string CNAME and should be determined by calling LENACT for CNAME prior to the call to this routine.

NNAME provides a count of the separate names in the pathname.

The logical flag LFULL is set to indicate whether the pathname provided is a full one or a relative one. LFULL=.TRUE. if the first character of CNAME is a '/' and .FALSE. otherwise.

For instance, the pathname (CNAME) '/TIME0/VEL/UVEL' would return:

LFULL=.TRUE. (It is a full pathname.)

NCHAR(1) = 2 (The first name in the path (TIME0) starts at character 2 of CNAME.)

NCHAR(2) = 8 (The second name in the path (VEL) starts at character 8 of CNAME.)

NCHAR(3) = 12 (The third name in the path (UVEL) starts at character 12 of CNAME.)

NNAME=3 (There are 3 names in the pathname.)

19.5.4.10. CCATI

```
      CHARACTER*(*) FUNCTION CCATI(STRING,NUMBER)
      CHARACTER*(*) STRING:- String prefix.
      INTEGER NUMER:- Number to be post-fixed as a string.
```

This function is used to concatenate a string and number. CCATI must be declared external in the routine calling it and must be typed to a length sufficient to hold the concatenated string and number. If CCATI is too short to hold the result, then it will be filled with asterisks. If CCATI is longer than necessary then it is padded to the right with blanks. For example, if STRING='FCS' and NUMBER = 4 then CCATI returns 'FCS4.'

19.5.5. Output Routines

19.5.5.1. MESAGE

```
      SUBROUTINE MESAGE (COMMAND, MESSAGE)
      CHARACTER*(*) COMMAND, MESSAGE (input)
```

This subroutine writes the character string MESSAGE. This character string can be written directly to the output file (that is, to standard output), or it can be written into a buffer, which may later be emptied into the output file.

The character string COMMAND determines exactly what is done with MESSAGE.

Usually, COMMAND will be set to 'WRITE' or 'WRITE-ASIS'. In that case, MESSAGE is written directly to the output file.

COMMAND = 'WRITE' indicates that MESSAGE will be rearranged to remove redundant spaces, and (if possible) to ensure that no word is broken at the end of a record.

COMMAND = 'WRITE-ASIS' means that MESSAGE will be written without rearrangement, except that an end-of-record is inserted every LENREC characters. Each call to MESAGE for 'WRITE' or 'WRITE-ASIS' starts a new record in the output file.

The buffer has a capacity of 4096 characters. Once it becomes full, any further writes to the buffer will be ignored, until it has been emptied or written to the output file.

Setting COMMAND to be 'BUFF' or 'BUFF-ASIS' causes MESSAGE to be written into the buffer (and not the output file).

COMMAND = 'BUFF' asks for MESSAGE to be rearranged to remove redundant spaces.

COMMAND = 'BUFF-ASIS' just writes MESSAGE into the buffer without rearrangement.

Each of these calls to MESAGE does not generally insert an end-of-record; when the buffer is finally written to the output file, only the LENREC character-record length is used to determine where re-

cords end. However, it is possible to forcibly insert an end-of-record into the buffer, by calling MESAGE with **COMMAND = 'BUFF-NL'**. For this call the contents of MESSAGE are ignored.

To write the buffer contents to the output file, and empty the buffer, just call MESAGE with **COMMAND = 'BUFF-OUT'**. Wherever possible, spaces will be inserted to ensure that no word is broken by an end-of-record. MESSAGE will be ignored in this call.

To empty the buffer without writing the contents to the output file, simply set **COMMAND = 'BUFF-DEL'**. Again, MESSAGE is ignored for this value of COMMAND.

Any other value for COMMAND is treated as 'WRITE'.

Note that the character handling functions described can be used to put numbers into a message. For example:

```
      CALL MESAGE( 'WRITE', 'Max Temp is '//CFROMR(TMAX)//' at vertex'//
     & CFROMI(IVX))
```

For details, see Character Handling (p. 674).

19.5.5.2. ERRMSG

This routine is used to print out error messages. Output can be buffered to enable the message to be gathered from various substrings.

```
      SUBROUTINE ERRMSG(MESSAGE)
      CHARACTER*(*) MESSAGE:- Message string to be output directly or buffered.
```

Simple strings will be written directly. If the first three characters of the message are '*B*' then the following calls to ERRMSG will not output any message but will copy the strings into a buffer. The buffer will be output when a string is passed to ERRMSG with the characters '*E*' as the last three characters. The following sets of calls all produce the same message:

```
      CALL ERRMSG('My name is ' // first_name // last_name)
      CALL ERRMSG('*B*My name is ')
      CALL ERRMSG(first_name)
      CALL ERRMSG(last_name //'*E*')
      CALL ERRMSG('*B*')
      CALL ERRMSG('My name is ')
      CALL ERRMSG(first_name)
      CALL ERRMSG(last_name)
      CALL ERRMSG('*E*')
```

When using the buffer facility, it is strongly recommended that the '*B*' and '*E*' strings be passed as separate calls to ERRMSG as in the last example. Note that the first example might be illegal Fortran if either of the string variables first_name or last_name were declared to be of length *(*) and the second would be illegal if last_name was declared *(*).

19.6. CFX Memory Management System (MMS)

The following topics will be discussed:

- Introduction to the Memory Management System (p. 679)
- Error Conventions (p. 680)

- Stack Pointers (p. 681)
- Subroutines (p. 682)

19.6.1. Introduction to the Memory Management System

This introduction to the memory management system, as used in the CFX-Solver, enables you to operate on and pass around your own data structures between user routines.

All arrays (except perhaps for very small local arrays) are stored in global stacks, there being one stack for REAL (RZ), one for DOUBLE PRECISION (DZ), one for INTEGER (IZ), one for CHARACTER (CZ), and one for LOGICAL (LZ). Similarly, any scalar variables that are not local may be stored in a stack. Such sections of the stack (arrays or individual scalars) be referred to as data areas.

The total number of data areas available in the memory management system, also referred to as the **catalogue size**, can be set by the programmer.

Within each stack, data areas can be created and deleted, but each individual data area will be contiguous.

When data areas are deleted, holes appear in the stack, but the stack catalogue is managed so as to ensure that no two holes are adjacent (that is, holes will be merged). A new data area will usually be placed in the smallest hole available, which is large enough to contain it.

Some data areas can be flagged as "vulnerable" so that, when the stack is full, one or more of these will be deleted automatically to make space. However, if the stack does not become full then all vulnerable data areas will remain unchanged.

The naming convention for data areas follows a UNIX-like structure, employing the concept of directories. Thus, a data area can be referred to by its full pathname /TIME-0/ GRID2/ VEL, say, or by its local name VEL, provided the current directory is /TIME-0 /GRID2/. It is also possible to refer to data areas and directories using relative pathnames, GRID2/ VEL, say, if the current directory is /TIME-0/. As in UNIX, the current and parent directories may be referenced by the character strings '.' and '..'. Note that '/' is referred to as the *home directory*, and that 'HOME' is not allowed as a directory or data area name.

There are utilities available for changing the current directory, pushing and popping to directories, creating new directories, creating new data areas, and so on. It is also possible to link directories and data areas.

Data area and directory names are currently limited to, at most, MXDNAM = 20 characters. Full pathnames of directories and data areas are currently limited to at most MXPNAM = 200 characters. MXDNAM and MXPNAM are both declared in the include file `MMS.h`, which should be included in all routines requiring character variables of these lengths to be declared.

Example:

```
#include "MMS.h"
      CHARACTER*(MXDNAM) VARNAME
      CHARACTER*(MXPNAM) PATHNAME
      ....
      ....
      VARNAME = 'MY_VARIABLE'
      PATHNAME = '/USER_DATA/MY_VARIABLE'
```

As some compilers save local variables, the use of character variables of length MXPNAM should be kept to a minimum.

19.6.2. Error Conventions

This section summarizes the conventions adopted for announcing and controlling errors in the memory management system, and fully describes the role of the character variables CERACT and CRESLT, which occur in many of the argument lists for subroutines.

A typical memory management subroutine contains two character strings in its argument list: CERACT and CRESLT. CERACT is passed as input to the subroutine, and determines the action to be taken when there is an error. CRESLT is set as output, and indicates the result of the call to the subroutine. If the subroutine call is successful then CRESLT = 'GOOD' on exit.

Both CERACT and CRESLT should be declared as CHARACTER*4 in the calling subroutine.

Below are the standard values that can be set for CERACT.

```
CERACT = 'SKIP'
```

In the event of an error, there is no warning message and the program continues (control being returned to the calling subroutine).

```
CERACT = 'STOP'
```

In the event of an error, an error message is produced and the program is stopped.

```
CERACT = 'WARN'
```

In the event of an error, a warning message is produced and the program continues (control being returned to the calling subroutine).

If any other value is set for CERACT, then this is equivalent to 'STOP'.

There are many values that can be set for CRESLT by the various memory management subroutines, and these are listed below.

CRESLT	Description
ADRS	The address passed for locating an array entry (for example, JADRES in PEEK and POKE subroutines) is out of range.
BIG	When using POKECA or POKECS to set a character string, or even when using PEEKCA to obtain a character string, this value for CRESLT indicates that the character variable passed down is too big for this data area. Conversely, when using PEEKCS, to obtain a character string, this value for CRESLT indicates that the character variable passed to contain it is too small. This error code can also be returned by the subroutine SQZDAT, to show that you are trying to increase the length of a data area (or give it a negative length). Finally, this error code will be returned by subroutine MMSTAT if LENINF is set too small.

CRESLT	Description
BSL	The character string passed down to PEEKCA or POKECA, to contain an entry from a character array, has a length inconsistent with the total length of the character array data area (BSL = Bad String Length).
DAT	The name passed to the subroutine represents a data area, when it should (perhaps) be a directory.
DIR	The name passed to the subroutine represents a directory, when it should be a data area.
END	There are no more data areas or subdirectories to be found.
ERR	There is some unspecified error, not covered elsewhere in this list.
FCAT	The memory management catalogue is full (catalogue size has been exceeded). To increase the catalogue size, see Starting the CFX-Solver from the Command Line in the *CFX-Solver Manager User's Guide* and Configuring Memory for the CFX-Solver in the *CFX-Solver Manager User's Guide*.
FULL	The memory management stack is full.
GOOD	The subroutine successfully achieves what was asked of it.
ILEG	The name passed to the subroutine is illegal. This value of CRESLT is returned if one of the directories in the pathname does not exist, or if the pathname has an illegal character or is too long.
NLNK	The name passed to the subroutine is an existing data area or directory in its own right, when it should have been created by a call to MAKLNK. In the case of MMSTAT, this indicates an internal error within this subroutine.
NONE	The name passed to the subroutine could not be found.
OLD	The name passed to the subroutine is already in use.
SIZE	There is a problem with the size requested.
TEMP	The subroutine cannot find a temporary name for workspace.
TYPE	The data area is of an inappropriate type (integer, real, and so on.) for this particular POKE or PEEK subroutine. In the case of MMSTAT, this error condition implies that the value set for CTYPE has not been recognized.
OPTS	Illegal options passed to FNDFIL.

19.6.3. Stack Pointers

Data on the stacks is often first accessed by name then manipulated efficiently by using a pointer. By convention, a stack pointer is a variable with a name starting with a small "p". It is declared as of type `_stack_point_`, for example:

```
_stack_point_ pVAR
```

`_stack_point_` is a pre-processor macro that is currently equivalent to `INTEGER` but allows future extensions for long addresses. It can be accessed in any Fortran routine that contains the statement:

```
#include "stack_point.h"
```

before the macro is first used.

Pointers are returned by routines that create new data areas (MAKDAT) or find existing data areas (LOCDAT). In each the caller specifies a name for the data, for example, MY_VAR and receives a pointer in return, for example, pMY_VAR.

It is recommended that stack pointers are used locally within a routine where they are obtained; there is a clear distinction between routines that use pointers and those that do not. These stack points should not be stored for future calls because it is not guaranteed that the referenced item will remain in memory at the same location.

It is common practice to pass a stack location, for example, RZ(pVAR), as an argument down to another subroutine that can manipulate this data as an ordinary Fortran variable or array. For example:

```
CALL MY_SUB( RZ(pMY_ARRAY), LEN_MY_ARRAY)
....
....
SUBROUTINE MY_SUB( A, N )
   REAL A(N)
   DO I = 1, N
      A(I) = 0.0
   END DO
END
```

19.6.4. Subroutines

This section describes the subroutines available to a program developer, by giving a specification for each of the subroutines provided by the memory management system. A consistent approach to error handling is adopted, with character variables CERACT and CRESLT used to control and report errors. For details, see Error Conventions (p. 680).

19.6.4.1. Directories

19.6.4.1.1. MAKDIR

```
SUBROUTINE MAKDIR (CDRNAM, CERACT, CRESLT)
CHARACTER*(*) CDRNAM, CERACT (input)
CHARACTER*4 CRESLT (output)
```

This creates a new directory CDRNAM, where CDRNAM can be a full pathname (in which case it must start with '/') or it can be a relative pathname. Note that the relative pathname can include '.' to indicate the current directory or '..' to indicate the parent directory, just as in UNIX. However, you must be careful when using '..' in the presence of linked directories.

19.6.4.1.2. DELDIR

```
SUBROUTINE DELDIR (CDRNAM, CERACT, CRESLT)
CHARACTER*(*) CDRNAM, CERACT (input)
CHARACTER*4 CRESLT (output)
```

This deletes the directory CDRNAM, along with all subdirectories and data areas that it contains.

19.6.4.1.3. CHGDIR

```
SUBROUTINE CHGDIR (CDRNAM, CERACT, CRESLT)
```

```
      CHARACTER*(*) CDRNAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

This changes the current directory to CDRNAM.

Note that linked directories behave like UNIX *hard links* as far as changing directories is concerned. For example, suppose you have two directories 'A1' and 'A2', each with subdirectories 'B1' and 'B2', and that 'B1' is linked to 'B2':

A1/ B1 → A2/ B2

Suppose that 'A1' is the current directory, and you perform the following sequence of commands:

```
      CALL CHGDIR ('B1', 'STOP', CRESLT)
      CALL CHGDIR ('..', 'STOP', CRESLT)
```

Then you finish up back in 'A1', unlike UNIX *soft links*, which would finish up in 'A2'.

19.6.4.1.4. CHMDIR

```
      SUBROUTINE CHMDIR (CDRNAM, CERACT, CRESLT)
      CHARACTER*(*) CDRNAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

This changes the current directory to CDRNAM, if CDRNAM exists; otherwise it first makes the directory CDRNAM and then changes into it.

19.6.4.1.5. PSHDIR

```
      SUBROUTINE PSHDIR (CDRNAM, CERACT, CRESLT)
      CHARACTER*(*) CDRNAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

This places the name of the current directory on a stack of previous current directories, and then changes the current directory to CDRNAM. It is designed to be used in combination with POPDIR (see below).

19.6.4.1.6. PSHDRH

```
      SUBROUTINE PSHDRH (CDRNAM, CERACT, CRESLT)
      CHARACTER*(*) CDRNAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

This places the name of the current directory on a stack of previous current directories, and then changes the current directory to CDRNAM. It is designed to be used in combination with POPDIR (see below). This routine behaves the same as PSHDIR if a directory is not a link to another directory. However, if the input directory is a link, this routine follows the link so that '..' is actually the parent directory of CDRNAM. With PSHDIR, '..' follows the original linked directory.

19.6.4.1.7. POPDIR

```
      SUBROUTINE POPDIR (CERACT, CRESLT)
```

```
      CHARACTER*(*) CERACT (input)
      CHARACTER*4 CRESLT (output)
```

This changes back to the directory stored on the top of the stack of previous current directories, and removes this directory from the stack.

If this directory no longer exists, the current directory remains unchanged, but the top entry on the stack of previous current directories will still be removed. The value of CERACT then determines whether the program continues. Note that the stack is not cleared and any subsequent call to POPDIR attempts to change to the directory given by the new top entry on the stack.

POPDIR is designed to be used in combination with PSHDIR (see above). Note that the stack of current working directories is a last-in-first-out stack, so care must be taken if nesting PSHDIR's and POPDIR's.

19.6.4.1.8. LISDAT

```
      SUBROUTINE LISDAT
```

All subdirectories and data areas are listed (in the standard output file) for the current directory. The name of this current directory is also printed.

19.6.4.1.9. PUTDIR

```
      SUBROUTINE PUTDIR (CURDIR, LENCD)
      CHARACTER*200 CURDIR (output)
      INTEGER LENCD (output)
```

The full pathname of the current directory is written into CURDIR (which should be at least of length 200 bytes). The actual number of active bytes in the full pathname is returned as LENCD.

19.6.4.1.10. FNDFIL

```
      SUBROUTINE FNDFIL(CDRNAM, CNAME, COPTS, CFNAME, LFNAME, CLNAME, \
      LLNAME, CDTYPE, CERACT, ISIZE, JADRES, CRESLT)
      CHARACTER*(*) CDRNAM (input)
      CHARACTER*(*) CNAME (input)
      CHARACTER*(*) COPTS (input)
      CHARACTER*(*) CERACT (input)
      CHARACTER*208 CFNAME (output)
      CHARACTER*4 CDTYPE, CRESLT (output)
      CHARACTER*208 CLNAME (output)
      INTEGER ISIZE, JADRES, LFNAME, LLNAME (output)
```

This routine replaces the old TARLIS routine, and its call is done in very much the same way. Much more control over what is returned by this routine is provided than for TARLIS however, and the DEFAULT action is to search only in the directory CDRNAM and *not recursively* through subdirectories of CDRNAM.

CDRNAM is the directory to be searched.

CNAME is the filename to be searched for. The name may contain up to two wild cards for which the '*' character is used as in Linux. The following are valid examples of CNAME:

CNAME = 'FRED'

Returns only files called FRED

CNAME = '*DAT'

Returns only files ending in 'DAT'

CNAME = 'FCP*'

Returns only files beginning in 'FCP'

CNAME = '*STR*'

Returns only files including the string 'STR'

CNAME = '*'

Returns all filenames.

Note that CNAME should be set only on the first call. FNDFIL is always initialized for a new search whenever CNAME is non-blank.

COPTS is a string holding the file system type options. All options should be prepended with a '-' character, the minus sign. More than one option can be included, for example, COPTS='-opt1 -opt2 -opt3 .....'. Valid options are:

-R

Search recursively through all subdirectories

-ALL

Return all file types

-DIR

Return directory files

-LINK

Return link files

-'DATA'[>lev | <lev]

Return files of data type 'DATA'.

The optional part in [ ] after the data type option is used to specify a vulnerability level restriction on the data areas. For example:

```
-INTR > All integer data areas;
```

-INTR >10 > Integer data areas with a vulnerability >10;

```
-REAL <50 > Real data areas with a vulnerability <50.
```

Thus, to specify all vulnerable integer data areas use, '-INTR>0'. For all non-vulnerable logical files, use '-LOGL<1'.

Options are processed in order and so can be modified in the expected manner; thus, '-ALL-INTR<10' => all files but only integer files with a vulnerability less than 10.

Note if the options string is left blank then this is the same as '-ALL'.

CFNAME is returned with the relative path name of the file from the directory being listed, in the form './d1/d2/.../filename'.

If CRESLT = 'END' then there are no more data areas to be found, and so CFNAME will not contain the name of a new directory or data area.

Note that CRESLT = 'END' is treated like CRESLT = 'GOOD' as far as error handling is concerned, so it is not an error condition.

Once a call has produced the result CRESLT = 'END' or CRESLT = 'ERR' the next call to FNDFIL is treated as a first call and a new search begins.

LFNAME contains the length of CFNAME in characters, and CDTYPE contains the data type of CFNAME; this will be one of 'REAL', 'INTR', 'DBLE', 'LOGL' or 'CHAR'.

ISIZE is the size of the data area CFNAME, and JADRES is the start address for this data area in the relevant stack.

If ISIZE = 0 and LLNAME = 0 then CFNAME is actually a directory. But if ISIZE=0 and LLNAME is greater than zero, then CFNAME is a link to another data area or directory, in which case CLNAME and LLNAME are returned containing the name of the link, and the length of this name, respectively. When ISIZE=0, CDTYPE is not defined.

The following illustrates the calling sequence using FNFIL:

First call:

```
CALL FNDFIL('Dir. name', '*', '-R -ALL', ...)
```

Subsequent calls:

```
CALL FNDFIL(' ', ' ',' ', ...)
```

For the subsequent calls only the CNAME argument is checked and it must be blank otherwise a new search will be initiated.

```
      LOGICAL FUNCTION LDIR(CDIR)
      CHARACTER*(*) CDIR (input)
```

This function is true if the directory CDIR exists.

```
      LOGICAL FUNCTION LFIL(CFIL)
      CHARACTER*(*) CFIL (input)This function is true if the file CFIL exists
```

19.6.4.2. Data Areas

19.6.4.2.1. MAKDAT

```
      SUBROUTINE MAKDAT (CDANAM, CDTYPE, CERACT, ISIZE, JADRES, CRESLT)
      CHARACTER*(*) CDANAM, CERACT (input)
      CHARACTER*4 CDTYPE (input)
      INTEGER ISIZE (input)
      INTEGER JADRES (output)
      CHARACTER*4 CRESLT (output)
```

CDANAM can be set to either a program name or a data area name. If CDANAM is a program name, then CERACT is also set to 'CODE', and MAKDAT simply initializes the character string CCODE to the program name, and then returns. CCODE is then saved for future calls to MAKDAT. The program name can then be used in user friendly memory error messages.

If CDANAM is a data area name, then MAKDAT creates a new data area called CDANAM, which is of type CDTYPE (CDTYPE= 'REAL','INTR','DBLE','LOGL' or 'CHAR'). CERACT performs its usual function, and ISIZE defines the size of the new data area (for character arrays, ISIZE is the total number of bytes required).

The start address in the appropriate global stack is returned as JADRES.

Note that MAKDAT may delete vulnerable data areas if there is no room available in the stack for a new data area (see VULDAT below).

If MAKDAT fails due to lack of space on one of the stacks, that is, CRESLT = 'FULL', the following message is sent to standard output, providing CERACT is not set to 'SKIP':

```
*** INSUFFICIENT MEMORY ALLOCATED ***
```

Action required:

Depending on what program is running, there then follows a message relating to what memory stack is full. If the program is Ansys CFD-Flo, then the following message is shown:

```
Re-run CFX-Pre and increase the CDTYPE stack memory factor. Use the
Workspace Sizes menu under Advanced Control of Ansys CFD-Flo Paramet
ers.
```

Where CDTYPE is one of 'CHAR', 'DBLE', 'INTR', 'LOGL', or 'REAL', to indicate for which data type MAKDAT has failed. Otherwise, for other programs, the following message is shown:

```
Contact the CFDS Customer Helpline giving the following details
```

If MAKDAT fails due to the file catalogue being full, that is, CRESLT = 'FCAT', the following message is sent to standard output, providing CERACT is not set to 'SKIP':

```
>> Insufficient memory allocated. Try increasing file catalogue size.
```

To increase the catalogue size, see Starting the CFX-Solver from the Command Line in the *CFX-Solver Manager User's Guide* and Configuring Memory for the CFX-Solver in the *CFX-Solver Manager User's Guide*.

19.6.4.2.2. MAKVEC

```
SUBROUTINE MAKVEC (CDANAM, CDTYPE, CERACT, LENVEC,ISIZE, JADRES, CRESLT)
CHARACTER*(*) CDANAM, CERACT (input)
CHARACTER*4 CDTYPE (input)
INTEGER LENVEC (input)
INTEGER ISIZE (input)
INTEGER JADRES (output)
CHARACTER*4 CRESLT (output)
```

This routine is identical to MAKDAT in every respect except that instead of creating a data area of length ISIZE stack words it creates a data area of ISIZE*LENVEC stack words. The vector word length, LENVEC stack words, is stored in the MMS. This is particularly useful for character string arrays; LENVEC is then the string length in characters of the character array. Note this is important when using PEEKCA and POKECA.

19.6.4.2.3. GRBSTK

```
SUBROUTINE GRBSTK (CDANAM, CDTYPE, CERACT, ISIZE, JADRES, CRESLT)
CHARACTER*(*) CDANAM, CERACT (input)
CHARACTER*4 CDTYPE (input)
INTEGER ISIZE (output)
INTEGER JADRES (output)
CHARACTER*4 CRESLT (output)
```

This routine is identical to MAKDAT except that it returns a pointer to the largest free data area of the required type that is available. ISIZE is returned with the data area size in words. If no free areas exist, then the largest vulnerable data area is used.

19.6.4.2.4. SQZDAT

```
SUBROUTINE SQZDAT (CDANAM, CERACT, ISIZE, CRESLT)
CHARACTER*(*) CDANAM, CERACT (input)
INTEGER ISIZE (input)
CHARACTER*4 CRESLT (output)
```

This shrinks the data area CDANAM, reducing its length to ISIZE, which must be no larger than the original length of the data area.

19.6.4.2.5. RESIZE

```
SUBROUTINE RESIZE (CDANAM, CERACT, ISIZE, CRESLT, CZ, DZ, IZ, LZ, RZ)
CHARACTER*(*) CDANAM, CERACT (input)
INTEGER ISIZE (input)
CHARACTER*4 CRESLT (output)
CHARACTER CZ(*) (input/output)
DOUBLE PRECISION DZ(*) (input/output)
INTEGER IZ(*) (input/output)
LOGICAL LZ(*) (input/output)
REAL RZ(*) (input/output)
```

This resizes the data area CDANAM, to the new size ISIZE, which may be larger or smaller than the original length of the data area.

NOTE that the address of CDANAM within the stack may be changed as a result of calling RESIZE.

19.6.4.2.6. DELDAT

```
      SUBROUTINE DELDAT (CDANAM, CERACT, CRESLT)
      CHARACTER*(*) CDANAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

This deletes the data area CDANAM.

19.6.4.2.7. LOCDAT

```
      SUBROUTINE LOCDAT (CDANAM, CDTYPE, CERACT, ISIZE, JADRES, CRESLT)
      CHARACTER*(*) CDANAM, CERACT (input)
      INTEGER ISIZE, JADRES (output)
      CHARACTER*4 CDTYPE, CRESLT (output)
```

This subroutine locates an existing data area CDANAM, and provides its type CDTYPE, size ISIZE, and address JADRES.

Note that if CDANAM is a vulnerable data area, then LOCDAT makes it safe again until the next call to VULDAT for CDANAM.

19.6.4.2.8. INFDAT

```
      SUBROUTINE INFDAT (CDANAM, CDTYPE, CERACT, LENVEC, ISIZE, JADRES, LEVVUL,CRESLT)
      CHARACTER*(*) CDANAM, CERACT (input)
      INTEGER LENVEC,ISIZE, JADRES,LEVVUL (output)
      CHARACTER*4 CDTYPE, CRESLT (output)
```

This subroutine locates an existing data area CDANAM, and provides its type CDTYPE, vector word length, size ISIZE, in units of the vector word length, address JADRES and vulnerability level LEVVUL. If a character string array had been set up using MAKVEC with 20 strings of 10 character string length, then LENVEC would be returned with 10 and ISIZE with 20 a total length of 200 characters.

Note that unlike LOCDAT, this routine does NOT affect the vulnerability of a data area this is a pure inquiry routine.

19.6.4.3. Renaming And Linking

19.6.4.3.1. RENAM

```
      SUBROUTINE RENAM (COLDNM, CNEWNM, CERACT, CRESLT)
      CHARACTER*(*) COLDNM, CNEWNM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

The data area or directory COLDNM is given the new name CNEWNM. The subroutine will fail if CNEWNM already exists.

N.B. This routine was originally called RENAME. It has been renamed RENAM to avoid a clash with the UNIX system routine rename on some platforms.

19.6.4.3.2. MAKLNK

```
      SUBROUTINE MAKLNK (COLDNM, CNEWNM, CERACT, CRESLT)
      CHARACTER*(*) COLDNM, CNEWNM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

The data area or directory COLDNM is linked to a new data area or directory CNEWNM, that is, it is given the additional name CNEWNM. If CNEWNM is subsequently deleted (using DELLNK), then COLDNM still survives under its original name. However, if instead COLDNM is subsequently deleted (using DELDAT or DELDIR) then CNEWNM is also deleted automatically, so be careful with links! It is important to be aware of what may happen to the target of a linked data area.

Note that if COLDNM is actually a link, rather than a data area or directory, then MAKLNK operates as if COLDNM were replaced by the name of the data area or directory to which it is linked. Thus, there is never a link to another link; there are only links to data areas or directories.

Note that, as far as changing directories is concerned, linked directories behave as UNIX *hard links*, as explained in CHGDIR (p. 682).

19.6.4.3.3. DELLNK

```
      SUBROUTINE DELLNK (CLNNAM, CERACT, CRESLT)
      CHARACTER*(*) CLNNAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
```

The data area or directory CLNNAM is assumed to have been created by a call to MAKLNK (with CNEWNM=CLNNAM), and this call to DELLNK will delete that link.

19.6.4.4. Copying

19.6.4.4.1. COPDAT

```
      SUBROUTINE COPDAT (CDFROM, CDTO, CERACT, CRESLT, CZ, DZ, IZ, LZ, RZ)
      CHARACTER*(*) CDFROM, CDTO, CERACT (input)
      CHARACTER CZ(*) (input/output)
      DOUBLE PRECISION DZ(*) (input/output)
      INTEGER IZ(*) (input/output)
      LOGICAL LZ(*) (input/output)
      REAL RZ(*) (input/output)
      CHARACTER*4 CRESLT (output)
```

This subroutine copies one data area to another. CDFROM is the name of the original data area, and this must already exist. A new data area called CDTO is created, of the same size and type as CDFROM, and all data from CDFROM is copied to CDTO. If CDFROM is linked to another data area, then this is taken as the source for the data. The subroutine fails if CDTO already exists.

It is necessary to pass the global stacks CZ, DZ, IZ, LZ, RZ to this subroutine.

19.6.4.4.2. COPDIR

```
      SUBROUTINE COPDIR (CDFROM, CDTO, CERACT, CRESLT, CZ, DZ, IZ, LZ, RZ)
      CHARACTER*(*) CDFROM, CDTO, CERACT (input)
      CHARACTER CZ(*) (input/output)
```

```
      DOUBLE PRECISION DZ(*) (input/output)
      INTEGER IZ(*) (input/output)
      LOGICAL LZ(*) (input/output)
      REAL RZ(*) (input/output)
      CHARACTER*4 CRESLT (output)
```

This subroutine copies a whole directory (CDFROM) and all its subdirectories to another, new, directory CDTO. The names of all subdirectories and data areas are recreated in the new directory. COPDIR will work if CDFROM is linked to another directory, but the subroutine will fail if CDTO already exists.

It is important to note that any data areas or subdirectories of CDFROM, which are links, will not be copied.

19.6.4.5. Setting and Reading Individual Values

19.6.4.5.1. POKECA

```
SUBROUTINE POKECA (CDANAM, JADRES, CVALUE, CERACT, CRESLT, CZ)
```

19.6.4.5.2. POKECS

```
SUBROUTINE POKECS (CDANAM, CVALUE, CERACT, CRESLT, CZ)
```

19.6.4.5.3. POKED

```
SUBROUTINE POKED (CDANAM, JADRES, DVALUE, CERACT, CRESLT,DZ)
```

19.6.4.5.4. POKEI

```
SUBROUTINE POKEI (CDANAM, JADRES, IVALUE, CERACT, CRESLT,IZ)
```

19.6.4.5.5. POKEL

```
SUBROUTINE POKEL (CDANAM, JADRES, LVALUE, CERACT, CRESLT,LZ)
```

19.6.4.5.6. POKER

```
      SUBROUTINE POKER (CDANAM, JADRES, RVALUE, CERACT, CRESLT, RZ)
      CHARACTER*(*) CDANAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
      CHARACTER*(*) CVALUE (input)
      DOUBLE PRECISION DVALUE (input)
      INTEGER JADRES, IVALUE (input)
      LOGICAL LVALUE (input)
      REAL RVALUE (input)
      CHARACTER CZ(*) (output)
      DOUBLE PRECISION DZ(*) (output)
      INTEGER IZ(*) (output)
      LOGICAL LZ(*) (output)
      REAL RZ(*) (output)
```

These POKE subroutines are designed for setting a value at a single address, for a scalar or for one entry of an array, which is stored in the global stacks. This saves the programmer the job of locating the data area, and assigning a value. There is a separate subroutine for each Fortran variable type.

CDANAM is the name of the data area and JADRES is the position within the array (assuming CDANAM is an array) that is to be given a value. Note that if the data area CDANAM is a scalar, rather than an array, then JADRES is not used by the POKE subroutine.

CVALUE (or DVALUE and so on) is the actual value that is to be assigned.

The relevant stack is passed (CZ, DZ, and so on) as an argument, along with the usual error handling flags CERACT and CRESLT.

In the case of character strings there are two subroutines, POKECA for arrays and POKECS for scalar (single string) data areas. When using POKECA, it is important to have the correct length for the string CVALUE, as one element of the character array. As with the other POKE subroutines, JADRES is simply the location within the character array where the value CVALUE will be set. For example, for an array CHARACTER*2 CEXAMP(3) (total length 6 bytes), if you want to set the last entry CEXAMP(3) to be 'AB,' then you must set CDANAM to be CEXAMP, JADRES to be 3 and CVALUE to be 'AB.' If CVALUE is actually a local variable in the calling subroutine, then it must be declared as CHARACTER*2.

For string arrays created with MAKVEC, the string length is stored in the MMS and POKECA uses that information. In this case, the restrictions on CVALUE are reduced, it can be any length less than equal to the length of a string in the string array. If CVALUE is shorter than the string length of the array, then padding with blanks is used. If CVALUE has a length greater than the array string length, then POKECA fails. Note that INFDAT can be used to determine the array size and string length.

Clearly there is no need for the argument JADRES in the POKECS subroutine, because CDANAM will be a single character string. If CDANAM has length greater than CVALUE, then the string is completed with blanks, while if CVALUE has length greater than CDANAM, then the POKECS fails. Note that calling POKECS for a character array CDANAM will have the effect of filling most of the array with blanks, so caution should be exercised.

19.6.4.5.7. PEEKCA

```
SUBROUTINE PEEKCA (CDANAM, JADRES, CVALUE, CERACT, CRESLT, CZ)
```

19.6.4.5.8. PEEKCS

```
SUBROUTINE PEEKCS (CDANAM, CVALUE, CERACT, CRESLT, CZ)
```

19.6.4.5.9. PEEKD

```
SUBROUTINE PEEKD (CDANAM, JADRES, DVALUE, CERACT, CRESLT, DZ)
```

19.6.4.5.10. PEEKI

```
SUBROUTINE PEEKI (CDANAM, JADRES, IVALUE, CERACT, CRESLT, IZ)
```

19.6.4.5.11. PEEKL

```
SUBROUTINE PEEKL (CDANAM, JADRES, LVALUE, CERACT, CRESLT, LZ)
```

19.6.4.5.12. PEEKR

```
      SUBROUTINE PEEKR (CDANAM, JADRES, RVALUE, CERACT, CRESLT, RZ)
      CHARACTER*(*) CDANAM, CERACT (input)
      CHARACTER*4 CRESLT (output)
      CHARACTER*(*) CVALUE (output)
      DOUBLE PRECISION DVALUE (output)
      INTEGER IVALUE (output)
      INTEGER JADRES (input)
      LOGICAL LVALUE (output)
      REAL RVALUE (output)
      CHARACTER CZ(*) (input)
      DOUBLE PRECISION DZ(*) (input)
      INTEGER IZ(*) (input)
      LOGICAL LZ(*) (input)
      REAL RZ(*) (input)
```

Corresponding to the POKE subroutines, there is this set of PEEK subroutines. These are designed for reading individual values from array data areas, or scalar data areas (that is, of length 1) from the global stacks. Again, there is a separate subroutine for each Fortran variable type, with two subroutines for characters. CDANAM is the name of the data area, with JADRES the array entry, hence, the actual value CVALUE (or DVALUE and so on) is to be obtained.

A programmer using PEEKCA must know the size of each string in the character array CDANAM, and must declare CVALUE to be of this length in the calling subroutine.

For string arrays created with MAKVEC, the string length is stored in the MMS and PEEKCA uses that information. In this case, the restrictions on CVALUE are reduced, it can be any length greater than equal to the length of a string in the string array. If CVALUE is longer than the string length of the array, then padding with blanks is used. If CVALUE has a length less than the array string length, then PEEKCA fails. Note that INFDAT can be used to determine the array size and string length.

When using PEEKCS the programmer may not necessarily know the length of CDANAM. In this case, CVALUE should be declared sufficiently large and LOCDAT must be used to obtain the actual length required for CVALUE. After the call to PEEKCS, CVALUE must be truncated to the correct length.

19.6.4.6. Name lists

19.6.4.6.1. NAMLST

```
      SUBROUTINE NAMLST (COM, CLIST, IENTRY, CVALUE, NCVAL, CERACT, RESLT, CZ, IZ)
      CHARACTER COM*(*) (input)
```

```
      CHARACTER CLIST*(*) (input)
      INTEGER IENTRY (input/output)
      CHARACTER CVALUE*(*) (input/output)
      INTEGER NCVAL (output)
      CHARACTER CERACT*(*) (input)
      CHARACTER CRESLT*(4) (output)
      CHARACTER CZ(*)*(1) (input/output)
      INTEGER IZ(*) (input/output)
```

A name list is simply a long character string consisting of a set of substrings. Each substring corresponds to a name and may be of arbitrary length. NAMLST enables you to treat the name list as if it were an array of strings, each string being of arbitrary length. The main advantage of using NAMLST is the potential for saving memory if the names in the list are of widely varying length. Note, however, that no space will be saved unless the mean length of a name is greater than 8 characters.

COM is a command string that takes the following possible values: COMPRESS, CREATE, DELETE, PEEK, POKE & SIZE. This string controls the operation of the routine in a, hopefully, self evident way.

CLIST is the name of the character data area to be used to hold the name list. Note that CLIST MUST be created using the CREATE command of NAMLST.

IENTRY is the array address to be used for COMPRESSing, PEEKing or POKEing an entry in the name list. In addition, IENTRY is used to set the maximum number of entries in the name list on creation and is returned with this number if the SIZE command is used.

CVALUE is used to hold string values for PEEKing and POKEing. For a PEEK CVALUE must be long enough to hold the entire entry. For a POKE, trailing spaces in CVALUE are ignored.

NCVAL is used to return the active length of CVALUE, which is the length after trailing spaces have been removed, when PEEKing.

If COM equals COMPRESS, then the IENTRY'th value in the list is removed and then the entries above have their entry number reduced by one; that is, entry IENTRY+n becomes IENTRY+n-1.

If COM equals CREATE, then a new name list, CLIST, is created with IENTRY possible entries.

If COM equals DELETE, then the name list, CLIST, is deleted.

If COM equals PEEK, then the IENTRY'th name in the list is returned in CVALUE(1:NCVAL) provided CVALUE is at least NCVAL characters long, where NCVAL is the active length in characters of the IENTRY'th name. If the IENTRY'th name has not yet been set, then CRESLT will be returned with NONE.

If COM equals POKE, then the IENTRY'th name in the list is overwritten with CVALUE(1:NACTIV), where NACTIV is the active length of CVALUE. Note that if the length of the new entry differs from the original, then CLIST has to be restructured incurring a CPU penalty. If an entire name list is to be filled, then the first entry should be POKEd in first, the second second, and so on, for maximum efficiency.

If COM equals SIZE then IENTRY is returned with the maximum possible number of entries for the name list.

Note that NAMLST uses the vertical bar character, "|", as a separator in the CLIST array and so no entry should contain this character.

19.6.4.7. Memory Management Statistics

19.6.4.7.1. GETMMS

```
      SUBROUTINE GETMMS (CTYPE, CERACT, LENINF, INFO, CRESLT)
      CHARACTER*(*) CTYPE, CERACT (input)
      CHARACTER*4 CRESLT (output)
      INTEGER LENINF (input)
      INTEGER INFO (output)
```

This subroutine is used to obtain memory management statistics. You should set LENINF to be at least 25, and provide a local array INFO in the calling subroutine, which has dimension at least LENINF.

The character CTYPE should be set as one of 'CLCK,' 'FILE,' 'SYS,' 'REAL,' 'INTR,' 'DBLE,' 'LOGL' or 'CHAR' and this determines the information that will be returned in the INFO array.

If CTYPE = 'CLCK' then information is returned on the total number of memory management system calls so far:

Number of calls = $J+10^{8}\times I$ *

where I=INFO(1) and J=INFO(2).

If CTYPE = 'FILE' then information is returned on the number of memory management system files (data areas, directories, and so on):

INFO(1) is the number of data areas present now.

INFO(2) is the peak number of data areas used so far.

INFO(3) = I and INFO(4) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(5) is the number of directories present now.

INFO(6) is the peak number of directories used so far.

INFO(7) = I and INFO(8) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(9) is the number of link files present now.

INFO(10) is the peak number of link files used so far.

INFO(11) = I and INFO(12) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(13) is the total number of files (data areas + directories + link files) present now.

INFO(14) is the peak number of files used so far.

INFO(15) = I and INFO(16) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

If CTYPE = 'SYS' then information is returned on the system limit statistics:

INFO(1) is the maximum number of files permitted at any one time.

INFO(2) is the maximum number of free data areas permitted at any one time.

INFO(3) is the maximum permitted length of a filename in characters.

INFO(4) is the maximum number of directories allowed in a full pathname.

INFO(5) is the maximum vulnerability level allowed.

If CTYPE='CHAR,' 'DBLE,' 'INTR,' 'LOGL' or ' REAL' then statistics are returned just for data areas of that particular data type (namely character, double precision, integer, logical or real respectively):

INFO(1) is the number of data areas present now.

INFO(2) is the peak number of data areas used so far.

INFO(3) = I and INFO(4) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(5) is the address space used by data.

INFO(6) is the peak address space used so far.

INFO(7) = I and INFO(8) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(9) is the number of vulnerable data areas present now.

INFO(10) is the peak number of vulnerable data areas set so far.

INFO(11) = I and INFO(12) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(13) is the address space used by vulnerable data.

INFO(14) is the peak vulnerable address space set so far.

INFO(15) = I and INFO(16) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(17) is the number of free areas present now.

INFO(18) is the peak number of free areas created so far.

INFO(19) = I and INFO(20) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(21) is the free address space available now.

INFO(22) is the minimum amount of free address space available so far.

INFO(23) = I and INFO(24) = J, giving the number of memory management calls made when this peak usage occurred (using the * formula above).

INFO(25) is the number of vulnerable data areas deleted by the memory management system (to make space for new data) so far.

For all values of CTYPE, if MMSTAT is in an error condition no statistics are available, but if LENINF is at least 3, then the following will be returned:

INFO(1) = NUMERR, the error condition flag in MMSTAT.

INFO(2) = I and INFO(3) = J, giving the number of memory management calls made so far (using the * formula above).

19.6.4.7.2. WSTAT

```
SUBROUTINE WSTAT
```

This subroutine prints long memory management system statistics to standard output. These consist of all data available from subroutine GETMMS, arranged in tabular form where appropriate.

19.7. User Fortran in Ansys Workbench

Simulations using User Fortran can be run within Ansys Workbench. However, the use of User Fortran for Remote Solve Manager runs is unsupported (although it may work if the file paths for external files on the remote machine are the same as on the machine that sets up the case). For details, see Using CFX with the Remote Solve Manager in the *CFX Introduction*.

If you set user parameters in the USER command language object (see User Parameters (p. 642)) that include file or directory paths, then these can be automatically updated by Workbench when files are copied for Remote Solve Manager runs or for archiving purposes. This requires you add an additional user parameter "Workbench Path Parameters" which is set to a comma-separated list of all parameter names that should be automatically updated by Workbench.

For example, if you have a CCL file containing user parameters such as the following:

```
USER:
  Input File 1 = /home/cfxuser/u_profile.dat
  Input File 2 = /home/cfxuser/v_profile.dat
  InletTemperature = 300
  Workbench Path Parameters = Input File 1, Input File 2
END
```

then the paths given in "Input File 1" and "Input File 2" will automatically be updated and the referenced files copied and made available for RSM runs and archiving.

19.8. User CEL Examples

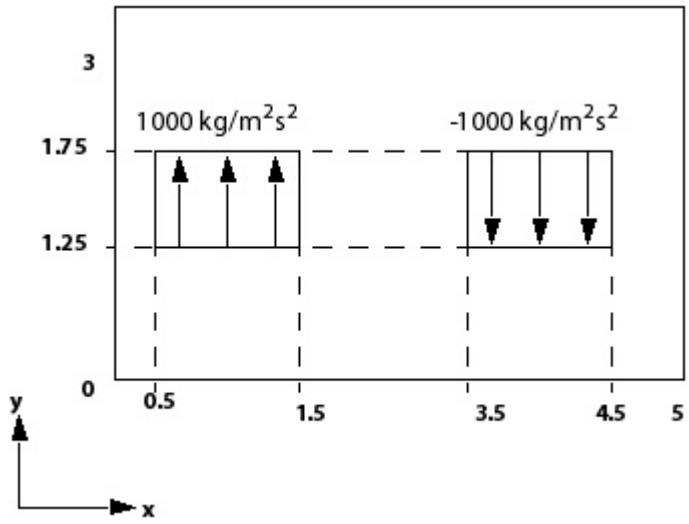
The following topics will be discussed:

- User CEL Example 1: User Defined Momentum Source (p. 698)
- User CEL Example 2: Using Gradients for an Additional Variable Source (p. 700)
- User CEL Example 3: Integrated Quantity Boundary Conditions (p. 704)

19.8.1. User CEL Example 1: User Defined Momentum Source

19.8.1.1. Problem Setup

A common application of user CEL functions is the specification of user defined source terms. In the following example, a constant source term for the y-component of the momentum equation has to be applied on two rectangular boxes characterized by their extension in the x and y coordinate direction.



19.8.1.2. Creating the User CEL Function

Additional information on creating user CEL functions in CFX-Pre is available in User Functions in the *CFX-Pre User's Guide*.

First, you should first create a user routine with the following settings:

- Routine Name: `UserSourceRoutine`
- Option: User CEL Function
- Calling Name: `user_source`
- Library Name: `MomentumSource1`
- Library Path: `/home/cfxuser/shared_libraries`

Next, you should create a User Function with the following settings:

- Function Name: UserSource

- Option: User Function
- User Routine Name:
- Argument List: `[m], [m]`
- Result Units: `[kg m^-2 s^-2]`

In this example, the compiled code for the user subroutine `MomentumSource1.F` is stored in the shared library `libMomentumSource1.so` (the prefix and suffix may vary depending on your platform), which can be found under the `/home/cfxuser/shared_libraries/<architecture>` directory. If there is a problem linking the shared library to the CFX-Solver, you can check that it has been created, but you will usually not need to know about this library.

The new user CEL function can now be used to set the momentum source components within the subdomain as follows:

- Momentum x-comp: `0.0`
- Momentum y-comp: `UserSource(X,Y)`
- Momentum z-comp: `0.0`

These values are set on the Subdomain Sources form. For details, see Sources Tab in the *CFX-Pre User's Guide*.

19.8.1.3. User Fortran Routine

Source terms for the momentum equations can be specified in CEL for a given subdomain. Because the user CEL routine defined the extent of the source, the source subdomain can be defined to cover the entire flow domain. The subroutine was developed from the template routine `ucf_template.F` available in `<CFXROOT>/examples/`. Note that some commented sections of the routine have not been included here. The routine `MomentumSource1.F` has the following form:

```
#include "cfx5ext.h"
 dllexport(user_source)
      SUBROUTINE USER_SOURCE (
     & NLOC,NRET,NARG,RET,ARGS,CRESLT,CZ,DZ,IZ,LZ,RZ)
C
C .....
C
C ------------------------------
C        Argument list
C ------------------------------
C
      INTEGER NLOC, NRET, NARG
      CHARACTER CRESLT*(*)
      REAL    RET(1:NLOC,1:NRET), ARGS(1:NLOC,1:NARG)
C
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C .....
C
C ------------------------------
C    Executable statements
```

```
C ------------------------------
C
C-------------------------------------------------------
C     SOURCE = RET(1:NLOC,1)
C     X      = ARGS(1:NLOC,1)
C     Y      = ARGS(1:NLOC,2)
C-------------------------------------------------------
C
C---- Low level user routine
      CALL USER_SOURCE_SUB (NLOC,RET(1,1),ARGS(1,1),ARGS(1,2))
C
      CRESLT = 'GOOD'
      END
      SUBROUTINE USER_SOURCE_SUB (NLOC,SOURCE,X,Y)
C
C .....
C
C ------------------------------
C        Local Variables
C ------------------------------
      INTEGER NLOC, ILOC
      REAL    SOURCE(NLOC), X(NLOC), Y(NLOC)
C-------------------------------------------------------
C     -  0.5<x<1.5 and 1.25<y<1.75 --> SOURCE =  1000.0
C     -  3.5<x<4.5 and 1.25<y<1.75 --> SOURCE = -1000.0
C-------------------------------------------------------
C --------------------------
C    Executable Statements
C --------------------------
      DO ILOC=1,NLOC
         SOURCE(ILOC) = 0.0
         IF (X(ILOC).GE.0.5  .AND. X(ILOC).LE.1.5  .AND.
     &    Y(ILOC).GE.1.25 .AND. Y(ILOC).LE.1.75) THEN
            SOURCE(ILOC) =  1000.0
         ELSE IF (X(ILOC).GE.3.5  .AND. X(ILOC).LE.4.5  .AND.
     &     Y(ILOC).GE.1.25 .AND. Y(ILOC).LE.1.75) THEN
            SOURCE(ILOC) = -1000.0
         END IF
      END DO
C
      END
```

19.8.2. User CEL Example 2: Using Gradients for an Additional Variable Source

For some applications, the source terms for the transport equations might depend on local gradients. Gradients can be accessed directly within CEL. However, gradients of most variables can also be accessed in CEL expressions through the use of user CEL functions. This is achieved by calling the utility USER_GETVAR with the 'Gradient' operator attached to the variable name.

The following example shows the use of a source term that depends on the gradients of one Additional Variable, φ_1, in the transport equation of another Additional Variable, φ_2.

$$S_{\varphi 2} = a\left[\left(\frac{\partial \varphi_1}{\partial x}\right)^2 + \left(\frac{\partial \varphi_1}{\partial y}\right)^2 + \left(\frac{\partial \varphi_1}{\partial z}\right)^2\right] \tag{19.1}$$

$$\nabla \cdot \left(\rho \underline{U} \varphi_2\right) = S_{\varphi 2} \tag{19.2}$$

For this demonstration density is constant, the flow is uniform and φ_1 is a simple algebraic variable:

$$\varphi_1 = x + 2y + 3z \tag{19.3}$$

so that the solution along a streamline can be trivially verified as:

$$\varphi_2(l) = \varphi_2(0) + 14\frac{a\,l}{\rho\,s} \tag{19.4}$$

where l is the distance from the inlet and s is the flow speed.

A user CEL function is given the coefficient a as an argument and computes the whole of the source term, $S_{\varphi 2}$. The variables φ_1 and φ_2 have dimensions of length, so their gradients are therefore dimensionless. The coefficient a has the same dimensions as the source term, which are those of density times velocity.

19.8.2.1. Problem Setup

19.8.2.1.1. Creating the Additional Variables

Before creating a domain, define two Additional Variables:

- Create an Additional Variable called `phi1` of **Type Unspecified** with **Units** of `[m]`
- Create an Additional Variable called `phi2` of **Type Specific** with **Units** of `[m]`

19.8.2.1.2. Creating the Domain

Create a domain that includes both Additional Variables phi1 and phi2:

- Declare `phi1` of **Type Algebraic,** and type in the expression $x+2\,y+3\,z$ to define it
- Declare `phi2` of **Type Transport Equation**. Do not set a Kinematic Diffusivity

Create a domain that includes both Additional Variables, solved using a transport equation. Do not set a kinematic diffusivity.

19.8.2.1.3. Creating the User CEL Routine and Function

In CFX-Pre, you should create a User Routine and then a User Function. For details, see User Routine Details View in the *CFX-Pre User's Guide*. Additional information on creating a User CEL Function in CFX-Pre is available; for details, see User Functions in the *CFX-Pre User's Guide*.

The User Routine takes the following form:

- Routine Name: `UserSource2Routine`
- Option: `User CEL Function`
- Calling Name: `user_source2`
- Library Name: `AdVarSource`
- Library Path: `/home/cfxuser/shared_libraries`

and the User Function is set up as follows:

- Function Name: `UserSource2`
- Option: `User Function`

- User Routine Name: `UserSource2Routine`
- Argument List: `[kg m^-2 s^-1]`
- Result Units: `[kg m^-2 s^-1]`

In this example, the user subroutine `AdVarSource.F` is stored in the shared library `libAdVarSource.so` (the prefix and suffix may vary depending on your platform), which can be found under the `/home/cfxuser/shared_libraries/<architecture>` directory.

19.8.2.1.4. Defining the Source Term

The new User CEL Function can now be used to set the Additional Variable source within the subdomain as follows:

- Create a subdomain and set the Additional Variable Source Value as: `UserSource2(1000 [kg m^-2 s^-1])`.

This is accessed from the Subdomain Sources form. For details, see Sources Tab in the *CFX-Pre User's Guide*.

The coefficient *a* in the nonlinear source term has been set to a constant value of 1000 [kg m^-2 s^-1].

19.8.2.2. User Fortran Routine

The subroutine was developed from the template routine `ucf_template.F` available in `<CFXROOT>/examples/`. Note that some commented sections of the routine have not been included here. This routine contains a call to USER_GETVAR to access variable data. For details, see Utility Routines for User Functions (p. 647). Note that USER_GETVAR requires the fluid prefix for user-supplied variable names. For a one-off application it is possible simply to call USER_GETVAR with an assumed name for the working fluid, for example, for a single phase problem using "Water":

```
      CALL USER_GETVAR (‘Water,phi1.Gradient’, CRESLT, pGRAD_PHI,
     &                  CZ,DZ,IZ,LZ,RZ)
```

However, to make it applicable in general, the example given here uses USER_ASSEMBLE_INFO to extract the equation and principal names, and GET_PHASE_FROM_VAR followed by CONVERT_NAME_S2U to extract the user's phase name. Hence, it works on any problem, independently of the choice of fluids.

- USER_ASSEMBLE_INFO (p. 668)
- GET_PHASE_FROM_VAR (p. 673)
- CONVERT_NAME_S2U (p. 672)

The routine `AdVarSource.F` has the following form:

```
#include "cfx5ext.h"
dllexport(user_source2)
      SUBROUTINE USER_SOURCE2 (
     & NLOC,NRET,NARG,RET,ARGS,CRESLT,CZ,DZ,IZ,LZ,RZ)
C--------------------
C       Details
C --------------------
```

```
C  ARGS(1:NLOC,1) holds parameter 'a' evaluated at all locations
C  RET(1:NLOC,1)  will hold return result
C ------------------------------
C     Preprocessor includes
C ------------------------------
#include "MMS.h"
#include "stack_point.h"
C ------------------------------
C        Argument list
C ------------------------------
      INTEGER NLOC,NARG,NRET
      CHARACTER CRESLT*(*)
      REAL ARGS(NLOC,NARG), RET(NLOC,NRET)
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C ------------------------------
C        External routines
C ------------------------------
      INTEGER LENACT
      EXTERNAL LENACT
C ------------------------------
C        Local Variables
C ------------------------------
      CHARACTER*(MXDNAM) ACTION,CGROUP,CEQN,CTERM,CPVAR,
     & CLVAR,CPATCH,CRESLOC,CPHASE
      CHARACTER*120 User_Phase_Name, User_Variable_Name
C ------------------------------
C        Stack pointers
C ------------------------------
      __stack_point__ pGRAD_PHI
C --------------------------
C    Executable Statements
C --------------------------
C  Initialise success flag.
      CRESLT = 'GOOD'
C  Initialise RET to zero.
      CALL SET_A_0 ( RET, NLOC*NRET )
C
C---- Determine user's phase name for use in USER_GETVAR
C
C  Use USER_ASSEMBLE_INFO to determine solver equation and principal
C  variable names CEQN, CPVAR.
      ACTION = 'GET'
      CALL USER_ASSEMBLE_INFO (ACTION,CGROUP,CEQN,CTERM,CPVAR,
     &  CLVAR,CPATCH,CRESLOC,
     & CZ,DZ,IZ,LZ,RZ)
      IF (CRESLOC.NE.'GOOD' .AND. CRESLOC.NE.'SOME') THEN
         CRESLT = 'FAIL'
         GO TO 999
      ENDIF
C  Extract phase name from principal variable
      CALL GET_PHASE_FROM_VAR (CPVAR, CPHASE)
C  Convert solver phase name to user phase name.
      CALL CONVERT_NAME_S2U('Phase',CPHASE,' ',User_Phase_Name,
     & CRESLT, CZ,DZ,IZ,LZ,RZ)
      IF (CRESLT .NE. 'GOOD') GO TO 999
C
C---- Obtain grad(phi1)
C     in array shape GRAD_PHI(1:3,1:NLOC) located at RZ(pGRAD_PHI)
C
      User_Variable_Name = User_Phase_Name(1:LENACT(User_Phase_Name))
     & // '.phi1.Gradient'
      CALL USER_GETVAR (User_Variable_Name, CRESLT, pGRAD_PHI,
     & CZ,DZ,IZ,LZ,RZ)
      IF (CRESLT .NE. 'GOOD') GO TO 999
C
C---- Calculate source expression in RET(1:NLOC,1)
C
```

```
      CALL USER_SOURCE_CAL( RET(1,1), ARGS(1,1), RZ(pGRAD_PHI), NLOC )
C
  999 CONTINUE
C
C  Send any diagnostics or stop requests via leader processor
      IF (CRESLT .NE. 'GOOD') THEN
         CALL MESAGE( 'BUFF', 'USER_SOURCE2 returned error:' )
         CALL MESAGE( 'BUFF', CRESLT )
         CALL MESAGE( 'BUFF-OUT', ' ' )
      END IF
C
C=======================================================================
      END
      SUBROUTINE USER_SOURCE_CAL (SOURCE, A, GRAD_PHI, NLOC)
C
C  Purpose: Source = a * grad(phi).grad(phi)
C
C  Inputs
      INTEGER NLOC
      REAL A(NLOC), GRAD_PHI(3,NLOC)
C  Outputs
      REAL SOURCE(NLOC)
C  Locals
      INTEGER ILOC
C
      DO ILOC = 1, NLOC
          SOURCE(ILOC) = A(ILOC) * ( GRAD_PHI(1,ILOC)**2
     &     + GRAD_PHI(2,ILOC)**2
     &     + GRAD_PHI(3,ILOC)**2 )
      END DO
C
      END
```

19.8.3. User CEL Example 3: Integrated Quantity Boundary Conditions

19.8.3.1. Problem Setup

One application of the integrated quantity functions would be to set a boundary inlet temperature based on some average outflow values from a domain. In this way, you could set up a boundary condition, which acts like a thermostat control for a room. This requires the use of a User CEL Function to set the inflow temperature, and one of the arguments, which is passed to the subroutine, is the average outflow temperature.

19.8.3.1.1. Creating the User Function

Further information on creating a User CEL Function in CFX-Pre is available in User Functions in the *CFX-Pre User's Guide*.

First, you should first create a User Routine with the following settings:

- Routine Name: `INLET T`
- Option: `User CEL Function`
- Calling Name: `inlet_t`
- Library Name: `InletTemperature`
- Library Path: `/home/cfxuser/shared_libraries`

Next, you should create a User Function with the following settings.

- Function Name: `INLET T`
- User Routine Name: `INLET T`
- Argument List: `[K], [Pa]`
- Result Units: `[K]`

In this example, the user subroutine `InletTemperature.F` is stored in the shared library `libInletTemperature.so` (the prefix and suffix may vary depending on your platform), which can be found under the `/home/cfxuser/shared_libraries/<architecture>` directory. The new User CEL Function can now be used to set the feedback loop for the inlet temperature as follows:

- On the **Inlet Boundary Condition Values** form set the **Heat Transfer** option to `Static Temperature` and enter the expression:

  ```
  INLET_T(areaAve(T)@Outflow,areaAve(p)@Outflow)
  ```

Note that the integrated quantity is passed into the inlet temperature function as an argument. The CFX-Solver recalculates these values during the coefficient loop so that the value is always up to date. For details, see Boundary Details: Inlet in the *CFX-Pre User's Guide*.

19.8.3.2. User Fortran Routine

The routine `InletTemperature.F` has the following form (note that this is not a complete routine, the purpose of this example is to demonstrate the quantities that can be passed to the subroutine).

```
#include "cfx5ext.h"
dllexport(inlet_t)
      SUBROUTINE INLET_T (NLOC,NRET,NARG,RET,ARGS,CRESLT,CZ,DZ,IZ,LZ,RZ)
C
C ------------------------------
C        Argument list
C ------------------------------
      INTEGER NLOC, NRET, NARG
      CHARACTER CRESLT*(*)
      REAL    RET(1:NLOC,1:NRET), ARGS(1:NLOC,1:NARG)
C
C---------------------------------------------------------------------
C ‘Static Temperature‘ is stored in RET(1:NLOC,1)
C ‘areaAve(T@Outflow)‘ is stored in ARGS(1:NLOC,1)
C ‘areaAve(p@Outflow)‘ is stored in ARGS(1:NLOC,2)
C---------------------------------------------------------------------
C
C ------------------------------
C    Executable statements
C ------------------------------
C
      ...
      END
```

19.9. User Junction Box Examples

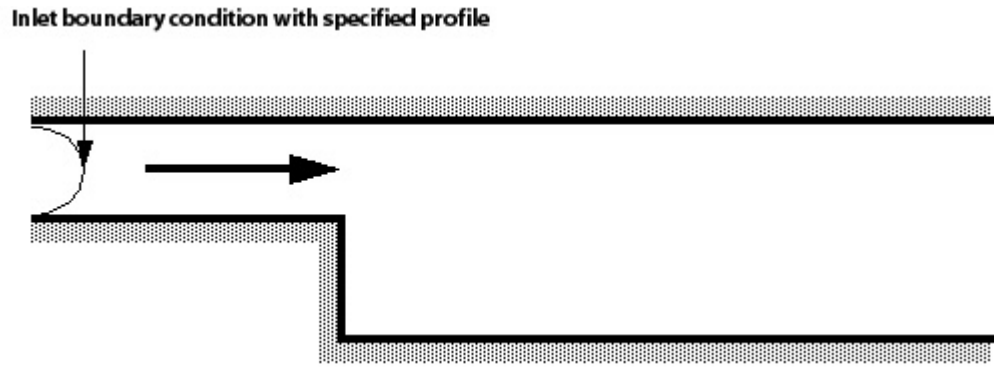
The following topics will be discussed:

- Junction Box Example 1: Profile Boundary Conditions (p. 706)
- Junction Box Example 2: Integrated Residence Time Distribution (p. 714)
- Junction Box Example 3: Timestep Control (p. 718)
- Junction Box Example 4: Solver Control (p. 723)
- Junction Box Example 5: Transient Information (p. 725)

19.9.1. Junction Box Example 1: Profile Boundary Conditions

19.9.1.1. Problem Setup

A common application of User Subroutines is the specification of a profile boundary condition. In this example, experimental data has been obtained for inflow boundary values at the inlet to a domain. The simulation is that of flow over a backward facing step; the inlet profile is given some step heights upstream of the step.



The experimental data is provided in a two-dimensional table. The first column contains the wall-normal positions and the remaining five columns contain values for the u, v and w velocity components, the turbulent kinetic energy and the turbulent eddy dissipation. The table has 29 wall-normal positions and appears as follows:

```
6   29
1.01724E-01 1.01665E-04 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
1.05172E-01 2.94293E-04 0.00000E+00 0.00000E+00 9.43200E+00 1.33962E+04
.......... .......... .......... .......... .......... ..........
1.94828E-01 2.94293E-04 0.00000E+00 0.00000E+00 8.94000E+00 8046000E+02
1.98276E-01 1.01665E-04 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
```

A Junction Box routine is required to read the data from the file and automatically pass it to the follower processors. The profile boundary is then set by a User CEL Function.

19.9.1.1.1. Creating the Junction Box Routine and User CEL Function

In CFX-Pre, you should create a Junction Box Routine, User CEL Routine and a User CEL Function. Further information on creating these objects in CFX-Pre is available in User Functions in the *CFX-Pre User's Guide*.

User CEL Routine:

- Routine Name: `U profile Routine`
- Option: `User CEL Function`
- Calling Name: `user_uprofile`
- Library Name: `backstep`
- Library Path: `/home/user/Shared_Libraries`

User CEL Function:

- Function Name: `U profile`
- Option: `User Function`
- User Routine Name: `U Profile Routine`
- Argument List: `[m]`
- Result Units: `[m/s]`

Junction Box Routine:

- Routine Name: `Load Inlet Profiles`
- Option: Junction `Box Routine`
- Calling Name: `user_input`
- Library Name: `backstep`
- Library Path: `/home/user/Shared_Libraries`
- Junction Box Location: `User input`

As can be seen from this example, both routines are stored in the shared library `libback-step.so` (the prefix and suffix may vary depending on your platform), which can be found in the `/home/user/Shared_Libraries/<architecture>` directory.

19.9.1.1.2. Setting the Boundary Condition

The new User CEL Function can now be used within a boundary condition in CFX-Pre. For example, create an inlet boundary condition using the **Cartesian Velocity Components** option and then set:

- `U = U profile(y)`

- V = 0.00 [m s^-1]
- W = 0.00 [m s^-1]

19.9.1.1.3. Enabling the Junction Box Routine

In order to ensure that the solver calls the declared Junction Box routine, you should:

- Go to the **Solver Control** panel in CFX-Pre. For details, see Basic Settings Tab in the *CFX-Pre User's Guide*.
- Select the Junction Box Routines box, and select the Junction Box Load Inlet Profile. For details, see Basic Settings Tab in the *CFX-Pre User's Guide*.

The following CCL should be added to the CCL File. You can also enter this into the **Command Editor** dialog box (**Tools > Command Editor**) in CFX-Pre and click **Process**.

```
USER:
 # Location of external inlet profiles.
 Input File = /home/user/u_profile.dat
 # Printing control. First character should be Y or N
 User Printing = Yes
END
```

The CCL parameter USER/Input File automatically becomes the MMS parameter USER/INPUT_FILE. This is used in the Fortran subroutines to read in the user data. The User Printing parameter becomes the MMS parameter USER/USER_PRINTING. This is used in the following subroutines to control printing.

19.9.1.2. User Fortran Junction Box Routines

The data table is stored in an ASCII file in /home/user/backstep.user.dat. The user code to read the data from this file is implemented in a user routine shown below. The routine was developed from the template routine jcb_template.F available in *<CFXROOT>*/examples/.

```
#include "cfx5ext.h"
C======================================================================
C
C Purpose: User Fortran for importing/interpolating external
C          velocity profile.
C
C Junction Box Routines:
C          USER_INPUT -> USER_INPUT_TABLE
C
C User CEL Functions:
C          USER_UPROFILE -> USER_UPROFILE_CAL -> USER_TABINT
C                                             -> USER_PRINT_PROFILE
C
C Command File data areas used:
C          /USER/INPUT_FILE
C          /USER/USER_PRINTING
C
C User Input data areas used:
C          /USER_DATA/NCOL
C          /USER_DATA/NLIN
C          /USER_DATA/TABLE(NCOL,NLIN)
C
C======================================================================
dllexport(user_input)
      SUBROUTINE USER_INPUT ( CZ,DZ,IZ,LZ,RZ )
CC
```

```
CC -------------------
CC        Input
CC -------------------
CC  none
CC
CC -------------------
CC      Modified
CC -------------------
CC
CC  Stacks possibly.
CC
CC -------------------
CC        Output
CC -------------------
CC  none
CC
CC -------------------
CC       Details
CC -------------------
CC
CC  Load a velocity profile from filename given in /USER/INPUT_FILE.
CC
CC  This routine is defined with 'Junction Box Location = User Input'.
CC  It reads data into the directory /USER_DATA.
CC
CC  In a parallel run, this routine is only called from the leader
CC  process, but any /USER_DATA data will then be automatically
CC  copied to the /USER_DATA directory on all follower processes.
CC=======================================================================
C ------------------------------
C     Preprocessor includes
C ------------------------------
#include "MMS.h"
#include "stack_point.h"
C
C ------------------------------
C        Argument list
C ------------------------------
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C ------------------------------
C        Local Parameters
C ------------------------------
C Fortran input channel
      INTEGER NCHANNEL
      PARAMETER (NCHANNEL=59)
C
C ------------------------------
C        Local Variables
C ------------------------------
C Result flag
      CHARACTER*4 CRESLT
C
C System info
      CHARACTER*(MXPNAM) WHICH_CALL
C
C Command file info
      CHARACTER*(MXPNAM) INPUT_FILE
      CHARACTER*(MXPNAM) USER_PRINTING
C
      INTEGER NCOL, NLIN
C
C ------------------------------
C        Stack pointers
C ------------------------------
C
      __stack_point__ pOINT, pTABLE
```

```
C
C=======================================================================
C --------------------------
C    Executable Statements
C --------------------------
C
C Look up the system info for junction box calls: WHICH_CALL
      WHICH_CALL = 'Unknown'
      CALL PEEKCS( '/USER/WHICH_CALL', WHICH_CALL, 'SKIP', CRESLT, CZ )
C
C Look up command file info: INPUT_FILE, USER_PRINTING.
      INPUT_FILE = 'Unknown'
      CALL PEEKCS( '/USER/INPUT_FILE', INPUT_FILE, 'STOP', CRESLT, CZ )
C
      USER_PRINTING = 'No'
      CALL PEEKCS( '/USER/USER_PRINTING', USER_PRINTING, 'SKIP',
     &            CRESLT, CZ )
C
C Send any diagnostic messages via leader process.
      IF ( USER_PRINTING(1:1) .NE. 'N' ) THEN
         CALL MESAGE( 'WRITE', 'Start USER_INPUT')
         CALL MESAGE( 'WRITE', 'WHICH_CALL='//WHICH_CALL )
         CALL MESAGE( 'WRITE', 'INPUT_FILE='//INPUT_FILE )
      END IF
C
C Ensure that directory /USER_DATA exists and make this the current directory.
C
      CALL PSHDIR( '/', 'STOP', CRESLT )
      CALL CHGDIR( 'USER_DATA', 'STOP', CRESLT )
C
C Read data from file.
C
      OPEN( NCHANNEL, FILE=INPUT_FILE )
C
C Create space for array sizes NCOL and NLIN
      CALL MAKDAT( 'NCOL', 'INTR', 'STOP', 1, pOINT, CRESLT )
      CALL MAKDAT( 'NLIN', 'INTR', 'STOP', 1, pOINT, CRESLT )
C
C Read data in NCOL and NLIN
      READ ( NCHANNEL, * ) NCOL, NLIN
      CALL POKEI( 'NCOL', 1, NCOL, 'STOP', CRESLT, IZ )
      CALL POKEI( 'NLIN', 1, NLIN, 'STOP', CRESLT, IZ )
C
C Create space for array TABLE(NCOL,NLIN)
      CALL MAKDAT( 'TABLE', 'REAL', 'STOP', NCOL*NLIN, pTABLE, CRESLT )
C
C Read data into TABLE(NLIN,NCOL) located in RZ at pointer pTABLE
      CALL USER_INPUT_TABLE( NCHANNEL, RZ(pTABLE), NCOL, NLIN )
C
      CLOSE(NCHANNEL)
C
C Return to current directory on entry.
C
      CALL POPDIR( 'STOP', CRESLT )
C
C Send any diagnostic messages via leader process.
      IF ( USER_PRINTING(1:1) .NE. 'N' ) THEN
         CALL MESAGE( 'WRITE','End USER_INPUT' )
      END IF
C
      END
C
C=======================================================================
      SUBROUTINE USER_INPUT_TABLE( NCHANNEL, TABLE, NCOL, NLIN )
C     ---------------------------------------------------------
C Read data into TABLE(NLIN,NCOL)
C
C Input arguments
      INTEGER NCHANNEL, NCOL, NLIN
C
C Output arguments
```

```
      REAL TABLE(NCOL,NLIN)
C
      READ( NCHANNEL, *, END=900, ERR=900 ) TABLE
      RETURN
C
C Error Handling via leader process.
 900  CONTINUE
      CALL MESAGE( 'WRITE', 'Unexpected error/end on user file' )
      CALL CFXSTP( 'USER_INPUT_TABLE' )
      END
```

After the subroutine USER_INPUT has been executed, the directory USER_DATA exists and contains the following information:

- /USER_DATA/NLIN: number of lines in the table.
- /USER_DATA/NCOL: number of columns in the table.
- /USER_DATA/TABLE: table containing the inflow boundary profiles.

The data specified in the profile table is used to specify the inlet values for the velocity components and the turbulence quantities. Therefore, each of these quantities requires a User CEL Routine to be created. The following routine is for the first velocity component, which refers to the second column of the inflow profile table:

```
C=======================================================================
dllexport(user_uprofile)
      SUBROUTINE USER_UPROFILE (
     & NLOC, NRET, NARG, RET, ARGS, CRESLT, CZ,DZ,IZ,LZ,RZ )
CC
CD User routine: interpolate U profile from imported data.
CC
CC --------------------
CC        Input
CC --------------------
CC  NLOC   - size of current locale
CC  NRET   - number of components in result
CC  NARG   - number of arguments in call
CC  ARGS() - (NLOC,NARG) argument values
CC
CC --------------------
CC      Modified
CC --------------------
CC  Stacks possibly.
CC
CC --------------------
CC        Output
CC --------------------
CC  RET()  - (NLOC,NRET) return values
CC  CRESLT - 'GOOD' for success
CC
CC --------------------
CC       Details
CC --------------------
CC  An array of profiles TABLE(NCOL,NLIN) has been read from file
CC  by junction box routine USER_INPUT. This routine now uses
CC  the table to return U values for each y value given in the
CC  first argument.
CC
CC=======================================================================
C ----------------------------
C     Preprocessor includes
C ----------------------------
#include "MMS.h"
#include "stack_point.h"
C
```

```
C ------------------------------
C        Argument list
C ------------------------------
      INTEGER NLOC,NARG,NRET
C
      CHARACTER CRESLT*(*)
C
      REAL ARGS(NLOC,NARG), RET(NLOC,NRET)
C
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C ------------------------------
C        Local Variables
C ------------------------------
C Table info
      INTEGER NCOL, NLIN
C
C Command file info
      CHARACTER*(MXPNAM) USER_PRINTING
C
C Arguments for LOCDAT call
      CHARACTER*4 CDTYPE
      INTEGER ISIZE
C
C ------------------------------
C        Stack pointers
C ------------------------------
      __stack_point__ pTABLE
C
C=======================================================================
C --------------------------
C    Executable Statements
C --------------------------
C Look up command file info: USER_PRINTING.
      USER_PRINTING = 'No'
      CALL PEEKCS( '/USER/USER_PRINTING', USER_PRINTING, 'SKIP',
     & CRESLT, CZ )
C
C Send any diagnostic messages via leader process.
      IF ( USER_PRINTING(1:1) .NE. 'N' ) THEN
         CALL MESAGE( 'WRITE','Start USER_UPROFILE' )
      END IF
C
C Initialise RET() to zero.
      CALL SET_A_0( RET, NLOC*NRET )
C
C Find TABLE(NCOL,NLIN)
      CALL PEEKI( '/USER_DATA/NCOL', 1, NCOL, 'STOP', CRESLT, IZ )
      CALL PEEKI( '/USER_DATA/NLIN', 1, NLIN, 'STOP', CRESLT, IZ )
      CALL LOCDAT( '/USER_DATA/TABLE', CDTYPE, 'STOP', ISIZE,
     &            pTABLE, CRESLT )
C
C Compute profile U(N) for Y(N) using TABLE(NCOL,NLIN)
      CALL USER_UPROFILE_CAL(
     & RET, ARGS, NLOC, RZ(pTABLE), NCOL, NLIN, USER_PRINTING )
C
C Send any diagnostic messages via leader process.
      IF ( USER_PRINTING(1:1) .NE. 'N' ) THEN
         CALL MESAGE( 'WRITE','End USER_UPROFILE' )
      END IF
C
C Set success flag.
      CRESLT = 'GOOD'
C
      END
C=======================================================================
      SUBROUTINE USER_UPROFILE_CAL(
```

```
     & U, Y, N, TABLE, NCOL, NLIN, USER_PRINTING )
C     ------------------------------------------
C Compute profile U(N) for Y(N) using TABLE(NCOL,NLIN)
C----------------------
C  Input arguments
C----------------------
C number of points
      INTEGER N
C
C y coordinate
      REAL Y(N)
C
C input table
C Y values are in the first column of the table.
C U values are in the second column of the table.
      INTEGER NCOL, NLIN
      REAL TABLE(NCOL,NLIN)
C
C printing control
      CHARACTER*(*) USER_PRINTING
C-------------------------
C  Output arguments
C-------------------------
      REAL U(N)
C---------------------
C  Locals
C---------------------
      INTEGER I
C
      DO I = 1, N
C
C interpolate U(I) for Y(I)
      CALL USER_TABINT( Y(I), U(I), 1, 2, TABLE, NCOL, NLIN )
C
C print I, U(I), Y(I)
C note that this data is unstructured, not ordered by Y value.
C
      IF ( USER_PRINTING(1:1) .NE. 'N' ) THEN
         CALL USER_PRINT_PROFILE( I, Y(I), U(I) )
      END IF
C
      END DO
C
      END
C=======================================================================
      SUBROUTINE USER_TABINT( X, Y, IX, IY, TABLE, NCOL, NLIN )
C     ------------------------------------------------------
C Linear interpolation of table data Y in column IY of TABLE(),
C against an increasing co-ordinate X in column IX of TABLE().
C--------------
C  Inputs:
C--------------
      REAL X
      INTEGER IX, IY
      INTEGER NCOL, NLIN
      REAL TABLE(NCOL,NLIN)
C--------------
C Outputs:
C--------------
      REAL Y
C--------------
C Locals:
C--------------
      INTEGER I, J
      REAL S
C
C Find J such that TABLE(IX,J) <= X < TABLE(IX,J+1)
      J = 0
      DO I = 1, NLIN
         IF ( TABLE(IX,I) .GT. X ) GO TO 100
         J = I
```

```
      END DO
 100  CONTINUE
C
C Linearly interpolate Y
      IF( J .LE. 0 ) THEN
         Y = TABLE(IY,1)
      ELSE IF ( J .GE. NLIN ) THEN
         Y = TABLE(IY,NLIN)
      ELSE
         S = (X-TABLE(IX,J))/(TABLE(IX,J+1)-TABLE(IX,J))
         Y = TABLE(IY,J)*(1.0-S) + TABLE(IY,J+1)*S
      END IF
      END
C
C=======================================================================
      SUBROUTINE USER_PRINT_PROFILE( I, Y, U )
C     ---------------------------------------
C Print (I, Y, U) via leader process.
C
      INTEGER I
      REAL Y, U
C
      CHARACTER CFROMI*5, CFROMR*15
      EXTERNAL CFROMI, CFROMR
C
      CALL MESAGE( 'BUFF-ASIS', CFROMI(I) )
      CALL MESAGE( 'BUFF-ASIS', ' Y(I) = ' )
      CALL MESAGE( 'BUFF-ASIS', CFROMR(Y) )
      CALL MESAGE( 'BUFF-ASIS', ' U(I) = ' )
      CALL MESAGE( 'BUFF-ASIS', CFROMR(U) )
      CALL MESAGE( 'BUFF-OUT', ' ' )
      END
```

19.9.2. Junction Box Example 2: Integrated Residence Time Distribution

This example makes use of the facility in order to print out integrated values of a user-defined Additional Variable that measures the residence time distribution in the flow domain. It uses the facilities to loop over all domains and all boundary condition patches, and it prints out the integrated residence time distribution on each of these locales. For details, see:

- USER_GET_GVAR (p. 651)
- USER_GET_MESH_INFO (p. 659)
- CONVERT_NAME_S2U (p. 672).

It also uses the facilities to look up the fluid name on each domain. Hence, it should work generically for any single phase problem. For details, see:

- USER_GET_PHYS_INFO (p. 664)
- CONVERT_NAME_S2U (p. 672).

19.9.2.1. Problem Setup

19.9.2.1.1. Creating the Additional Variables

Before creating the domains, define the Additional Variable representing `Residence Time Distribution`.

- Create an Additional Variable called RTD of **Type Volumetric.**
- Set **Units** of time `[s]`.

19.9.2.1.2. Creating the Domains

Create one or more single phase domains. On each domain, include the Additional Variable RTD.

- Declare RTD of **Type Transport Equation**.
- Set its kinematic diffusivity equal to zero.

19.9.2.1.3. Creating the Subdomains and Additional Variable Sources

- On each domain, create a subdomain that encompass the entire domain.
- On each such subdomain, for the Additional Variable RTD, set **Option** to `Source`, and set the dimensionless source term identically equal to unity.

This ensure that the Additional Variable obeys the transport equation:

$$\frac{D}{D t}\left(R T D\right)=1 \tag{19.5}$$

Hence, integrated along stream lines RTD, measure the residence time of fluid particles along streamlines.

19.9.2.1.4. Creating the Junction Box Routine

In CFX-Pre, you should create a User Routine with the following settings:

- Routine Name: `User Output`
- Option: Junction `Box Routine`
- Calling Name: `user_output`
- Library Name: `user output`
- Library Path: `/home/user/Shared_Libraries`
- Junction Box Location: `End of Run`

19.9.2.1.5. Enabling the Junction Box Routine

In order to ensure that the solver calls the declared Junction Box routine, you should:

- Go to the **Solver Control** panel in CFX-Pre. For details, see Basic Settings Tab in the *CFX-Pre User's Guide*.
- Select the **Junction Box Routines** box, and select the Junction Box **User Output**.

19.9.2.2. User Fortran Junction Box Routine

This routine looks up the number of domains, and loops over them. On each domain, it computes and prints out the arithmetic average and maximum values of the `Residence Time Distribution`. Additionally, on each domain, it looks up the number of boundary condition patches, and loops over them. On each BCP, it also computes and prints the arithmetic average `Residence Time Distribution`. Some of the comment lines are omitted. This routine `user_output.F` is supplied in the *<CFXROOT>*`/examples/UserFortran/` directory of your Ansys CFX installation.

```
#include "cfx5ext.h"
dllexport(user_output)
      SUBROUTINE USER_OUTPUT (CZ, DZ, IZ, LZ, RZ)
C
C Routine called at end of run
C
C Preprocessor includes
#include "MMS.h"
#include "cfd_constants.h"
C
C Arguments
      CHARACTER*(1) CZ(*)
      DOUBLE PRECISION DZ(*)
      INTEGER IZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C External routines
      CHARACTER*15 CFROMR
      INTEGER LENACT
      EXTERNAL CFROMR, LENACT
C
C Local parameters
      CHARACTER*(*) ROUTIN
      PARAMETER (ROUTIN = 'USER_OUTPUT')
C Local variables
      CHARACTER*4 CRESLT
      CHARACTER*(MXDNAM) CDIR_GLOB, CDIR_ZONE, CDIR_PHYS,
     &  CZONE, CBCP, CPHASE
      CHARACTER*120 User_Domain_Name, User_BCP_Name,
     &  User_Phase_Name, User_Var_Name
      INTEGER NZN,IZN, NBCP,IBCP
      REAL VAR
C
C Executable statements
      CALL MESAGE( 'WRITE', ' ')
      CALL MESAGE( 'WRITE', '-----------------')
      CALL MESAGE( 'WRITE', 'Start USER_OUTPUT')
      CALL MESAGE( 'WRITE', '-----------------')
      CALL MESAGE( 'WRITE', ' ')
C
C---- Obtain global meshdata
      CDIR_GLOB = ' '
      CALL USER_GET_MESH_INFO (ROUTIN,'GET','STOP','LATEST',
     &  ' ',' ',CDIR_GLOB,CRESLT, CZ,DZ,IZ,LZ,RZ)
C---- Look up number of domains, and loop over domains
      CALL PEEKI ('/USER/'//CDIR_GLOB//'/NZN',IONE,NZN,
     &  'STOP',CRESLT, IZ)
      IF (CRESLT .NE. 'GOOD') GOTO 999
      DO IZN = 1, NZN
C
C---- CZONE = solver domain name
        CALL PEEKCA ('/USER/'//CDIR_GLOB//'/CZONE',IZN,CZONE,
     &  'STOP',CRESLT, CZ)
        IF (CRESLT .NE. 'GOOD') GOTO 999
C
C---- Convert to user domain name
```

```
        CALL CONVERT_NAME_S2U ('Domain',CZONE,' ',User_Domain_Name,
     &  CRESLT, CZ,DZ,IZ,LZ,RZ)
        IF (CRESLT .NE. 'GOOD') GOTO 999
        CALL MESAGE( 'WRITE', ' ')
        CALL MESAGE( 'WRITE', '-----------------')
        CALL MESAGE( 'WRITE', 'Domain: '//User_Domain_Name)
        CALL MESAGE( 'WRITE', '-----------------')
        CALL MESAGE( 'WRITE', ' ')
C
C---- Get physics info.
        CDIR_PHYS = ' '
        CALL USER_GET_PHYS_INFO (ROUTIN,'GET','STOP','LATEST',
     &               CZONE,' ',CDIR_PHYS,CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C---- CPHASE = solver phase name (assumes single phase)
        CALL PEEKCA ('/USER/'//CDIR_PHYS//'/CPHASE',IONE,CPHASE,
     &              'STOP',CRESLT, CZ)
        IF (CRESLT .NE. 'GOOD') GOTO 999
C
C---- Convert to user phase name
        CALL CONVERT_NAME_S2U ('Phase',CPHASE,' ',User_Phase_Name,
     &                         CRESLT, CZ,DZ,IZ,LZ,RZ)
        IF (CRESLT .NE. 'GOOD') GOTO 999
C
C---- Set User Variable Name = User_Phase_Name.RTD
C---- (RTD = Residence Time Distribution).
        User_Var_Name = User_Phase_Name(1:LENACT(User_Phase_Name))
     &              // '.RTD'
        CALL MESAGE( 'WRITE', ' ')
        CALL MESAGE( 'WRITE', '-----------------')
        CALL MESAGE( 'WRITE', 'Variable: '//User_Var_Name)
        CALL MESAGE( 'WRITE', '-----------------')
        CALL MESAGE( 'WRITE', ' ')
C
C---- Obtain global values on domain
        CALL USER_GET_GVAR (User_Var_Name, User_Domain_Name,
     &                      'ave', CRESLT, VAR, CZ,DZ,IZ,LZ,RZ)
        IF (CRESLT .NE. 'GOOD') GOTO 999
        CALL MESAGE( 'WRITE', 'RDT Mean Value = '//CFROMR(VAR))
C
        CALL USER_GET_GVAR (User_Var_Name,User_Domain_Name,
     &                      'maxVal', CRESLT, VAR, CZ,DZ,IZ,LZ,RZ)
        IF (CRESLT .NE. 'GOOD') GOTO 999
        CALL MESAGE( 'WRITE', 'RDT Max Value = '//CFROMR(VAR))
C
C---- Obtain zonal meshdata
        CDIR_ZONE = ' '
        CALL USER_GET_MESH_INFO (ROUTIN,'GET','STOP','LATEST',
     &                           CZONE,' ',CDIR_ZONE,CRESLT,
     &                           CZ,DZ,IZ,LZ,RZ)
C
C---- Look up number of Boundary Condition Patches (BCP's),
C---- and loop over BCP's.
        CALL PEEKI ('/USER/'//CDIR_ZONE//'/NBCP',IONE,NBCP,
     &             'STOP',CRESLT, IZ)
        IF (CRESLT .NE. 'GOOD') GOTO 999
        DO IBCP = 1, NBCP
C
C---- CBCP = solver BCP name
          CALL PEEKCA ('/USER/'//CDIR_ZONE//'/CBCP',IBCP,CBCP,
     &               'STOP',CRESLT, CZ)
          IF (CRESLT .NE. 'GOOD') GOTO 999
C
C---- Convert to user BCP name
          CALL CONVERT_NAME_S2U ('Bcp',CBCP,' ',User_BCP_Name,
     &                           CRESLT, CZ,DZ,IZ,LZ,RZ)
          IF (CRESLT .NE. 'GOOD') GOTO 999
          CALL MESAGE( 'WRITE', ' ')
          CALL MESAGE( 'WRITE', '-----------------')
          CALL MESAGE( 'WRITE', 'Boundary: '//User_BCP_Name)
          CALL MESAGE( 'WRITE', '-----------------')
```

```
          CALL MESAGE( 'WRITE', ' ')
C
C---- Obtain global values on BCP
          CALL USER_GET_GVAR (User_Var_Name,User_BCP_Name,
     &                        'ave', CRESLT, VAR, CZ,DZ,IZ,LZ,RZ)
          IF (CRESLT .NE. 'GOOD') GOTO 999
          CALL MESAGE( 'WRITE', 'RDT Mean Value = '//CFROMR(VAR))
C
          CALL USER_GET_GVAR (User_Var_Name,User_BCP_Name,
     &                        'maxVal',CRESLT, VAR, CZ,DZ,IZ,LZ,RZ)
          IF (CRESLT .NE. 'GOOD') GOTO 999
          CALL MESAGE( 'WRITE', 'RDT Max Value = '//CFROMR(VAR))
C
C---- End Loop Over BCP's
        END DO
C
C---- End Loop Over Zones
      END DO
C
C---- Delete Mesh Data
      CALL USER_GET_MESH_INFO (ROUTIN,'DELETE','STOP','LATEST',
     &                         ' ',' ',CDIR_GLOB,CRESLT,
     &                         CZ,DZ,IZ,LZ,RZ)
C
999   CONTINUE
      IF (CRESLT .NE. 'GOOD') THEN
        CALL MESAGE( 'WRITE', 'User abort, CRESLT = '//CRESLT)
        CALL CFXSTP (ROUTIN)
      ENDIF
C
      END
```

19.9.3. Junction Box Example 3: Timestep Control

This Junction Box subroutine should be called at the end of each timestep in a transient run. It is designed to dynamically control the timestep size during the transient run. The algorithm is similar to one of the adaptive timestepping algorithms available in the solver, but serves to illustrate how a junction box can be used to control the timestep size. The algorithm for modifying the timestep is as follows: if the previous timestep needed more iterations than MAXLOOPS, the next timestep size is decreased by a factor FACDEC. If the previous timestep needed fewer iterations than MINLOOPS, the next timestep size is increased by a factor FACINC. The junction box calls USER_GET_TRANS_INFO to retrieve current timestep data and creates a data area called 'NEWDT' to store the next timestep size. The procedure requires a User CEL function called Retrieve_deltat to use the timestep calculated by the junction box. Retrieve_deltat requires an additional argument (passed in through the argument list) called 'Timesteps', which will be the initial timestep size. The routines t_step_control.F and Retrieve_deltat.F are supplied in the *<CFX-ROOT>*/examples/UserFortran/ directory of your CFX installation.

The following is a listing of t_step_control.F:

```
#include "cfx5ext.h"
dllexport(tstepcontrol)
C======================================================================
      SUBROUTINE tstepcontrol(CZ,DZ,IZ,LZ,RZ)
CC
CD Recalculates timestep size to fit into a maximum and minimum
CD coefficient loops.
CC
CC --------------------
CC        Modified
CC --------------------
CC
CV Stacks
```

```
CC
CC -------------------
CC      Modified
CC -------------------
CC
CC -------------------
CC        Output
CC -------------------
CC
CC -------------------
CC Modification History
CC -------------------
CC
CC
CC -------------------
CC Variable dictionary
CC -------------------
CC
CC -------------------
CC Local variables & parameters
CC -------------------
CC
CC   CCLOOP,LCLOOP,NCLOOP    : Coefficient loop name,length, and number
CC   CSTEP,LTSTEP,NTSTEP     : Coefficient step name,length, and number
CC   DT                      : Timestep size
CC   MAXDT,MINDT             : Maximum and minimum timestep size
CC   FACINC,FACDEC           : Increment and decrement factors
CC
CC=====================================================================
CC ------------------------------
C     Preprocessor includes
C ------------------------------
C
#include "MMS.h"
#include "stack_point.h"
#include "cfd_constants.h"
C
C ------------------------------
C        Argument List
C ------------------------------
C
      CHARACTER CZ(*)*(1)
C
      DOUBLE PRECISION DZ(*)
C
      INTEGER IZ(*)
C
      LOGICAL LZ(*)
C
      REAL RZ(*)
C
C ------------------------------
C        External routines
C ------------------------------
C
      LOGICAL LFIL
      EXTERNAL CFROMR,CFROMI,LFIL
C
C ------------------------------
C        Local Variables
C ------------------------------
C
      CHARACTER*4 CRESLT
      CHARACTER*(10) CFROMI
      CHARACTER*(20) CCLOOP,CTSTEP,CFROMR,FNAME
C
      INTEGER LCLOOP,NCLOOP,LTSTEP,NTSTEP,MINLOOPS,MAXLOOPS,IUNIT,ICALL
C
      REAL DT,MAXDT,MINDT,TIME,FACINC,FACDEC
C
C ------------------------------
```

```
C        Local Parameters
C ------------------------------
C
      PARAMETER(IUNIT=91,FNAME='dt_ncloop.dat')
C
C--Saved data.
C
      SAVE ICALL,FACINC,FACDEC,MINDT,MAXDT,MINLOOPS,MAXLOOPS
C
C ------------------------------
C        Stack pointers
C ------------------------------
C
      __stack_point__ pOINT
C
C --------------------------
C    Executable Statements
C --------------------------
C
      IF (ICALL.EQ.0) THEN
        ICALL=1
C
C---- Set defaults:
C     MINDT is used to enforce a positive timestep size
C
        MINDT = 0.
        MAXDT = 1.E10
        MINLOOPS=3
        MAXLOOPS=8
C
C---- Negative means: increase by 10/decrease by 0.1 each given
C     iteration number, positive means: factor itself
C
        FACINC=-40.
        FACDEC=-10.
C
C---- Get from USER area if there:
C
        CALL USER_PEEKR('MINDT',1,MINDT,'SKIP',CRESLT,CZ)
        CALL USER_PEEKR('MAXDT',1,MAXDT,'SKIP',CRESLT,CZ)
        CALL USER_PEEKI('MINLOOPS',1,MINLOOPS,'SKIP',CRESLT,CZ)
        CALL USER_PEEKI('MAXLOOPS',1,MAXLOOPS,'SKIP',CRESLT,CZ)
        CALL USER_PEEKR('FACINC',1,FACINC,'SKIP',CRESLT,CZ)
        CALL USER_PEEKR('FACDEC',1,FACDEC,'SKIP',CRESLT,CZ)
C
C---- Increment by factor 10 each ## time steps
C
        IF (FACINC.LE.0) FACINC=10.**(1./ABS(FACINC))
C
C---- Decrement by factor 0.1 each ## time steps
C
        IF (FACDEC.LE.0) FACDEC=0.1**(1./ABS(FACDEC))
C
C---- Output parameters
C
        CALL MESAGE('WRITE','MINDT is '//CFROMR(MINDT))
        CALL MESAGE('WRITE','MAXDT is '//CFROMR(MAXDT))
        CALL MESAGE('WRITE','MINLOOPS is '//CFROMI(MINLOOPS))
        CALL MESAGE('WRITE','MAXLOOPS is '//CFROMI(MAXLOOPS))
        CALL MESAGE('WRITE','FACINC is '//CFROMR(FACINC)//
     &              ' ('//CFROMI(INT( 1./LOG10(FACINC)+0.5))//')')
        CALL MESAGE('WRITE','FACDEC is '//CFROMR(FACDEC)//
     &              ' ('//CFROMI(INT(-1./LOG10(FACDEC)+0.5))//')')
        IF (IUNIT>0) OPEN(UNIT=IUNIT,FILE=FNAME)
      END IF
C
C---- Get transient info
C
      CALL USER_GET_TRANS_INFO('TSTEPCONTROL','GET',' ',
     &                         CZ,DZ,IZ,LZ,RZ)
C
```

```
C---- Get time step
C
      CALL PEEKR('/USER/TRANS_INFO/DT',1,DT,'STOP',CRESLT,RZ)
C
C---- Write into file
C
      IF (IUNIT>0) THEN
C
C---- Get time and current coefficient loop
C
        CALL PEEKR('/USER/TRANS_INFO/ATIME',1,TIME,'STOP',CRESLT,RZ)
        CALL PEEKI('/USER/TRANS_INFO/NCLOOP',1,NCLOOP,'STOP',CRESLT,IZ)
        WRITE(IUNIT,*) DT,NCLOOP,TIME
C
      END IF
C
C---- Control structure
C
      CALL MESAGE('WRITE','Number of coeff loops was: '//CFROMI(NCLOOP))
C
      IF (NCLOOP.LT.MINLOOPS) THEN
        DT=MIN(FACINC*DT,MAXDT)
        CALL MESAGE('WRITE','Time step increased to: '//CFROMR(DT))
      ELSE IF (NCLOOP.GT.MAXLOOPS) THEN
        DT=FACDEC*DT
        IF (MINDT.LE.0.AND.DT.LE.-MINDT) THEN
          IF (IUNIT.GT.0) CLOSE(IUNIT)
          CALL MESAGE('WRITE','Stop program due to bad convergence.')
          OPEN(UNIT=91,FILE='stp')
          WRITE(91,*) ' '
          CLOSE(91)
        END IF
          DT=MAX(DT,MINDT)
          CALL MESAGE('WRITE','Time step decreased to: '//CFROMR(DT))
      ELSE
        CALL MESAGE('WRITE','Time step stayed at: '//CFROMR(DT))
      END IF
C
C---- Set new time step
C
      IF (.NOT.LFIL('/USER/NEWDT')) THEN
        CALL PSHDIR ('/USER','STOP',CRESLT)
          CALL MAKDAT('NEWDT','REAL','STOP',1,pOINT,CRESLT)
        CALL POPDIR ('STOP',CRESLT)
      ENDIF
      CALL POKER('/USER/NEWDT',1,DT,'STOP',CRESLT,RZ)
C
C---- Release transient data
C
      CALL USER_GET_TRANS_INFO('TSTEPCONTROL','RELEASE',' ',
     &                         CZ,DZ,IZ,LZ,RZ)
C
      END
```

The following is a listing of `Retrieve_deltat.F`:

```
#include "cfx5ext.h"
dllexport(get_timestep)
      SUBROUTINE get_timestep (
     &  NLOC,NRET,NARG,RET,ARGS,CRESLT,CZ,DZ,IZ,LZ,RZ)
CC
CD User routine: retrieves delta t stored in /USER directory
CC
CC --------------------
CC        Input
CC --------------------
CC
CC  NLOC   - size of current locale
CC  NRET   - number of components in result
CC  NARG   - number of arguments in call
```

```
CC  ARGS() - (NLOC,NARG) argument values
CC
CC --------------------
CC      Modified
CC --------------------
CC
CC  Stacks possibly.
CC
CC --------------------
CC        Output
CC --------------------
CC
CC  RET()  - (NLOC,NRET) return values
CC  CRESLT - 'GOOD' for success
CC
CC --------------------
CC       Details
CC --------------------
CC
CC=============================================================================
C
C ------------------------------
C     Preprocessor includes
C ------------------------------
C
#include "MMS.h"
#include "cfd_constants.h"
C
C ------------------------------
C        Global Parameters
C ------------------------------
C
C
C ------------------------------
C        Argument list
C ------------------------------
C
      CHARACTER CRESLT*(*)
      CHARACTER CZ(*)*(1)
C
      DOUBLE PRECISION DZ(*)
C
      INTEGER IZ(*)
      INTEGER NLOC,NARG,NRET
C
      LOGICAL LZ(*)
C
      REAL RZ(*)
      REAL ARGS(NLOC,NARG), RET(NLOC,NRET)
C
C
C ------------------------------
C        External routines
C ------------------------------
C
C
C ------------------------------
C        Local Parameters
C ------------------------------
C
C
C ------------------------------
C        Local Variables
C ------------------------------
C
      CHARACTER*(4) CRES
C
      REAL  DT
C
C --------------------------
C    Executable Statements
```

```
C --------------------------
C
C
C Set success flag.
C
      CRES = 'GOOD'
C
C---- Retrieves DT value. If there is none, it uses default the value
C     passed as argument.
C
      CALL PEEKR('/USER/NEWDT',IONE,DT,'SKIP',CRES,RZ)
      IF (CRES .NE. 'GOOD') DT = ARGS(1,1)
      RET(1,1) = DT
C
      END
```

19.9.4. Junction Box Example 4: Solver Control

This example stops the Solver when the max, min, or mean value of a variable falls in a specified range. The user specifies the variable, its range and whether to use max, min, or mean in the USER section of their CCL. This method can be useful for ventilation cases, where a transient simulation must be performed to determine when a flammable gas or contaminant falls below a certain limit. The routine should be called at the end of each timestep. It is a specialist routine, as it requires knowledge of solver data structures in order to issue the STOP instruction. This may require different operation depending on whether the simulation is transient or steady-state.

The following routine `solver_control.F` is supplied in the *<CFXROOT>*`/examples/UserFortran/` directory of your CFX installation. Some comment lines may be omitted below.

```
#include "cfx5ext.h"
dllexport(stopinteg)
      SUBROUTINE STOPINTEG (CZ,DZ,IZ,LZ,RZ)
C
C Stops solver when a variable integrated quantity falls in a range
C Junction box routine, to be called at the end of each timestep
C
C Include directives
#include "cfd_constants.h"
#include "MMS.h"
#include "stack_point.h"
C
C Arguments
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C External functions
      INTEGER LENACT
      EXTERNAL  LENACT
C
      CHARACTER*15 CFROMR
      EXTERNAL CFROMR
C
C Local parameters
      CHARACTER*(*) ROUTIN
      PARAMETER (ROUTIN = 'STOPINTEG')
C
C Local Variables
      CHARACTER*4 CRESLT
      CHARACTER*(MXDNAM) CDIR_GLOB, CDIR_TEST, CZONE
      CHARACTER*80 USER_DOMAIN, USER_VAR, USER_OPER
C
      REAL USER_MIN, USER_MAX, VAR
C
```

```
C ======================================================================
C Executable Statements
C ======================================================================
C
C ----------------------------------------------------------------------
C Obtain user domain name, assuming one domain only.
C ----------------------------------------------------------------------
C
C---- Obtain global meshdata
C
      CDIR_GLOB = ' '
      CALL USER_GET_MESH_INFO (ROUTIN,'GET','STOP','LATEST',
     &              ' ',' ',CDIR_GLOB,CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C---- CZONE = first solver domain name
C
      CALL PEEKCA ('/USER/'//CDIR_GLOB//'/CZONE',IONE,CZONE,
     &             'STOP',CRESLT, CZ)
C
C---- Convert to user domain name
C
      CALL CONVERT_NAME_S2U ('Domain',CZONE,' ',USER_DOMAIN,
     &                       CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C ----------------------------------------------------------------------
C Get names and values from /USER
C ----------------------------------------------------------------------
C
C Get names and values from /USER
C
      CALL USER_PEEKCS ('UserVarName', USER_VAR, 'STOP', CRESLT, CZ)
      CALL USER_PEEKCS ('UserVarOper', USER_OPER, 'STOP', CRESLT, CZ)
      CALL USER_PEEKR ('UserVarMin',IONE, USER_MIN, 'STOP', CRESLT, CZ)
      CALL USER_PEEKR ('UserVarMax',IONE, USER_MAX, 'STOP', CRESLT, CZ)
C
C ----------------------------------------------------------------------
C Obtain the integrated quantity required
C ----------------------------------------------------------------------
C
      CALL USER_GET_GVAR (USER_VAR, USER_DOMAIN, USER_OPER,
     &                    CRESLT, VAR, CZ,DZ,IZ,LZ,RZ)
C
      CALL MESAGE( 'WRITE', ' ')
      CALL MESAGE( 'WRITE', '--------------------------------')
      CALL MESAGE( 'WRITE', 'Entered subroutine '//ROUTIN)
      CALL MESAGE( 'WRITE', '----------------------0---------')
      CALL MESAGE( 'WRITE', ' ')
      CALL MESAGE( 'WRITE', 'Domain Name  = '//USER_DOMAIN)
      CALL MESAGE( 'WRITE', 'User Variable  = '//USER_VAR)
      CALL MESAGE( 'WRITE', 'User Operation = '//USER_OPER)
      CALL MESAGE( 'WRITE', 'Minimum Value  = '//CFROMR(USER_MIN))
      CALL MESAGE( 'WRITE', 'Maximum Value  = '//CFROMR(USER_MAX))
      CALL MESAGE( 'WRITE', 'Integrated Value = '//CFROMR(VAR))
C
C ----------------------------------------------------------------------
C If conditions are met, print a banner and stop after this timestep
C ----------------------------------------------------------------------
C
C Stop achieved by setting /FLOW/ALGORITHM/CONTROL/NEXT_TSTEP to FALSE
      IF (VAR.GE.USER_MIN .AND. VAR.LE.USER_MAX) THEN
C
        CALL MESAGE( 'WRITE', ' ')
        CALL MESAGE( 'WRITE', '-------------------------------------')
        CALL MESAGE( 'WRITE', 'Stopping the solver in SOLVER_CONTROL ')
        CALL MESAGE( 'WRITE', '-------------------------------------')
        CALL MESAGE( 'WRITE', ' ')
C
C Stop flag in case of transient simulation
C
      CALL POKEL ('/FLOW/ALGORITHM/CONTROL/NEXT_TSTEP',IONE,.FALSE.,
     &              'STOP', CRESLT, LZ)
```

```
C
C Stop flag in case of steady-state simulation
C
C        CALL POKEL ('/FLOW/ALGORITHM/CONTROL/STOP_TSL',IONE,.TRUE.,
C     &              'STOP', CRESLT, LZ)
CC
      END IF
C
      END
```

19.9.5. Junction Box Example 5: Transient Information

This is an example of how to obtain transient data from within a User Fortran junction box routine. In a junction box routine, it is necessary to loop over all zones (domains). For a single domain problem, there will be only one zone, but the approach shown is more general, because it is possible to set a different timestep in each domain. The code shown first obtains and prints out non-zone-specific transient data, and then loops over all zones and obtains zone-specific transient data. Standard CFX memory management system routines, such as PSHDIR, POPDIR, PEEKI and PEEKR, are used to move around the data-structures and obtain the required data.

To use these files, it will first be necessary to modify the file `run.ccl` so that the shared library path is set to the correct location on your filesystem. The User Fortran will then need to be compiled as follows:

```
cfx5mkext -name transinfo transinfo.F
```

The case may be run once as follows:

```
cfx5solve -def transient.def -ccl run.ccl.
```

A restart may be carried out from the first run in order to complete additional time-steps and demonstrate how the accumulated timesteps count and value operates:

```
cfx5solve -def transient_001.res -ccl run.ccl
```

This routine `transinfo.F` is supplied in the *<CFXROOT>*`/examples/UserFortran/` directory of your CFX installation.

```
#include "cfx5ext.h"
dllexport(get_tstep_info)
      SUBROUTINE GET_TSTEP_INFO (CZ, DZ, IZ, LZ, RZ)
      IMPLICIT NONE
C
C Example of obtaining transient info for all zones
C
C ======================================================================
C Include directives
C ======================================================================
C
#include "MMS.h"
#include "cfd_constants.h"
C
C ======================================================================
C Arguments
C ======================================================================
C
C Stacks
      CHARACTER*(1) CZ(*)
      DOUBLE PRECISION DZ(*)
      INTEGER IZ(*)
```

```
      LOGICAL LZ(*)
      REAL RZ(*)
C
C ======================================================================
C External functions
C ======================================================================
C
C Function to concatonate an integer on to the end of a string
      CHARACTER*(MXDNAM) CCATI
      EXTERNAL CCATI
C
C ======================================================================
C Local variables
C ======================================================================
C
C Result
      CHARACTER*4 CRESLT
C Zone (domain)
      CHARACTER*(MXDNAM) CZONE
C Path name
      CHARACTER*(MXPNAM) CDRNAM
C
C Number of zones & iterator
      INTEGER NZN, IZN
C Number of timesteps, accumulated timesteps, coefficient loops
      INTEGER NTSTEP, ATSTEP, NCLOOP
C
C Time, accumulated time, timestep value
      REAL DT, TIME_DTEND, RTIME_DTEND
C
C ======================================================================
C Executable statements
C ======================================================================
C
C ----------------------------------------------------------------------
C Obtain and write non zone specific data
C ----------------------------------------------------------------------
C
C Obtain number of timesteps, accumulated timesteps & coefficient loops
      CALL PSHDIR ('/FLOW/SOLUTION', 'STOP', CRESLT)
      CALL PEEKI ('NTSTEP', IONE, NTSTEP, 'STOP', CRESLT, IZ)
      CALL PEEKI ('ATSTEP', IONE, ATSTEP, 'STOP', CRESLT, IZ)
      CALL PSHDIR (CCATI('TSTEP',ATSTEP), 'STOP', CRESLT)
      CALL PEEKI('NCLOOP', IONE, NCLOOP, 'STOP', CRESLT, IZ)
      CALL POPDIR ('STOP', CRESLT)
      CALL POPDIR ('STOP', CRESLT)
C
C Write to standard output
      WRITE(*,*)
      WRITE(*,*) '            TIMESTEP:', NTSTEP
      WRITE(*,*) 'ACCUMULATED TIMESTEP:', ATSTEP
      WRITE(*,*) '    COEFFICIENT LOOP:', NCLOOP
C
C ----------------------------------------------------------------------
C Obtain and write zone specific data
C ----------------------------------------------------------------------
C
C Determine the number of zones (domains)
      CALL PEEKI ('/FLOW/GEOMETRY/NZN', IONE, NZN, 'STOP', CRESLT, IZ)
C
C Loop over all zones
      DO IZN = 1,NZN
         CALL PEEKCA ('/FLOW/GEOMETRY/CZONE',
     &    IZN, CZONE, 'STOP', CRESLT, CZ)
C
C Obtain time, accumulated time & timestep value for this zone
         CDRNAM = '/FLOW/SOLUTION/LATEST/' // CZONE
         CALL PSHDIR (CDRNAM, 'STOP', CRESLT)
         CALL PEEKR ('TIME_DTEND', IONE, TIME_DTEND, 'STOP', CRESLT, RZ)
         CALL PEEKR ('RTIME_DTEND',IONE,RTIME_DTEND, 'STOP', CRESLT, RZ)
         CALL PEEKR ('DT',         IONE, DT, 'STOP', CRESLT, RZ)
```

```
         CALL POPDIR ('STOP', CRESLT)
C
C Write to standard output
         WRITE(*,*) '                ZONE:', IZN
         WRITE(*,*) '                  TIME:', RTIME_DTEND
         WRITE(*,*) '      ACCUMULATED TIME:', TIME_DTEND
         WRITE(*,*) '        TIMESTEP VALUE:', DT
      END DO
C
      END
```

19.10. Using CFX-4 Routines in CFX

With the availability of CFX Expression Language (CEL), many of the documented user routines presented in CFX can be set up using CEL in CFX. For details, see CFX Expression Language (CEL) in the *CFX Reference Guide*.The following table lists some user routines from CFX-4, and how they can be set up in CFX.

CFX routine and description	How to set up in CFX
USRBCS A routine that enables the specification of real information at inlets, mass flow boundaries, pressure boundaries, walls and conducting boundaries, both at the start of a run and on each timestep.	An inlet profile using User Fortran is described in Junction Box Example 1. For details, see Junction Box Example 1: Profile Boundary Conditions (p. 706). Time-varying boundary conditions can be set up using CEL, with an expression containing **t**.
USRBF To enable you to specify body forces in the momentum equations.	This routine has been replaced by both CEL and loss models. For details, see Sources (p. 80).
USRDEN Used to overwrite the default equation of state in a compressible flow, or to set a varying density in an incompressible flow.	There is no need to set $\partial \rho / \partial p$ as this is calculated by the solver. Density can be specified using CEL.
USRDIF used to overwrite the default diffusion coefficients in each transport equation. The routine is called for each equation. The equation label enters through the argument list.	Can be specified using CEL, except for orthotropic diffusivity.
USRDMP Enables more complex control of output files.	Output control in CFX-Pre largely replaces the need for this routine. For details, see Output Control in the *CFX-Pre User's Guide*.
USRDRG Enables you to overwrite the standard drag factor on particles in the particle transport model. This would normally involve setting	Particle User Fortran can be used in CFX. For details, see Particle User Sources (p. 374).

CFX routine and description	How to set up in CFX
the CD factor as a function of the relative speed of the particle and surrounding fluid.	
USRGRD Enables you to define a grid and to specify movement of the grid with time.	This is covered by the explicit mesh movement feature of the moving mesh capability. For details, see Mesh Deformation in the *CFX-Pre User's Guide*.
USRINT Used to overwrite the initial conditions, particularly on restarts.	CCL and/or CEL can be used here.
USRIPM Used to set complex interphase mass transfer terms.	CEL
USRIPT Used to set complex interphase transfer terms.	CEL
USRRAC To enable you to inspect or change reaction rate constants and their derivatives.	CEL and/or CCL
USRSLP To enable you to specify the slip velocity in the algebraic slip model.	CEL can be used.
USRSPH To enable you to specify specific heat as an arbitrary constitutive function.	CEL can be used.
USRSRC This routine enables you to intervene and change the equations; in particular, to add sources and sinks of momentum and energy.	This can be carried out using User CEL. An example is available in User CEL Example 1: User Defined Momentum Source (p. 698).
USRTRN The routine is called after the initial guess and at the end of each timestep. The routine can be used to monitor the calculation, or to produce special output for each timestep. It may also be used at the end of the job to calculate additional quantities from the basic solution variables.	The User Junction Box routines can handle this type of application.
USRVIS	CEL

CFX routine and description	How to set up in CFX
enables you to specify a variable viscosity as a function of any combination of the other variables for use in a laminar flow calculation.	
USRWRK This routine enables you to reserve work space that may then be accessed in any of the user-defined routines.	This operation is implemented by making space available in the /USER data area. Space can be created using the MAKDAT routine. For details, see MAKDAT (p. 687).

19.11. State Point Evaluation

The user state point query function enables the evaluation of thermodynamic properties at a user specified state. This is accomplished by calling the following utility routine from within User Fortran subroutines (User CEL functions or Junction Boxes):

```
 CALL USER_STATEPT(VAR,CPROCALC,CPROPIND,RPROPIND,NPROPIND,
&                  CERACT,CRESLT,CZ,DZ,IZ,LZ,RZ)
```

The input arguments for this utility are:

- CHARACTER*(*) CPROPCALC - Name of requested output property variable.
- CHARACTER*(*) CPROPIND(NPROPIND) - Array of independent thermodynamic property names.
- REAL RPROPCALC(NPROPIND) - Array of independent thermodynamic property values.
- INTEGER NPROPIND - Number of independent thermodynamic properties.
- CHARACTER*4 CERACT - Error action: STOP to stop on error, or SKIP to continue silently, with the result code returned in CRESLT.
- CHARACTER*1(*) CZ - CHARACTER data stack.
- DOUBLE DZ - DOUBLE PRECISION data stack.
- DOUBLE IZ - INTEGER data stack.
- DOUBLE LZ - LOGICAL data stack.
- DOUBLE RZ - REAL data stack.

The output arguments for this utility are:

- REAL VAR - Return value of CPROPCALC at the user-specified thermodynamic state.
- CHARACTER*4 CRESLT - result code, which is GOOD if successful.

19.11.1. General Usage Notes

The following are guidelines on how to set the various input arguments:

- The function call to USER_STATEPT must provide the input arguments in terms of thermodynamic properties and mixture composition, if applicable.
- User state point function calls do not support phases that use the "Saturation Temperature" or the "Small Droplet Temperature" heat transfer models.
- Each user state point function call must specify the material for which the output property is evaluated. The equation of state and constitutive relationship for the specified material are then used to evaluate the property.
- User state point function calls can accept an arbitrary number of single valued input arguments.
- Input arguments can be only thermodynamic properties and composition. Other variables are not supported.
- User state point function calls return a single valued result from the specified input arguments. The requested result must be a material property that is not one of the input arguments.
- In a simulation that involves disconnected domains, the specification of unique fluid names is required. Otherwise, the CFX-Solver may not be able to identify the corresponding materials.

19.11.2. Supported Independent Thermodynamic Input Properties

The following input properties are currently supported with USER_STATEPT:

- Temperature
- Absolute Pressure
- Absolute Pressure and Temperature
- Absolute Pressure and Enthalpy
- Absolute Pressure and Entropy
- Pressure
- Pressure and Temperature
- Pressure and Enthalpy
- Pressure and Entropy
- Mixture Composition (if the material is a variable composition mixture)

The format of the input arguments is an array of property names. The names must be set to the valid short or long names.

19.11.3. Supported Thermodynamic Output Properties

The following output properties are currently supported with USER_STATEPT:

- Pressure and Absolute Pressure

- Temperature
- Static Enthalpy
- Static Entropy
- Specific Heat at Constant Pressure
- Specific Heat at Constant Volume
- Dynamic Viscosity
- Density
- Isentropic Compressibility
- Isothermal Compressibility
- Local Speed of Sound

19.11.4. Input and Output Property Name Format

The input and output arguments are specified using standard CFX Expression Language syntax. Both short and long names are supported. The syntax of the arguments is either `<fluid name>.<property>` or `<component>.<property>`, where `<fluid name>` is the name of the fluid, `<component>` is the mixture component name (if needed) and `<property>` is the name of the property in short or long form.

19.11.4.1. Pure Substance Example

The following code section shows the use of short or long names for input and output properties for a simulation that involves a pure substance:

```
C
C---- Input thermodynamic properties:
C     ? Temperature and pressure
C
      CPROPIND(1) = 'T'
      CPROPIND(2) = 'pabs'
C
C---- Input values for temperature and pressure
C
      RPROPIND(1) = 498.15
      RPROPIND(2) = 101325.
C
C-------------------------------------------------------------------------
C     Short names for input properties, long name for output property
C-------------------------------------------------------------------------
C
      CPROPCALC = 'Air Ideal Gas.Static Enthalpy'
      CALL USER_STATEPT(H,CPROPCALC,CPROPIND,RPROPIND,
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C-------------------------------------------------------------------------
C     Long names for input properties, short names for output property
C-------------------------------------------------------------------------
C
      CPROPIND(1) = 'Temperature'
      CPROPIND(2) = 'Absolute Pressure'
C
```

```
      CPROPCALC = 'Air Ideal Gas.Entropy'
      CALL USER_STATEPT(S,CPROPCALC,CPROPIND,RPROPIND,
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C---------------------------------------------------------------------
C     Mixed long and short names for input/output properties
C---------------------------------------------------------------------
C
      CPROPIND(1) = 'T'
      CPROPIND(2) = 'Absolute Pressure'
C
      CPROPCALC = 'Air Ideal Gas.Cp'
      CALL USER_STATEPT(Cp,CPROPCALC,CPROPIND,RPROPIND,
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
…
```

19.11.4.2. Mixture Example

The following code section shows the use of short or long names for input and output properties for a simulation that involves a mixture consisting of four different materials:

```
C
C---- Input thermodynamic properties:
C     ' Temperature and pressure
C
      CPROPIND(1) = 'T'
      CPROPIND(2) = 'pabs'
C
C---- Input values for temperature and pressure
C
      RPROPIND(1) = 498.15
      RPROPIND(2) = 101325.
C
C---- Specify mass fractions for each fluid component material (short
C     names)
C
      CPROPIND(3) = 'N2 Ideal Gas.mf'
      CPROPIND(4) = 'O2 Ideal Gas.mf'
      CPROPIND(5) = 'CO2 Ideal Gas.mf'
      CPROPIND(6) = 'CO2 Const Prop.mf'
C
C---- Fluid mixture mass fractions
C
      RPROPIND(3) = 0.79
      RPROPIND(4) = 0.18
      RPROPIND(5) = 0.01
      RPROPIND(6) = 0.01
C
C---------------------------------------------------------------------
C     Mixture enthalpy, short names for component mass fractions
C---------------------------------------------------------------------
C
      CPROPCALC = 'MyMixture.Static Enthalpy'
      CALL USER_STATEPT(H,CPROPCALC,CPROPIND,RPROPIND,
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C---------------------------------------------------------------------
C     Compute mixture specific heat capacity, long names for
C     component mass fractions
C---------------------------------------------------------------------
C
      CPROPIND(3) = 'N2 Ideal Gas.Mass Fraction'
      CPROPIND(4) = 'O2 Ideal Gas.Mass Fraction'
      CPROPIND(5) = 'CO2 Ideal Gas.Mass Fraction'
      CPROPIND(6) = 'CO2 Const Prop.Mass Fraction'
C
      CPROPCALC = 'MyMixture.Cp'
      CALL USER_STATEPT(CpM,CPROPCALC,CPROPIND,RPROPIND,
```

```
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C---------------------------------------------------------------------
C     Compute specific heat capacity for one of the mixture components
C     use long names for component mass fractions
C---------------------------------------------------------------------
C
      CPROPCALC = 'MyMixture.O2 Ideal Gas.Cp'
      CALL USER_STATEPT(CpO2,CPROPCALC,CPROPIND,RPROPIND,
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
…
```

19.11.5. Supported Material Types

The state point utility function can evaluate thermodynamic properties of pure substances, mixtures of pure substances and specified components of a mixture.

19.11.6. Error Checks

The following error checking is performed for user state point functions:

- Validation of input and output thermodynamic properties.
- Check if output property can be computed from specified input data.
- For fluid mixtures, the following additional checks are performed:
  - Check if fluid is a mixture.
  - Check if specified components are valid mixture components.
  - Check if all mixture components have been specified.
  - Check if specified mass fractions sum up to unity. Mass fractions are assumed to be in the range between 0 and 1 and no clipping is performed if this is not the case. The user specified mass fractions are renormalized in case they do not sum up to unity with a tolerance of 2%.

19.11.7. USER_STATEPT Example 1

The following example shows the usage of the user state point function to calculate the specific heat capacity for various materials. The heat capacity is calculated for a mixture (`MyAir`) as well as for each component of the mixture (`MyAir.N2 Ideal Gas`, `MyAir.O2 Ideal Gas`, `MyAir.CO2 Ideal Gas` and `MyAir.CO2 Const Prop`).

The CCL that is required to plot the efficiency in the CFX-Solver Manager is as follows:

```
LIBRARY:
  CEL:
    FUNCTION: Function 1
      Argument Units = [[],[K],[Pa],[],[],[],[]]
      Option = User Function
      Result Units = [J kg^-1 K^-1]
      User Routine Name = User Routine 1
    END
    EXPRESSIONS:
      MyCp1 = Function 1(1,Tave,Pave,N2mf,O2mf,CO2mf,CO21mf)
      MyCp2 = Function 1(2,Tave,Pave,N2mf,O2mf,CO2mf,CO21mf)
      MyCp3 = Function 1(3,Tave,Pave,N2mf,O2mf,CO2mf,CO21mf)
```

```
      MyCp4 = Function 1(4,Tave,Pave,N2mf,O2mf,CO2mf,CO21mf)
      MyCp5 = Function 1(5,Tave,Pave,N2mf,O2mf,CO2mf,CO21mf)
#
      A = 0.1 [J kg^-1 K^-2]
      MyCp  = 1004 [J kg^-1 K^-1] + A*Temperature
      CpCO2 = 851  [J kg^-1 K^-1] + A*Temperature
      CpO2  = 920  [J kg^-1 K^-1] + A*Temperature
      CpN2  = 1040 [J kg^-1 K^-1] + A*Temperature
#
      Tave   = massFlowAve(Temperature)@Outflow
      Pave   = massFlowAve(Absolute Pressure)@Outflow
      N2mf   = massFlowAve(N2 Ideal Gas.Mass Fraction)@Outflow
      O2mf   = massFlowAve(O2 Ideal Gas.Mass Fraction)@Outflow
      CO2mf  = massFlowAve(CO2 Ideal Gas.Mass Fraction)@Outflow
      CO21mf = massFlowAve(CO2 Const Prop.Mass Fraction)@Outflow
    END
  END
  USER ROUTINE DEFINITIONS:
    USER ROUTINE: User Routine 1
      Calling Name = my_test
      Library Name = mytest
      Library Path = ..
      Option = User CEL Function
    END
  END
END
FLOW: Flow Analysis 1
  OUTPUT CONTROL:
    MONITOR OBJECTS:
      MONITOR POINT: Monitor Point 1
        Expression Value = MyCp1
        Option = Expression
      END
      MONITOR POINT: Monitor Point 2
        Expression Value = MyCp2
        Option = Expression
      END
      MONITOR POINT: Monitor Point 3
        Expression Value = MyCp3
        Option = Expression
      END
      MONITOR POINT: Monitor Point 4
        Expression Value = MyCp4
        Option = Expression
      END
      MONITOR POINT: Monitor Point 5
        Expression Value = MyCp5
        Option = Expression
      END
    END
    MONITOR OBJECTS:
      MONITOR BALANCES:
        Option = Full
      END
      MONITOR FORCES:
        Option = Full
      END
      MONITOR PARTICLES:
        Option = Full
      END
      MONITOR RESIDUALS:
        Option = Full
      END
      MONITOR TOTALS:
         Option = Full
      END
    END
    RESULTS:
      File Compression Level = Default
      Option = Standard
    END
```

```
  END
END
```

The Fortran source code for this example is given below:

```
#include "cfx5ext.h"
dllexport(my_test)
      SUBROUTINE MY_TEST (
     &  NLOC, NRET, NARG, RET, ARGS, CRESLT, CZ,DZ,IZ,LZ,RZ )
CC
CD User routine: template for user CEL function
CC
CC --------------------
CC        Input
CC --------------------
CC
CC  NLOC   - size of current locale
CC  NRET   - number of components in result
CC  NARG   - number of arguments in call
CC  ARGS() - (NLOC,NARG) argument values
CC
CC --------------------
CC      Modified
CC --------------------
CC
CC  Stacks possibly.
CC
CC --------------------
CC        Output
CC --------------------
CC
CC  RET()  - (NLOC,NRET) return values
CC  CRESLT - 'GOOD' for success
CC
CC --------------------
CC       Details
CC --------------------
CC
CC  Template routine for user state point code
CC
CC=========================================================================
C
C ----------------------------
C        Argument list
C ----------------------------
C
      INTEGER   NLOC,NARG,NRET
C
      CHARACTER CRESLT*(*)
C
      REAL      ARGS(NLOC,NARG), RET(NLOC,NRET)
C
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C ----------------------------
C        External routines
C ----------------------------
C
      INTEGER  LENACT
      EXTERNAL LENACT
C
C ----------------------------
C        Local Parameters
C ----------------------------
C
C ----------------------------
```

```
C        Local Variables
C ------------------------------
C
      INTEGER        NPROPIND, NTESTPROP, IMAT, NMAT, IPROP
      CHARACTER*(80) CPROPCALC, CPROPIND(6)
C
      REAL           RPROPIND(6)
C
      CHARACTER*(30) INDPROP(13)
      CHARACTER*(30) MAT(5)
C
      DATA INDPROP /'Static Enthalpy','Static Entropy','Cp','Cv',
     &              'Dynamic Viscosity','Density',
     &              'Pressure','Absolute Pressure',
     &              'Temperature','Absolute Temperature',
     &              'Isentropic Compressibility',
     &              'Isothermal Compressibility',
     &              'Local Speed of Sound'/
C
      DATA MAT     /'MyAir','MyAir.N2 Ideal Gas',
     &              'MyAir.O2 Ideal Gas','MyAir.CO2 Ideal Gas',
     &              'MyAir.CO2 Const Prop'/
C
C ------------------------------
C        Stack pointers
C ------------------------------
C
C=======================================================================
C
C --------------------------
C    Executable Statements
C --------------------------
C
C---- Initialize RET(1:NLOC*NRET) to zero.
C
      CALL SET_A_0(RET,NLOC*NRET)
C
C-----------------------------------------------------------------------
C     Setup data required for state point calculation
C-----------------------------------------------------------------------
C
      NPROPIND    = 6
C
      CPROPIND(1) = 'Temperature'
      CPROPIND(2) = 'Absolute Pressure'
C
      CPROPIND(3) = 'N2 Ideal Gas.mf'
      CPROPIND(4) = 'O2 Ideal Gas.mf'
      CPROPIND(5) = 'CO2 Ideal Gas.mf'
      CPROPIND(6) = 'CO2 Const Prop.mf'
C
C---- 'Material' number:
C     --> '1': Mixture
C         '2': N2 Ideal Gas
C         '3': O2 Ideal Gas
C         '4': CO2 Ideal Gas
C         '5': CO2 Const Prop

      IMAT  = ARGS(NLOC,1)
C
C---- Input values of temperature and absolute pressure
C
      RPROPIND(1) = ARGS(NLOC,2)
      RPROPIND(2) = ARGS(NLOC,3)
C
C---- Input values of component mass fractions
C
      RPROPIND(3) = ARGS(NLOC,4)
      RPROPIND(4) = ARGS(NLOC,5)
      RPROPIND(5) = ARGS(NLOC,6)
      RPROPIND(6) = ARGS(NLOC,7)
```

```
C
C-----------------------------------------------------------------------
C     Compute return variable value
C-----------------------------------------------------------------------
C
C---- Compute Specific Heat capacity (= property '3' in above list)
C
      IPROP = 3
C
      IF (IMAT.GT.1) THEN
        NPROPIND = 2
      ENDIF
C
C---- Create output property name
C
      CPROPCALC = MAT(IMAT)(1:LENACT(MAT(IMAT)))//'.'//
     &            INDPROP(IPROP)
      CALL USER_STATEPT(RVALUE,CPROPCALC,CPROPIND,RPROPIND,
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C---- Copy output property value to RET(1:NLOC,1).
C     --> NLOC = 1
C
      RET(1,1) = RVALUE
C
      END
```

19.11.8. USER_STATEPT Example 2

Another application of the user state point function could be to calculate the isentropic expansion efficiency of a device. The following example code shows the usage of the USER_STATEPT function to accomplish this task. Input arguments for this routine are the mass flow averaged values of total enthalpy (H_1) and entropy (S_1) at a reference plane 1 as well as the mass flow averaged values of total enthalpy (H_2) and total pressure (P_2) at a second reference plane, such as an outlet. The isentropic expansion efficiency is defined as:

$$\eta_E = \frac{H_2 - H_1}{H_{S'} - H_1} \tag{19.6}$$

With the isentropic enthalpy, $H_{S'}$, being the enthalpy $h = h\left(S_1, P_2\right)$.

To calculate the isentropic efficiency, the following input data is provided to USER_STATEPT:

- Number of independent input properties: 2
- Input property names: `Static Entropy`, `Pressure`
- Input property values: S1, P2
- Output property name: `Air Ideal Gas.Static Enthalpy`

Note:

Even though P2 holds the value of total pressure, the input property name must be called `Pressure`.

The CCL that is required to plot the efficiency in the CFX-Solver Manager is as follows:

```
LIBRARY:
  USER ROUTINE DEFINITIONS:
    USER ROUTINE: User Routine 1
      Calling Name = my_eff
      Library Name = efficiency
      Library Path = ..
      Option = User CEL Function
    END
  END
  CEL:
    FUNCTION: Function 1
      Argument Units = [J/kg],[J/kg],[J/kg/K],[Pa]
      Option = User Function
      Result Units = []
      User Routine Name = User Routine 1
    END
    EXPRESSIONS:
      H1 = massFlowAve(Total Enthalpy in Stn Frame)@Inflow
      H2 = massFlowAve(Total Enthalpy in Stn Frame)@Outflow
      S1 = massFlowAve(Static Entropy)@Inflow
      P2 = massFlowAve(Total Pressure in Stn Frame)@Outflow
#
      Eff = Function 1(H1,H2,S1,P2)
    END
  END
END
FLOW: Flow Analysis 1
  OUTPUT CONTROL:
    MONITOR OBJECTS:
      MONITOR BALANCES:
        Option = Full
      END
      MONITOR FORCES:
        Option = Full
      END
      MONITOR PARTICLES:
        Option = Full
      END
      MONITOR RESIDUALS:
        Option = Full
      END
      MONITOR TOTALS:
         Option = Full
      END
      MONITOR POINT: Eff1
        Option = Expression
        Expression Value = Eff
      END
    END
    RESULTS:
      File Compression Level = Default
      Option = Standard
    END
  END
END
```

The Fortran source code for the expansion efficiency calculation is given below:

```
#include "cfx5ext.h"
dllexport(my_eff)
      SUBROUTINE MY_EFF (
     &  NLOC,NRET,NARG,RET,ARGS,CRESLT, CZ,DZ,IZ,LZ,RZ)
CC
CD User routine: Compute isentropic efficiency
CC
CC --------------------
CC        Input
CC --------------------
CC
CC  NLOC   - size of current locale
```

```
CC  NRET   - number of components in result
CC  NARG   - number of arguments in call
CC  ARGS() - (NLOC,NARG) argument values
CC
CC --------------------
CC      Modified
CC --------------------
CC
CC  Stacks possibly.
CC
CC --------------------
CC       Output
CC --------------------
CC
CC  RET()  - (NLOC,NRET) return values
CC  CRESLT - 'GOOD' for success
CC
CC --------------------
CC       Details
CC --------------------
CC
CC  Template routine for user state point code
CC
CC======================================================================
C
C ------------------------------
C        Argument list
C ------------------------------
C
      INTEGER   NLOC,NARG,NRET
      CHARACTER CRESLT*(*)
      REAL      ARGS(NLOC,NARG), RET(NLOC,NRET)
C
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C ------------------------------
C        External routines
C ------------------------------
C
C ------------------------------
C        Local Parameters
C ------------------------------
C
C ------------------------------
C        Local Variables
C ------------------------------
C
      INTEGER        NPROPIND
      CHARACTER*(80) CPROPCALC, CPROPIND(2)
C
      REAL           RPROPIND(2), H1, H2, HIS, S1, SN, P2, DHIS, DHLOC,
     &               ETA_E
C
C ------------------------------
C        Stack pointers
C ------------------------------
C
C======================================================================
C
C --------------------------
C    Executable Statements
C --------------------------
C
C---- Initialize RET(1:NLOC*NRET) to zero.
C
      CALL SET_A_0(RET,NLOC*NRET)
C
```

```
C-----------------------------------------------------------------------
C     Averaged values of inlet total enthalpy, inlet entropy, outlet
C     total enthalpy and outlet total pressure are passed into his
C     routine.
C     Mass flow averaging is used.
C-----------------------------------------------------------------------
C
      H1 = ARGS(NLOC,1)
      H2 = ARGS(NLOC,2)
      S1 = ARGS(NLOC,3)
      P2 = ARGS(NLOC,4)
C
C-----------------------------------------------------------------------
C     Compute isentropic efficiency
C-----------------------------------------------------------------------
C
C---- Number of input properties
C
      NPROPIND    = 2
C
C---- Input properties are entropy and pressure
C
      CPROPIND(1) = 'Static Entropy'
      CPROPIND(2) = 'Pressure'
C
C---- Input values are s1 and p2
C
      RPROPIND(1) = S1
      RPROPIND(2) = P2
C
C---- Compute isentropic enthalpy, h = h(s1,p2)
C
      CPROPCALC = 'Air Ideal Gas.Static Enthalpy'
      CALL USER_STATEPT(HIS,CPROPCALC,CPROPIND,RPROPIND,
     &                  NPROPIND,'SKIP',CRESLT, CZ,DZ,IZ,LZ,RZ)
C
C---- Expansion efficiency
C
      DHIS = HIS - H1
      DHLOC = H2 - H1
C
      IF (DHIS.LT.0.0 .AND.DHLOC.LT.0.0 .AND.DHIS.LT.DHLOC) THEN
        ETA_E = (H2 - H1)/(HIS - H1 + SN)
      ELSE
        ETA_E = 1.0
      ENDIF
C
C-----------------------------------------------------------------------
C
C---- Copy output property value to RET(1:NLOC,1).
C     --> NLOC = 1
C
      RET(1,1) = ETA_E
C
C---- Set success flag.
C
      CRESLT = 'GOOD'
C
C-----------------------------------------------------------------------
      END
```

19.12. Calculating the Particle Drag Coefficient

The User Defined option enables you to use a User Fortran subroutine to calculate the particle drag coefficient. A particle user routine example is presented later in this section.

To use a particle user routine, you should first write the Fortran routine and create a shared library. Next, you should create a User Routine of type Particle User Routine in CFX-Pre. When defining the domain, particle user sources are enabled by selecting the **Particle User Source** check box for the appropriate quantity, then selecting the routine from the drop-down list. For details, see Creating the Shared Libraries (p. 640) and Particle User Routines in the *CFX-Pre User's Guide*.

The Argument Variables list should contain the fluid and particle variables needed to calculate the source (help on how to use variable names in CFX, as well as a list of all variables, can be found in the VARIABLES file, in the `/etc/` directory of your CFX installation). The Return Variables list enables you to return only the Particle Drag Coefficient.

19.12.1. USER_PARTICLE_INFO Routine

The USER_PARTICLE_INFO routine enables you to access or update particle data that is not directly accessible in the user interface. For example:

- particle seed (ISEED) (required for anything that uses random numbers)
- particle mode (PMODE) (required to distinguish 'regular' particles from 'wall (film)' particles)
- breakup flag (BREAKUP) (indicates whether particle broke up due to external forces)

The USER_PARTICLE_INFO routine is called with three arguments (where all three arguments are integer values): For example `CALL USER_PARTICLE_INFO(IACTION,IWHAT,IVALUE)`.

IACTION can have two values, 1 or 2.

- 1 transfers the value stored in IVALUE back to the particle tracker to be used there
- 2 picks up the requested data from internal tracker data structures and stores it in IVALUE for use in User Fortran code.

IWHAT can have values of 1, 2 or 3, and these refer to the different internal particle properties ISEED, PMODE and BREAKUP respectively.

The proper use of USER_PARTICLE_INFO is as follows (the example is for a particle seed):

```
C
C---- Pick up particle seed from tracker
C
      IACTION = 2
      IWHAT = 1
      CALL USER_PARTICLE_INFO(IACTION,IWHAT,ISEED)

<User code follows, that needs and updates ISEED>

C
C---- Store updated particle seed back to tracker
C
      IACTION = 1
      IWHAT = 1
      CALL USER_PARTICLE_INFO(IACTION,IWHAT,ISEED)
```

You can retrieve or write an integer IVALUE to any of these data properties, which in turn will affect the internal particle calculations.