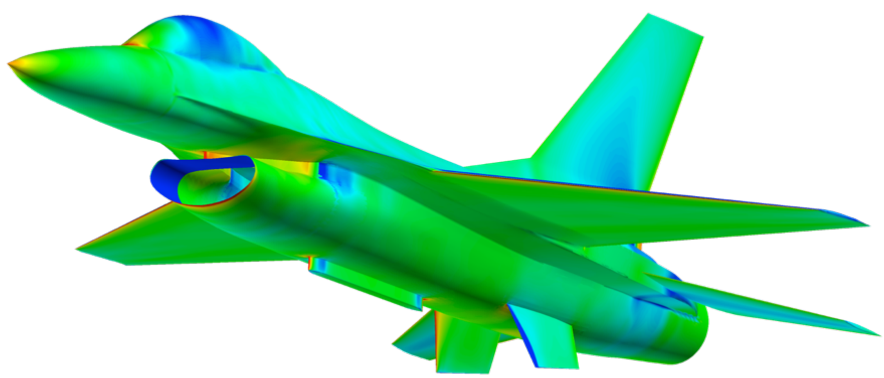
THE EZAIR CFD SOLVERS SUITE:

EZAir

THEORETICAL AND USER'S MANUAL

Version 5.43-OWL

ISCFDC Report 2025-10



Prepared by

Ya'eer Kidron, Sahar Shpitz, and Yuval Levy

ISCFDC LTD

Copyright and Trademark Information

Contents

Abstract **xxi**

1 Introduction **1**

2 Physical Models **3**

2.1 Introduction 3

2.2 Single-component Perfect Gas (SPG) 3

2.2.1 Governing Equations 3

2.2.2 Integral Form For Moving Grids 5

2.2.3 Thermodynamic Properties 6

2.2.4 Transport Properties 7

2.3 Selection of Physical Model 8

3 Turbulence Models **9**

3.1 Baldwin-Lomax Turbulence Model 10

3.2 RANS Turbulence Model Equations 12

3.3 The Spalart Allmaras Model 14

3.3.1 SA-Edwards Model 16

3.3.2 SA-2020 Model 17

3.3.3 Boundary Conditions 18

3.4 Formulation of k-ω Turbulence Models 18

3.4.1 The k-ω-TNT Turbulence Model 19

3.4.2 The k-ω-SST Model 21

3.4.2.1 The k-ω-SST-2003 Model 21

3.4.2.2 k-ω-SST-SAS Model 23

3.5 Hybrid RANS/LES Turbulence Models 24
3.5.1 Subgrid Length Scale 25
3.5.1.1 Volume Based Subgrid Scale 25
3.5.1.2 Edge Size Subgrid Scale 25
3.5.1.3 LSQ Subgrid Scale 25
3.5.1.4 Vorticity Aligned Subgrid Scale 26
3.5.1.5 Shear Layer Adapted Subgrid Scale 26
3.5.2 X-LES Hybrid RANS/LES Turbulence Model 26
3.5.3 DDES/IDDES Hybrid RANS/LES Turbulence Model 28
3.5.3.1 The k-ω-SST-DDES Models 29
3.5.3.2 The k-ω-SST-IDDES Models 29
3.5.4 Partially Averaged Navier Stokes Turbulence Model 31
3.6 Boundary Conditions for the k-ω and Hybrid Turbulence Models . . . 32
3.7 Evaluation of LEVM Reynolds Stress Tensor 33
3.8 Turbulence-Mean-flow Coupling 34
3.9 SSG/LRR-ω Model 34
3.10 Turbulence Model and Scheme Selection 35

4 Computational Methods 36
4.1 Spatial Discretization 36
4.1.1 Geometric Conversation Law 37
4.2 Flux Approximation Schemes 37
4.2.1 HLLC 38
4.2.2 AUSM 40
4.2.2.1 AUSM$^+$-up 41
4.2.2.2 Extension of the AUSM$^+$-up to Unsteady Low Mach Number Flows: AUSM-JCAM13 43
4.2.2.3 AUSMDV 44
4.2.3 LDFSS 46
4.2.3.1 Extension of the LDFSS to Unsteady Low Mach Number Flows: LDFSS-JCAM13 48

4.2.4 The AUFSR and AUFSR+ Schemes 49
4.2.4.1 Low Mach Number Extension, the AUFSR+ Scheme 51
4.2.5 Passive Scalar Approach 52
4.2.6 Choosing the Flux Approximation 53
4.2.7 Monotone Upstream-Centered Scheme for Conservation Laws (MUSCL) 53
4.2.8 High Order Flux Approximation Schemes 55
4.2.8.1 Monotonicity-Preserving (MP) Reconstruction . . . 55
4.2.8.2 Weighted Essentially Non-Oscillatory (WENO) Reconstruction Scheme 56
4.3 Hybrid Convection Flux Scheme 57
4.3.1 Hybrid Convection Flux Blending Functions 58
4.3.1.1 Detached Eddy Simulation Based Blending Function 59
4.3.1.2 Ducros Blending Function 59
4.3.1.3 Larsson Blending Function 60
4.3.1.4 Travin Blending Function 60
4.3.1.5 Flow Simulation Methodology Based Blending Function 61
4.4 Diffusive Flux Vector Discretization 62
4.4.1 Full-Viscous Approximation of Face Gradients 62
4.4.2 Thin-Layer Approximation of Face Gradients 62
4.5 Special Jacobians for Implicit Time Marching 63
4.5.1 Steger-Warming Flux Vector Splitting Jacobians 63
4.6 Time Marching Schemes 64
4.6.1 The Time Step 65
4.6.2 Explicit schemes 66
4.6.2.1 Explicit Euler Scheme 66
4.6.2.2 Runge-Kutta Schemes 66
4.6.3 Implicit Time Marching Formulation 67
4.6.3.1 Point Gauss-Seidel 67
4.6.3.2 Alternating Line Gauss-Seidel 68
4.6.3.3 ADI and DDADI Schemes 68

4.6.4 B2 Two Stage Time Integration Scheme 68
4.6.5 Selection of Time Marching Scheme 69
4.7 Dual Time Step Formulation 69
4.8 Preconditioned Time Marching Formulation 70
4.9 Multigrid Convergence Acceleration Method 73
4.9.1 Full Approximation Storage (FAS) Multigrid 74
4.9.1.1 Fine-Grid Pre-Relaxation 76
4.9.1.2 Restriction to a Coarser Grid 76
4.9.1.3 Coarse-Grid Relaxations 76
4.9.1.4 Coarse-Grid Correction 77
4.9.1.5 Fine-Grid Post-Relaxation 77
4.9.2 Full Multi Grid (FMG) Initialization 77

5 Boundary Conditions 79
5.1 Introduction 79
5.2 Wall Boundary Conditions 79
5.2.1 Impermeable Wall Conditions 79
5.2.2 No-Slip Condition 80
5.2.3 Adiabatic Wall 80
5.2.4 Isothermal Wall 80
5.2.5 Prescribed Heat Flux 81
5.2.6 Radiative Equilibrium 81
5.3 Far Field Conditions 81
5.3.1 Turkel Type Conditions 81
5.3.1.1 Turkel Inlet 82
5.3.1.2 Turkel Outlet 82
5.3.2 Riemann Type Conditions 82
5.3.2.1 Riemann Inlet 82
5.3.2.2 Riemann Outlet 83
5.3.3 Fixed (Supersonic Inlet) 83
5.3.4 Extrapolation (Supersonic Outlet) 83

5.3.5 Inout . . . 83
5.3.6 Stagnation Inlet . . . 84
5.4 Axis Boundary Conditions . . . 84
5.5 Symmetry Boundary Conditions . . . 84
5.6 Intake Conditions . . . 85
5.7 E-Mesh Boundary Conditions . . . 85
5.8 Periodic Boundary Conditions . . . 85
5.9 Actuator Disk Model . . . 85

6 Computational Mesh **86**
6.1 Single Mesh Topology . . . 86
6.2 Definition of a Body . . . 88
6.3 Chimera Module . . . 88
6.3.1 Automatic Overlap Minimization . . . 89
6.3.1.1 Recommended Practice . . . 90
6.3.2 The Chimera Procedure . . . 90
6.4 Envelopes . . . 96
6.5 Phantoms . . . 97

7 Six Degrees of Freedom Simulation **100**
7.1 Translational Equation of Motion . . . 101
7.2 Rotational Equation of Motion . . . 101
7.2.1 Quaternions . . . 102
7.3 Integration . . . 103
7.4 Motion with More than Six Degrees of Freedom . . . 103

8 Static and Dynamic Aeroelasticity **104**
8.1 Static Aeroelasticity . . . 105
8.2 Equations of Motion for Uncoupled Aeroelastic Motion . . . 106
8.2.1 Numerical Scheme . . . 106
8.2.2 Solution Method . . . 107
8.3 Mesh Deformations . . . 107

9 Parallelization **108**

9.1 Section By Section Parallelism . . . 108

9.1.1 Main Section . . . 108

9.1.2 Boundary Conditions . . . 110

9.1.3 I/O Operations . . . 110

9.1.4 Chimera Interpolation Seek . . . 111

10 Input File, Run Preparation, and Execution **112**

10.1 Preface . . . 112

10.2 Scripting Language . . . 114

10.3 Auxiliary Options . . . 115

10.4 Simulation Options . . . 116

10.4.1 Automatic Chimera Module . . . 128

10.4.2 Simulation Chimera Options . . . 137

10.5 Geometry Options . . . 144

10.5.1 Formation Options . . . 144

10.5.2 Body Options . . . 155

10.5.3 Zone Options . . . 165

10.5.4 Zonal Chimera Options . . . 190

10.5.4.1 Manual Hole Cutting . . . 196

10.5.4.2 Force Overlap . . . 203

10.5.4.3 Phantom Options . . . 207

10.5.5 Shape Options . . . 215

10.5.6 Disturb Options . . . 223

10.5.7 Vertex Options . . . 231

10.5.8 Force Options . . . 232

10.5.9 Special Summation Options . . . 235

10.6 Solver Options . . . 238

10.6.1 Equation Sets . . . 238

10.6.2 Schemes and Order . . . 254

10.6.3 Time Step . . . 270

10.6.4 Reference Quantities . . . 273
10.6.5 Simulation Control . . . 277
10.6.6 Flow Conditions . . . 286
10.6.7 Gas Constants . . . 290
10.7 Multi Grid . . . 292
10.8 Boundary Conditions Options . . . 297
10.8.1 Boundary Conditions General Flags . . . 297
10.9 Movement Options . . . 315
10.9.1 Motion Simulation . . . 315
10.9.2 Aeroelasticity . . . 319
10.10 Adaptive Mesh Refinement . . . 325
10.10.1 AMR Sensor Implementation . . . 325
10.11 Output Options . . . 330
10.11.1 Log File . . . 330
10.11.2 Post-processing Options . . . 332
10.12 Parallelization Options . . . 335
10.13 Run-time Options . . . 336

11 User Defined "N" Degrees of Freedom Motion 337
11.1 Library Interface . . . 337
11.2 Automatic Tool . . . 342
11.2.1 Global Variables . . . 342
11.2.2 Split 'main' Function . . . 343
11.2.3 Add Save and Load Functions . . . 346
11.2.4 Add 'From' Hooks . . . 350
11.2.5 Add 'To' Hooks . . . 351
11.2.6 Sample Flow Primitives . . . 353
11.3 Compilation of "N" DOF Library . . . 356

A DOF Options 357

B Log Options 364

C Function File Types **385**

D NERR and NB Legends **394**

Bibliography **401**

List of Figures

4.1 Wave structure of the HLLC approximate Riemann solver 39
4.2 A schematic description of the V-Cycle for a four-level hierarchy. . . . 75
4.3 Schematic description of Full-Multigrid (FMG) initialization for a four-level hierarchy. 78

6.1 O type Mesh with an axis 87
6.2 E type Mesh 87
6.3 The surface.cut directive controls the hole cutting of the cell that is marked by a dashed green curve 92
6.4 Outcome of surface.cut no 92
6.5 Outcome of surface.cut far 93
6.6 Outcome of surface.cut near 93
6.7 Outcome of surface.cut tight 94
6.8 Holes cut by envelopes 96
6.9 Phantom; region 97
6.10 Phantom; ellipse 98
6.11 Phantom; ellipsoid 98
6.12 Phantom; cylinder 99
6.13 Phantom; fan 99

9.1 Single grid, lists of "live" cells for ξ, η, and ζ directions 109
9.2 Ideal division of a 2D grid into groups of lines for two processes . . . 109

10.1 Example of collar grid generation 167
10.2 Example of force overlap 203

List of Tables

3.1 k-ω-SST-2003 coefficients . . . 22

10.1 Auxiliary options . . . 115
10.2 Simulation prob option . . . 116
10.3 Simulation echo option . . . 116
10.4 Simulation detect option . . . 117
10.5 Simulation detect.tolerance option . . . 117
10.6 Simulation wise.tolerance option . . . 118
10.7 Simulation load option . . . 118
10.8 Simulation save option . . . 119
10.9 Simulation backup option . . . 119
10.10 Simulation store velocity averages option . . . 120
10.11 Simulation store resolved stress averages option . . . 120
10.12 Simulation store turbulent stress averages option . . . 120
10.13 Simulation store statistical averages option . . . 121
10.14 Simulation wall.distance option . . . 121
10.15 Simulation dualtime option . . . 122
10.16 Simulation unsteady option . . . 123
10.17 Simulation format option . . . 124
10.18 Simulation bc option . . . 125
10.19 Simulation turbulent bc option . . . 126
10.20 Simulation Sigma bc option . . . 127
10.21 Simulation turbulent.wall.distance bc option . . . 127
10.22 Simulation Chimera auto/manual option . . . 128
10.23 Simulation automatic Chimera option . . . 129

10.24 Simulation Chimera suitability option 130
10.25 Simulation Chimera auto.wall.distance option 131
10.26 Simulation Chimera global logical interpolation options 132
10.27 Simulation Chimera global force overlap options 133
10.28 Simulation Chimera enclave option 134
10.29 Simulation Chimera iterations option 135
10.30 Simulation Chimera algorithm option 135
10.31 Simulation Chimera leaf option 136
10.32 Simulation Chimera opt.dsf option 136
10.33 Simulation chimera.tolerance option 137
10.34 Simulation chimera options 138
10.35 Simulation Chimera interpolation options 139
10.36 Simulation Chimera auto.logical.interpolation option 139
10.37 Simulation Chimera complementary interpolation options 140
10.38 Simulation Chimera surface.cut option 141
10.39 Simulation chimera save/load options 142
10.40 Simulation save.reconstruct option 143
10.41 Simulation load.reconstruct option 143
10.42 Geometry formation new option 144
10.43 Geometry formation refarea option 144
10.44 Geometry formation reflen option 145
10.45 Geometry formation refvel option 145
10.46 Geometry formation refrho option 146
10.47 Geometry formation refpoint option 146
10.48 Geometry formation clear option 146
10.49 Geometry formation offset option 147
10.50 Geometry formation rotate options 147
10.51 Geometry formation scale option 148
10.52 Geometry formation matrix option 149
10.53 Geometry formation dilute options 150
10.54 Geometry formation redistribute options (equal spacing) 151

10.55 Geometry formation redistribute tanh options (single sided) 152
10.56 Geometry formation redistribute tanh options (two sided) 153
10.57 Geometry formation cnv options 154
10.58 Geometry body new option 155
10.59 Geometry body clear option 156
10.60 Geometry body set option 156
10.61 Geometry body offset option 157
10.62 Geometry body rotate options 157
10.63 Geometry body scale option 158
10.64 Geometry body matrix option 159
10.65 Geometry body dilute options 160
10.66 Geometry body redistribute options (equal spacing) 161
10.67 Geometry body redistribute tanh options (single sided) 162
10.68 Geometry body redistribute tanh options (two sided) 163
10.69 Geometry body cnv options 164
10.70 Geometry zone newp3d option 165
10.71 Geometry zone collar option 166
10.72 Geometry zone set option 167
10.73 Geometry zone crop option 168
10.74 Geometry zone dilute options 168
10.75 Geometry zonal redistribute options (equal spacing) 169
10.76 Geometry zonal redistribute tanh options (single sided) 170
10.77 Geometry zonal redistribute tanh options (two sided) 171
10.78 Geometry zone cnv options 172
10.79 Geometry zone offset option 172
10.80 Geometry zone rotate options 173
10.81 Geometry zone scale option 173
10.82 Geometry zone matrix option 174
10.83 Geometry zone detect option 175
10.84 Geometry zone detect.tolerance option 175
10.85 Geometry zone chimera.tolerance option 176

10.86 Geometry zone wise.tolerance option 176
10.87 Geometry zone bc option 177
10.88 Geometry zone order option 178
10.89 Geometry zone order option (MP-4) 179
10.90 Geometry zone order option (MP-5) 180
10.91 Geometry zone order option (WENO-5) 181
10.92 Geometry zone tu option 182
10.93 Geometry zone tnt.xles option 183
10.94 Geometry zone sst.ddes option 184
10.95 Geometry zone sst.iddes option 185
10.96 Geometry zone sst.pans option 186
10.97 Geometry zone hybrid.central.upwind option 187
10.98 Geometry zone cfl.factor option 188
10.99 Geometry zone board option 188
10.100 Geometry zone multigrid option 189
10.101 Geometry Chimera do.not.cut.holes.in option 190
10.102 Geometry Chimera cut.holes.in option 191
10.103 Geometry Chimera do.not.cut.me.with option 191
10.104 Geometry Chimera cut.me.with option 192
10.105 Geometry zone Chimera option 193
10.106 Geometry Chimera interpolation options 194
10.107 Geometry Chimera complementary interpolation options 195
10.108 Geometry Chimera envelope option 196
10.109 Geometry Chimera surface.cut option 197
10.110 Geometry Chimera surface.cut override option 198
10.111 Geometry Chimera kill/revive option 199
10.112 Geometry Chimera interpolation.priority option 199
10.113 Geometry Chimera do.not.interpolate.from option 200
10.114 Geometry Chimera auto/manual option 200
10.115 Geometry zonal automatic Chimera option 201
10.116 Geometry Chimera suitability option 201

10.117 Geometry Chimera enclave option 202
10.118 Geometry Chimera force.overlap option 204
10.119 Geometry Chimera force.project option 205
10.120 Geometry Chimera forcing option 206
10.121 Geometry phantom entity.box option 207
10.122 Geometry phantom region option 208
10.123 Geometry phantom ellipse option 209
10.124 Geometry phantom ellipsoid option 210
10.125 Geometry phantom cylinder option 211
10.126 Geometry phantom fan.cylinder option 212
10.127 Geometry phantom sphere option 213
10.128 Geometry phantom project option 214
10.129 Geometry phantom revive option 214
10.130 Geometry shape init option 215
10.131 Geometry shape entity.box option 216
10.132 Geometry shape region option 217
10.133 Geometry shape ellipse option 218
10.134 Geometry shape ellipsoid option 219
10.135 Geometry shape cylinder option 220
10.136 Geometry shape fan.cylinder option 221
10.137 Geometry shape project option 222
10.138 Geometry disturb import option 223
10.139 Geometry disturb freestream option 224
10.140 Geometry disturb uniform option 225
10.141 Geometry disturb shift option 226
10.142 Geometry disturb rotate option 227
10.143 Geometry disturb manipulation based on the X coordinate option . 228
10.144 Geometry disturb manipulation based on the Y coordinate option . 229
10.145 Geometry disturb manipulation based on the Z coordinate option . 230
10.146 Geometry vertex $vertex option 231
10.147 Geometry force impermeable/noslip option 233

10.148 Geometry force back.pressure option 234
10.149 Geometry mass flow summation option 235
10.150 Geometry heat flux summation option 236
10.151 Geometry momentum summation option 237
10.152 Solver equationset option 238
10.153 Solver model.tu SA option 239
10.154 Solver model.tu TNT option 240
10.155 Solver model.tu CES option 241
10.156 Solver model.tu X-LES option 242
10.157 Solver model.tu SST option 243
10.158 Solver model.tu DDES option 244
10.159 Solver model.tu IDDES option 245
10.160 Solver model.tu PANS option 246
10.161 Solver model.tu ADHRL option 247
10.162 Solver model.tu RSM option 248
10.163 Solver model.tu SST.gammaRt option 249
10.164 Solver model.tu RSM.gammaRt option 250
10.165 Solver pans.reset.averages option 251
10.166 Solver realize.rsm option 251
10.167 Solver realize.rsm.chimera.old option 252
10.168 Solver trb.ambient.sustain option 252
10.169 Solver sigma option 253
10.170 Solver ausm.kp/ausm.ku option 253
10.171 Solver scheme option 254
10.172 Solver scheme.tu option 255
10.173 Solver B2 option 255
10.174 Solver precondition option 256
10.175 Solver order option 256
10.176 Solver order option (MP-4) 257
10.177 Solver mp4.epsilon option 257
10.178 Solver order option (MP-5) 258

10.179 Solver mp5.epsilon option 259
10.180 Solver order option (WENO-5) 259
10.181 Solver weno5.epsilon option 260
10.182 Solver shock.sensor option 260
10.183 Solver no.dt.sweep option 261
10.184 Solver limiter option 262
10.185 Solver hybrid.central.upwind option 263
10.186 Solver hybrid.shield option 264
10.187 Solver convrhs option 265
10.188 Solver dx.div.dt.coef option 266
10.189 Solver jcam13.mach.correction option 266
10.190 Solver convjacs option 267
10.191 Solver convjacs.tu option 268
10.192 Solver visgrad option 268
10.193 Solver visgrad.tu option 269
10.194 Solver CFL option 270
10.195 Solver CFL.TU option 271
10.196 Solver dualtime.blend.alpha option 272
10.197 Solver reflong option 273
10.198 Solver refarea option 273
10.199 Solver reflen option 274
10.200 Solver refpoint option 274
10.201 Solver refvel option 275
10.202 Solver refvel.trb option 275
10.203 Solver refmach option 276
10.204 Solver iters option 277
10.205 Solver wall.slowstart option 277
10.206 Solver dq.factor option 278
10.207 Solver SOR option 279
10.208 Solver sweep option 280
10.209 Solver convstop option 281

10.210 Solver axisymmetric option 282
10.211 Solver mode option 283
10.212 Solver exact wall distance option 284
10.213 Solver freeze wall distance option 284
10.214 Solver axes.impermeable option 285
10.215 Solver force.freestream.pressure option 285
10.216 Solver vel/mach options 286
10.217 Solver aoa option 286
10.218 Solver beta option 287
10.219 Solver u option 287
10.220 Solver v option 287
10.221 Solver w option 288
10.222 Solver T option 288
10.223 Solver p option 288
10.224 Solver ti option 289
10.225 Solver mt option 289
10.226 Solver gas constants option 290
10.227 Solver Sutherland's constants option 291
10.228 Solver multigrid option 293
10.229 Solver mg multigrid.init option 294
10.230 Solver mg defect factor option 295
10.231 Solver mg DQF factor option 296
10.232 BC periodic option 298
10.233 BC axis option 299
10.234 BC symmetry option 300
10.235 BC emesh option 301
10.236 BC impermeable/noslip option 305
10.237 BC inlet/outlet option 306
10.238 BC massflux.in option 307
10.239 BC massflux.out option 308
10.240 BC extrap option 309

10.241 BC fixed option 312
10.242 BC intake option 313
10.243 BC disk.model option 314
10.244 DOF reference.frame option 315
10.245 DOF time span options 316
10.246 Aeroelasticity spline option 320
10.247 Aeroelasticity surface options 321
10.248 Aeroelasticity static.deformation option 322
10.249 Aeroelasticity dynamic.deformation option 323
10.250 Dynamic aeroelasticity integration methods 324
10.251 AMR option 326
10.252 AMR refinement types 327
10.253 AMR sensors 328
10.254 AMR references 329
10.255 AMR p3d options 329
10.256 Log options 331
10.257 Post-processing options 334
10.258 Parallelization board option 335
10.259 Run-time options 336

A.1 DOF options 363

B.1 Log options 384

C.1 Function file types 393

D.1 ERR file legend 394
D.2 NB file legend 395

Listings

10.1 Example of a simple EZAir input file 113
11.1 EZAir Input File, DOF Interface 338
11.2 Requirement - Initialization Interface 338
11.3 Requirement - Integration Interface 338
11.4 Requirement - End Interface 338
11.5 Requirement - Save/Load Interface 339
11.6 Requirement - Vertex Interface 340
11.7 EZAir Input File, DOF Interface 340
11.8 Aerodynamic Forces From EZAir to the Library 341
11.9 EZAir Input File, DOF Interface 341
11.10 Movement FromLibrary to EZAir 341
11.11 Library Global Variable, HingedAirfoils Sample 342
11.12 Library Init Function, HingedAirfoils Sample 343
11.13 Library Integrate Function, HingedAirfoils Sample 346
11.14 Library End Function, HingedAirfoils Sample 346
11.15 Library Save, HingedAirfoils Sample 346
11.16 Library Load 348
11.17 EZAir Aerodynamic Forces To HingedAirfoils Library 350
11.18 HingedAirfoils Library Movement Parameter to EZAir 351
11.19 HingedAirfoils Library Vertex Allocation 354
11.20 HingedAirfoils Library Vertex Callback 354
11.21 Compiling The Library 356

Abstract

This manual describes the algorithms, methods, and input and output files of the EZAir suite of codes. In addition, the manual serves as the user's manual of the codes. The current version of the suite contains advanced Chimera capabilities, a semi-automatic collar grid generator module , a 6 degrees of freedom that contains relative motion module, an "n" degrees of freedom motion simulation interface, an aeroelasticity module, and a spline suite to support the aeroelasticity module. The flow solver contains various flux approximation methods, *e.g.*, the HLLC approximate Riemann solver, the hybrid advection upstream splitting method, AUSM^{+}-up, the hybrid artificial upstream flux splitting AUFSR scheme, and the all-speed AUFSR+ scheme. It has explicit time marching, with or without Runge-Kutta steps, and implicit time marching, with or without dual-time-stepping. The flow solver has a few linear RANS turbulence models, the algebraic Baldwin-Lomax, the Spalart Allmaras turbulence model, the $k-\omega$-TNT turbulence model, and the $k-\omega$-SST turbulence model, The $k-\omega$-TNT model contains flags that turn it into a hybrid model, namely the $X-LES$ model. The Chimera suite contains phantom hole cutting and fail safe mechanisms for interpolations. The aeroelasticity module is based on the Karpel-Raveh modal approach. The code is fully parallel.

Chapter 1

Introduction

The EZAir structured solvers suite is a multi-zone, multi-block Euler/Navier-Stokes suite of multi-physics flow solvers. The EZAir suite of codes replaces ISCFDC first generation solver and like their predecessor, the EZAir solvers are capable of simulating complex, time-accurate flows about dynamically deforming geometries. This includes relative motion between surfaces as well as deformations caused due to aeroelastic effects.

The EZAir solvers are based on a common framework that includes advanced structured grid identification and manipulation functions, robust temporal and spatial discretization schemes, and a range of turbulence models. The solvers differ mainly in the available physical models, and accompanying numerical schemes. This manual covers the following codes:

1. EZAir - A highly-efficient flow solver designed for the simulations of ideal-gas (air) flows.

The codes handle complex geometries using patched grids or the Chimera overset grid topology [1]. The codes automatically handle various grid topologies such as C-C, C-H, and C-O grid topologies. When the grid topology is identified, the appropriate boundary conditions are set. To provide higher flexibility, the user may override the boundary conditions using the input file. The code is written using the C language. The program is parallelized using MPI and OpenMP.

The report is arranged in the following manner: Chapter 2 contains a description of the available physical models, Chapter 3 describes the turbulence models that are used in the code. Chapter 4 is dedicated to a detailed description of the computational methods. Chapter 5 briefly describes the boundary conditions while Chapter 6 describes the Chimera module. Chapter 7 contains a brief description of the six degrees of freedom motion simulation capabilities. Chapter 8 contains a description of the aeroelastic module. Chapter 9 describes the parallelization of the code. Chapter 10 contains a detailed description of the input file syntax, and actually serves as the code's reference manual. Chapter 11 entails the "n" degrees of freedom motion simulation module. Together, Chapters 2-9 comprise the theoretical part of the manual while the remaining chapters and appendices comprise the user's and reference manual.

Chapter 2

Physical Models

2.1 Introduction

Computer simulations are generally based upon the numerical solution of the model equations in a discretized mode. The accuracy of the computations depends mainly on the physical modeling, the numerical algorithm, and the quality of the computational mesh. This chapter contains a description of the physical models that are available in the solvers covered by this manual.

2.2 Single-component Perfect Gas (SPG)

2.2.1 Governing Equations

The equations governing single-component, perfect gas fluid flow are derived from the laws of conservation of mass, momentum, and total energy. The set of five partial differential equations is known as the Navier-Stokes equations and can be represented in a conservation-law form that is convenient for numerical simulations, namely

$$\frac{\partial Q}{\partial t} + \frac{\partial \left(E_c - E_d\right)}{\partial x} + \frac{\partial \left(F_c - F_d\right)}{\partial y} + \frac{\partial \left(G_c - G_d\right)}{\partial z} = 0 \tag{2.1}$$

where Q is the vector of conserved mass, momentum, and energy

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix} \tag{2.2}$$

where the density is denoted by ρ, the Cartesian velocity vector components are denoted by u, v, and w, and E denotes the total (internal and kinetic) energy of the gas. The time is denoted by t and x, y, and z denote the Cartesian coordinates. The inviscid flux vectors, E_c, F_c, and G_c, are

$$E_c = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u\,(E+p) \end{bmatrix}, \quad F_c = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v\,(E+p) \end{bmatrix}, \quad G_c = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w\,(E+p) \end{bmatrix} \tag{2.3}$$

and the viscous flux vectors, E_d, F_d, and G_d, are

$$E_d = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ \beta_x \end{bmatrix}, \quad F_d = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ \beta_y \end{bmatrix}, \quad G_d = \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ \beta_z \end{bmatrix} \tag{2.4}$$

where

$$
\begin{aligned}
\tau_{xx} &= \lambda\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+2\mu\frac{\partial u}{\partial x}\\
\tau_{yy} &= \lambda\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+2\mu\frac{\partial v}{\partial y}\\
\tau_{zz} &= \lambda\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+2\mu\frac{\partial w}{\partial z}\\
\tau_{xy} &= \tau_{yx}=\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\\
\tau_{xz} &= \tau_{zx}=\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\\
\tau_{yz} &= \tau_{zy}=\mu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\\
\beta_x &= u\tau_{xx}+v\tau_{xy}+w\tau_{xz}+\kappa\frac{\partial T}{\partial x}\\
\beta_y &= u\tau_{yx}+v\tau_{yy}+w\tau_{yz}+\kappa\frac{\partial T}{\partial y}\\
\beta_z &= u\tau_{zx}+v\tau_{zy}+w\tau_{zz}+\kappa\frac{\partial T}{\partial z}
\end{aligned}
\qquad (2.5)
$$

where T is the temperature. Stokes hypothesis, $\lambda = -\frac{2}{3}\mu$, is typically used to further simplify Equation (2.5).

Finally, in the perfect-gas model, the source-term vector, S, may only contain contributions from the turbulence model (see Chapter 3).

2.2.2 Integral Form For Moving Grids

For a three-dimensional flow through a finite volume Ω, enclosed by the boundary surface $\partial\Omega \equiv S$ that is moving with a grid velocity $\overline{V}_g$, the integral form of the conservation equations in an inertial frame of reference is given as:

$$\frac{\partial}{\partial t}\int\limits_{\Omega} Q d\Omega + \oint\limits_{S} d\overline{S}\cdot\overline{H} = 0 \qquad (2.6)$$

where

$$\overline{H} = (E_c - E_d)\,\hat{i} + (F_c - F_d)\,\hat{j} + (G_c - G_d)\,\hat{k} \qquad (2.7)$$

In Equation (2.6) the vector of dependent variables, Q, remains as in Equation (2.2).Similarly, the viscous flux vectors, E_d, F_d, and G_d, appearing in Equation (2.7), remain as in Equations (2.4) and (2.5) (see Section 2.2.1). In contrast, the inviscid flux vectors take a form that reflects the motion of the grid as follows:

$$
\begin{aligned}
E_c &= \begin{bmatrix} \rho\left(u-u_g\right) \\ \rho u\left(u-u_g\right)+p \\ \rho v\left(u-u_g\right) \\ \rho w\left(u-u_g\right) \\ (E+p)\left(u-u_g\right)+u_g p \end{bmatrix} \\
F_c &= \begin{bmatrix} \rho\left(v-v_g\right) \\ \rho u\left(v-v_g\right) \\ \rho v\left(v-v_g\right)+p \\ \rho w\left(v-v_g\right) \\ (E+p)\left(v-v_g\right)+v_g p \end{bmatrix} \\
G_c &= \begin{bmatrix} \rho\left(w-w_g\right) \\ \rho u\left(w-w_g\right) \\ \rho v\left(w-w_g\right) \\ \rho w\left(w-w_g\right)+p \\ (E+p)\left(w-w_g\right)+w_g p \end{bmatrix}
\end{aligned}
\tag{2.8}
$$

where u_g, v_g, and w_g are the Cartesian components of the grid velocity vector $\overline{V}_g$.

2.2.3 Thermodynamic Properties

To close the system of fluid dynamics equations, it is necessary to establish relations between the thermodynamics variables, p, ρ, T, and the internal energy, e_I. Assuming a perfect gas, the pressure and temperature may be obtained from the following equation of state:

$$p = \rho R T \tag{2.9}$$

where R is the gas constant ($R = 287.22$ for air). By assuming further that air is a calorically perfect gas (and hence the specific heats C_p and C_v are constant), the equation of state takes the form:

$$p = \rho\,(\gamma - 1)\,e_I \tag{2.10}$$

where e_I is the internal energy of the gas, and γ is the (constant) ratio of specific heats (c_p/c_v). In terms of the flow variables, the pressure and temperature are calculated using:

$$\begin{aligned} p &= (\gamma - 1)\left[E - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right] \\ T &= \frac{\gamma - 1}{R}\left[e - \frac{1}{2}\left(u^2 + v^2 + w^2\right)\right] \end{aligned} \tag{2.11}$$

where $e = \frac{E}{\rho}$ is the specific total energy.
See Section 10.6.7, Table 10.226 for means to set R and γ, thus selecting a different perfect gas than air.

2.2.4 Transport Properties

In addition to the equation of state, it is also necessary to establish relations for the coefficients of viscosity, μ, and thermal conductivity, κ. In the single-component perfect gas (SPG) model, the Sutherland formulae are exclusively used to evaluate these coefficients as follows:

$$\begin{aligned} \mu &= 1.458 \times 10^{-6}\frac{T^{\frac{3}{2}}}{T + 110.3} \\ \kappa &= 2.495 \times 10^{-3}\frac{T^{\frac{3}{2}}}{T + 194} \end{aligned} \tag{2.12}$$

The relations for both μ and κ in Equation 2.12 have the form:

$$f = C_1\frac{T^{\frac{3}{2}}}{T + C_2} \tag{2.13}$$

See Section 10.6.7, Table 10.227 for means to set C_1 and C_2 so that a different perfect gas than air can be selected.

2.3 Selection of Physical Model

The physical model used in the simulation may be selected by the user using the “-solve” directive along with the “equationset” parameter. Table 10.152 contains instructions and limitations pertinent to the physical model selection.

Chapter 3

Turbulence Models

The unsteady Navier-Stokes equations are generally considered to govern turbulent flows in the continuum flow regime. However, turbulent flow cannot be numerically simulated as easily as laminar flow. To resolve a turbulent flow by direct numerical simulation (DNS) requires that all relevant length scales be properly resolved. Such requirements place great demands on the computer resources, a fact that renders the possibility of conducting DNS analysis about complete aircraft configurations infeasible.

A practical approach to simulating turbulent flows is to solve the time-averaged Navier-Stokes equations. These equations are know as the "Reynolds averaged Navier-Stokes" (RANS) equations. The averaging of the equations of motion gives rise to new terms that are called the Reynolds stresses. To solve the averaged equations, the Reynolds stress tensor must be related to the flow variables through turbulence models. The models are used to "close" the system through an additional set of assumptions. The models are classified based on the number of additional partial differential equations that must be solved. The EZAir suite of codes currently provides the choice between zero equations, *i.e.*, an algebraic model, a one-equation model, two two-equation models, and a Reynolds stress model, having seven equations. One of the two-equation models also provides the means to conduct detached eddy simulations (DES) using a hybrid model. The next section is dedicated to the algebraic (zero-equation) Baldwin-Lomax turbulence model while the remaining sections of this chapter describe RANS turbulence models that are based on partial differential equations.

3.1 Baldwin-Lomax Turbulence Model

The Baldwin-Lomax algebraic turbulence model was developed originally by Baldwin and Lomax [2] for a flat-plate boundary layer, based on the two-layer model reported by Cebeci *et al* [3]. On the basis of the algebraic model, the coefficients of viscosity μ and thermal conductivity κ are replaced by the relations

$$\begin{aligned} \mu &= \mu_l + \mu_t \\ \kappa &= \kappa_l + \frac{c_p \mu_t}{Pr_t} \end{aligned} \tag{3.1}$$

The turbulent eddy viscosity coefficient μ_t is computed using the Baldwin-Lomax algebraic eddy-viscosity model [2] as follows,

$$\mu_t = \begin{cases} (\mu_t)_{inner}\,, & y \leq y_c \\ (\mu_t)_{outer}\,, & y > y_c \end{cases} \tag{3.2}$$

where y is the normal distance from the body surface and y_c is the smallest value of y for which values from the formula for the inner region are equal to values from that for the outer region. The formula for μ_t in the inner region is given by the Prandtl-Van Driest formulation,

$$(\mu_t)_{inner} = \rho l^2 \left|\omega\right| \tag{3.3}$$

where l, the length scale, is

$$l = ky \left[1 - e^{-\left(y^+/A^+\right)}\right] \tag{3.4}$$

The quantity $|\omega|$ is the magnitude of the local vorticity vector, A^+ is a constant, and y^+ is the normal law-of-the-wall coordinate defined by

$$y^+ = \frac{\sqrt{\rho_w \tau_w}}{\mu_w} y \tag{3.5}$$

where ρ_w is the mass density at the body surface, τ_w is the wall shear stress, and μ_w is the viscosity coefficient at the wall. In the outer region, for attached boundary layers the turbulent viscosity coefficient is given by

$$(\mu_t)_{outer} = K C_{cp} \rho F_{wake} F_{kleb}(y) \tag{3.6}$$

where K (the Clauser constant) and C_{cp} are constants, and

$$F_{wake} = y_{max} F_{max} \tag{3.7}$$

In Equation (3.7), F_{max} is the maximum value of $F(y)$

$$F(y) = |\omega| \, y \left[1 - e^{-\left(y^+/A^+\right)}\right] \tag{3.8}$$

and y_{max} is the value of y where F_{max} is obtained. The function $F_{kleb}(y)$ is the Klebanoff intermittency factor given by

$$F_{kleb}(y) = \left[1 + 5.5 \left(\frac{C_{kleb} y}{y_{max}}\right)^6\right]^{-1} \tag{3.9}$$

The constants appearing in Equations (3.1-3.9) were determined by Baldwin and Lomax [2]

$$\begin{array}{lcrlcr} Pr_t & = & 0.9 & k & = & 0.4 \\ K & = & 0.0168 & A^+ & = & 26 \\ C_{cp} & = & 1.6 & C_{kleb} & = & 0.3 \end{array}$$

The Baldwin Lomax turbulence model has been added to the EZAir suite of codes only for reasons of compatibility. It is mainly used under the zonal turbulence paradigm. It is invoked by using “laminar” as the equation set (see Table 10.152) and by setting certain zones to solve inviscid flow and other zones to solve turbulent flows using the Baldwin-Lomax model. For details contact the developer.

3.2 RANS Turbulence Model Equations

The EZAir suite of codes treats the mean flow and turbulence models equations in a unified manner. To this end, the Navier-Stokes equation set is extended to include the turbulence model equations. Consequently, the discretization of the various fluxes can be conducted in the same manner.

The equations governing turbulent flows are obtained by Favre-averaging the Navier-Stokes equations and by modeling the Reynolds stress tensor. The unknown averaged Reynolds stress tensor is modeled either using the Boussinesq assumption via a linear eddy-viscosity model or by directly solving a transport equation for each of the Reynolds stress components via a second moment closure. The general form of the resulting Navier-Stokes equations and the turbulence model equations has the form (for simplicity of the representation, with no loss of generality, the formulation under the perfect gas physical model assumption is brought herein; it can be easily extended to any physical model):

$$\frac{\partial Q}{\partial t} + \frac{\partial \left(E_c - E_d\right)}{\partial x} + \frac{\partial \left(F_c - F_d\right)}{\partial y} + \frac{\partial \left(G_c - G_d\right)}{\partial z} = S \tag{3.10}$$

where S is the source term associated with the turbulence model only (once again, under the perfect gas physical model assumption). Hence, for turbulent flow simulations, Equation (3.10) replaces Equation (2.1).

In what follows, the symbol ($\bar{}$) indicates non-weighted averaging, the symbol ($\tilde{}$) signifies mass weighted Favre averaging, and the symbol ($''$) denotes Favre fluctuations. Depending on whether the turbulence model has one, two, or m equations, the vector of dependent variables, Q, and the vector of source terms S now take the form:

$$Q = \begin{bmatrix} \bar{\rho} \\ \bar{\rho}\tilde{u} \\ \bar{\rho}\tilde{v} \\ \bar{\rho}\tilde{w} \\ \tilde{E} \\ \bar{\rho}q_1 \\ \dots \\ \bar{\rho}q_m \end{bmatrix}, \qquad S = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ s1 \\ \dots \\ s_m \end{bmatrix} \tag{3.11}$$

where q is the vector of turbulence model dependent variables and s, is the source terms vector. Note that the source terms differ from model to model. The vectors E_c, F_c, and G_c usually take the form:

$$E_c = \begin{bmatrix} \bar{\rho}\tilde{u} \\ \bar{\rho}\tilde{u}^2 + \bar{p} \\ \bar{\rho}\tilde{u}\tilde{v} \\ \bar{\rho}\tilde{u}\tilde{w} \\ \tilde{u}\left(\tilde{E} + \bar{p}\right) \\ \bar{\rho}\tilde{u}q_1 \\ \dots \\ \bar{\rho}\tilde{u}q_m \end{bmatrix}, \quad F_c = \begin{bmatrix} \bar{\rho}\tilde{v} \\ \bar{\rho}\tilde{u}\tilde{v} \\ \bar{\rho}\tilde{v}^2 + \bar{p} \\ \bar{\rho}\tilde{v}\tilde{w} \\ \tilde{v}\left(\tilde{E} + \bar{p}\right) \\ \bar{\rho}\tilde{v}q_1 \\ \dots \\ \bar{\rho}\tilde{v}q_m \end{bmatrix}, \quad G_c = \begin{bmatrix} \bar{\rho}\tilde{w} \\ \bar{\rho}\tilde{u}\tilde{w} \\ \bar{\rho}\tilde{v}\tilde{w} \\ \bar{\rho}\tilde{w}^2 + \bar{p} \\ \tilde{w}\left(\tilde{E} + \bar{p}\right) \\ \bar{\rho}\tilde{w}q_1 \\ \dots \\ \bar{\rho}\tilde{w}q_m \end{bmatrix} \tag{3.12}$$

Similarly, the vectors, E_d, F_d, and G_d, usually take the form:

$$E_d = \begin{bmatrix} 0 \\ \bar{\tau}_{xx} - \overline{\rho u'' u''} \\ \bar{\tau}_{yx} - \overline{\rho v'' u''} \\ \bar{\tau}_{zx} - \overline{\rho w'' u''} \\ \bar{\beta}_x \\ e_{d_1} \\ \dots \\ e_{d_m} \end{bmatrix}, \quad F_d = \begin{bmatrix} 0 \\ \bar{\tau}_{xy} - \overline{\rho u'' v''} \\ \bar{\tau}_{yy} - \overline{\rho v'' v''} \\ \bar{\tau}_{zy} - \overline{\rho w'' v''} \\ \bar{\beta}_y \\ f_{d_1} \\ \dots \\ f_{d_m} \end{bmatrix}, \quad G_d = \begin{bmatrix} 0 \\ \bar{\tau}_{xz} - \overline{\rho u'' w''} \\ \bar{\tau}_{yz} - \overline{\rho v'' w''} \\ \bar{\tau}_{zz} - \overline{\rho w'' w''} \\ \bar{\beta}_z \\ g_{d_1} \\ \dots \\ g_{d_m} \end{bmatrix} \tag{3.13}$$

where the vectors e_d, f_d, and g_d differ from model to model and

$$\begin{aligned}
\bar{\beta}_x &= \tilde{u}\left(\bar{\tau}_{xx} - \overline{\rho u''u''}\right) + \tilde{v}\left(\bar{\tau}_{xy} - \overline{\rho u''v''}\right) + \tilde{w}\left(\bar{\tau}_{xz} - \overline{\rho u''w''}\right) + (\bar{\kappa} + \bar{\kappa}_t)\frac{\partial \bar{T}}{\partial x} \\
\bar{\beta}_y &= \tilde{u}\left(\bar{\tau}_{yx} - \overline{\rho v''u''}\right) + \tilde{v}\left(\bar{\tau}_{yy} - \overline{\rho v''v''}\right) + \tilde{w}\left(\bar{\tau}_{yz} - \overline{\rho v''w''}\right) + (\bar{\kappa} + \bar{\kappa}_t)\frac{\partial \bar{T}}{\partial y} \\
\bar{\beta}_z &= \tilde{u}\left(\bar{\tau}_{zx} - \overline{\rho w''u''}\right) + \tilde{v}\left(\bar{\tau}_{zy} - \overline{\rho w''v''}\right) + \tilde{w}\left(\bar{\tau}_{zz} - \overline{\rho w''w''}\right) + (\bar{\kappa} + \bar{\kappa}_t)\frac{\partial \bar{T}}{\partial z}
\end{aligned} \tag{3.14}$$

The terms $\bar{\kappa}$ and $\bar{\kappa}_t$ are the averaged molecular and turbulent heat conductivities, respectively. They are calculated using

$$\bar{\kappa} = \frac{c_p \bar{\mu}}{Pr} \tag{3.15}$$

$$\bar{\kappa}_t = \frac{c_p \bar{\mu}_t}{Pr_t} \tag{3.16}$$

where $\bar{\mu}$ denotes the average molecular viscosity, evaluated using Sutherland's law, and $\bar{\mu}_t$ denotes the turbulent viscosity. The term c_p is the specific heat capacity at constant pressure, Pr is the Prandtl number set to $Pr = 0.72$, and Pr_t is the turbulent Prandtl number set to $Pr_t = 0.9$. The average turbulent viscosity, $\bar{\mu}_t$ differs from model to model. Note: from here on the bars and tildes above all averaged flow variables are omitted and an averaged physical domain is assumed.

3.3 The Spalart Allmaras Model

The one-equation SA model was originally published in 1992 [4]. The original model, also known as the *standard* SA model (according to the TMR web site) heavily relies on calibration by references to a wide range of experimental data, including flows subjected to adverse pressure gradient. The rich body of empirical information that the model contains gives it advantages over other models when applied to 'complex' boundary layers approaching separation, and this has made the model popular for aeronautical practice. However, the *standard* SA model suffers from certain anomalies. To circumvent these anomalies, Edwards and Chandra [5] proposed a modification to the model, resulting in a far more robust model. Although the work by Edwards and

Chandra concerns the incompressible form of the model, a compressible formulation of their model is adopted herein (denoted the *SA-Edwards* model). In addition, the EZAir code offers a second variant of the SA model, based on the work by Allmaras *et al* [6] with a slight modification as proposed in the work by Rumsey *et al* [7]. The second variant is denoted as the *SA-2020* model. Both models were desgined to overcome the above mentioned anomalies.

A unified formulation of the two aforementioned variants may be written as follows:

$$\frac{\partial \rho \tilde{\nu}}{\partial t} + \frac{\partial \rho \tilde{\nu} u_k}{\partial x_k} = \frac{\partial}{\partial x_k}\left[\frac{1}{\sigma}\left(\mu + \rho\tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial x_k}\right] + P_{SA} - D_{SA} + SD_{SA} + C_{SA} \tag{3.17}$$

The model is comprised of a transport equation for the undamped eddy viscosity, $\tilde{\nu}$, where t denotes the time and x_j=[x,y,z] denote the Cartesian coordinates. The fluid density is denoted by ρ while the Cartesian velocity vector components are denoted by u_j=[u,v,w].

The production term, P_{SA}, and the destruction term, D_{SA}, are given as by:

$$P_{SA} = c_{b_1}\tilde{S}\rho\tilde{\nu} \tag{3.18}$$

and

$$D_{SA} = c_{w_1} f_w \rho \left(\frac{\tilde{\nu}}{d}\right)^2 \tag{3.19}$$

where d is the distance from the nearest wall. The near-wall damping function f_w is defined as:

$$f_w = g\left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right)^{1/6} \tag{3.20}$$

where

$$g = r + c_{w2}\left(r^6 - r\right) \tag{3.21}$$

The source diffusion term, SD_{SA} is given by:

$$SD_{SA} = \rho\frac{c_{b_2}}{\sigma}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k} \tag{3.22}$$

The definition of the term C_{SA} is addressed later on.

The turbulent viscosity, μ_t, is defined as:

$$\mu_t = \rho \tilde{\nu} f_{v1} \tag{3.23}$$

where the the damping function, f_{v1}, is given by:

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{3.24}$$

where χ is defined by (ν being the kinematic viscosity):

$$\chi \equiv \frac{\tilde{\nu}}{\nu} \tag{3.25}$$

The model constants are given as follows:

$$\begin{aligned} c_{b1} = 0.1355, \quad c_{b2} = 0.622, \quad \kappa = 0.41, \quad \sigma = 2/3 \\ c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}, \quad c_{w2} = 0.3, \quad c_{w3} = 2, \quad c_{v1} = 7.1 \end{aligned} \tag{3.26}$$

The differences between the *SA-Edwards* model and the *SA-2020* model lie in the definitions of the terms, $\tilde{S}$, r, and C_{SA}.

3.3.1 SA-Edwards Model

Edwards and Chandra [5] desgined the term $\tilde{S}$ as follows

$$\tilde{S} = \sqrt{S}\left[\left(\frac{1}{\chi}\right) + f_{v1}\right] \tag{3.27}$$

where S is the magnitude of the mean-strain rate tensor,

$$S = \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} - \frac{2}{3}\left(\frac{\partial u_k}{\partial x_k}\right)^2 \tag{3.28}$$

The parameter r was mofied from the original SA model and is given as:

$$r = \frac{\tanh\left[\dfrac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}\right]}{\tanh(1.0)} \tag{3.29}$$

The term C_{SA} is zero in the present implemenation of the *SA-Edwards* model. Thanks to the specific formulation of $\tilde{S}$ and r, the resulting model is far more robust with improved convergence characteristics.

3.3.2 SA-2020 Model

The *SA-2020* model is given in a fully consistent compressible formulation. The term $\tilde{S}$ is defined as

$$\tilde{S} = \begin{cases} \Omega_s + \bar{S} & \bar{S} \geq -c_{v_2}\Omega_s \\ \Omega_s + \dfrac{\Omega_s\left(c_{v_2}^2\Omega_s + c_{v_3}\bar{S}\right)}{\left(c_{v_3} - 2c_{v_2}\right)\Omega_s - \bar{S}} & \text{Otherwise} \end{cases}$$

where

$$\Omega_s = \sqrt{0.5\left(2s_{ij}s_{ij} + 2\omega_{ij}\omega_{ij}\right)} \tag{3.30}$$

with c_{v_2}=0.7 and c_{v_3}=0.9. The term $\bar{S}$ is defined as:

$$\bar{S} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \tag{3.31}$$

where the function f_{v2} is defined as:

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{3.32}$$

The parameter r is defined as:

$$r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\right) \tag{3.33}$$

Finally, the term C_{SA} complements the fully consistent compressible formulation of the model, namely:

$$C_{SA} = -\frac{1}{\sigma}\left(\nu + \tilde{\nu}\right)\frac{\partial \rho}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k} \tag{3.34}$$

3.3.3 Boundary Conditions

To close the solution of the model, the boundary conditions are specified. The no-slip, wall boundary condition of $\tilde{\nu}$, denoted by $\tilde{\nu}_{wall}$, is specified by:

$$\tilde{\nu}_{wall} = 0 \tag{3.35}$$

The recommended far field inlet boundary condition of $\tilde{\nu}$, denoted by $\tilde{\nu}_{\infty}$, for external flows is given by: [8]:

$$\tilde{\nu}_{\infty} = 3\nu_{\infty} \ to \ 5\nu_{\infty} \tag{3.36}$$

This range of inflow values is appropriate for fully turbulent behavior, with $\tilde{\nu}_{\infty} = 3\nu_{\infty}$ being somewhat preferable at the lower Reynolds numbers regime. This is because it results in a free stream turbulent viscosity of $\mu_t/\mu \approx 0.2$, well below 1.0. Higher values may slightly help convergence at high Reynolds numbers by smoothing out the edges of turbulent regions, but not enough to motivate giving up the simple recommendation.

3.4 Formulation of k-ω Turbulence Models

There are numerous suggestions in the literature for two-equation turbulence models, many being variations of a few baseline models. As most two equation models contain one equation for the turbulence kinetic energy, k, one important issue is the quantity chosen to represent the length scale. While ϵ is the most obvious and popular option, it is one that causes significant problems in practice, especially in near-wall flows approaching separation. In computational aerodynamics, the most popular alternative to ϵ itself is the turbulent specific dissipation rate, ω ($\epsilon \propto k\omega$). The attraction of ω-based models is rooted in the observation that it gives a superior representation of the near-wall behavior, especially in adverse pressure gradient regions.

The transport form of the compressible k-ω turbulence model contains two transport equations. A transport equation for the turbulence kinetic energy, k,

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_j k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\mu + \sigma_k \mu_t)\frac{\partial k}{\partial x_j}\right] + P_k - \beta_k \rho k \omega \tag{3.37}$$

and a second transport equation for the turbulent specific dissipation rate, ω,

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\mu + \sigma_\omega \mu_t)\frac{\partial \omega}{\partial x_j}\right] + P_\omega - \beta_\omega \rho \omega^2 + \mathcal{CD} \tag{3.38}$$

where t denotes the time and x_j=[x,y,z] denote the Cartesian coordinates. The fluid density is denoted by ρ while the Cartesian velocity vector components are denoted by u_j=[u,v,w]. Based on the Boussinesq approximation, the production term, denoted by P_k, is defined as:

$$P_k = \mathfrak{R}_{ij}\frac{\partial u_i}{\partial x_j} \tag{3.39}$$

where $\mathfrak{R}_{ij} = \overline{\rho u_i'' u_j''}$ are the Reynolds stress tensor components. The production term is therefore modeled as:

$$P_k = \left\{\mu_t \left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right] - \frac{2}{3}\rho k \delta_{ij}\right\}\frac{\partial u_i}{\partial x_j} \tag{3.40}$$

The model coefficients as well as the cross diffusion term, $\mathcal{CD}$, depend on the actual k-ω model. Apart from that, in general, most of the k-ω turbulence models have the same formulation as in Equations (3.37) and (3.38). The various models differ by the coefficients and the form of the source terms. The above formulation may also serve as the based model for advanced hybrid RANS/LES models, as will be outlined ahead.

3.4.1 The k-ω-TNT Turbulence Model

The TNT turbulence model has two clear advantages over other two-equation turbulence models: it uses a topology-free approach, and it is insensitive to the specific turbulence dissipation rate free-stream boundary condition. Following the notation set forth in Equation (3.11), the source terms pertaining to the kinetic energy equation and the specific dissipation rate equation of the k-ω-TNT turbulence model are given by:

$$S = \left\{ \begin{matrix} P_k - \beta_k \rho k \omega \\ \underbrace{\alpha_\omega \frac{\omega}{k} P_k}_{P_\omega} - \beta_\omega \rho \omega^2 + \underbrace{\sigma_d \frac{\rho}{\omega} \max\left(\frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 0 \right)}_{\mathcal{CD}} \end{matrix} \right\} \tag{3.41}$$

The turbulent viscosity is defined as

$$\mu_t = \frac{\rho k}{\omega} \tag{3.42}$$

All of the coefficients of the model are set as constants, given by: $\sigma_k = 2/3$, $\sigma_\omega = 0.5$, $\sigma_d = 0.5$, $\beta_k = 0.09$, $\beta_\omega = 0.075$, and $\alpha_\omega = \dfrac{\beta_\omega}{\beta_k} - \dfrac{\sigma_\omega \kappa^2}{\sqrt{\beta_k}}$, with $\kappa = 0.41$. Note that the constants σ_k and σ_ω may be modified using the options listed in Table 10.169. The k-ω-TNT model that is implemented in the EZAir suite of codes includes a modification pertaining to limiting the production term, namely, P_k is limited using:

$$P_k = \min\left(P_k, 10\beta_k \rho \omega k\right) \tag{3.43}$$

This was adopted from the k-ω-SST-2003 model (see Section 3.4.2.1). Another modification to the model pertains to the realizability of the linear Reynolds stress tensor through limiting of ω. The modification is based on the fact that two-equation turbulence models may suffer from spurious overproduction of turbulent kinetic energy in certain regions. This phenomenon, known as the *stagnation point anomaly*, usually appears in highly strained flows. The realizability constrains impose a lower bound on the specific dissipation rate. Note that the enforcement of realizability improves the robustness of the model.

A description of the boundary conditions appears in Section 3.6.

3.4.2 The k-ω-SST Model

One serious flaw exhibited by ω based models is the extreme sensitivity to the value of ω at irrotational boundaries of shear flows and, by implication, also to the value in weak-shear regions within a complex shear flow. This, as well as other defects, have led Menter [9] to formulate a hybrid model which blends the k-ω model near-wall regions with the k-ϵ model in regions that are far from walls. In recent years, this model has become the most popular two-equation model in aeronautical CFD practice, especially in weakly separated flows. Over the years, several variations and modifications to the original k-ω-SST model have appeared. The k-ω-SST-2003 (following the naming convention from the TMR website) has been chosen and implemented in the EZAir suite of codes.

3.4.2.1 The k-ω-SST-2003 Model

The k-ω-SST-2003 model source term is given by:

$$S = \left\{ \begin{matrix} P_k - \beta_k \rho k \omega \\ \underbrace{\alpha_\omega \frac{\rho}{\mu_t} P_k}_{P_\omega} - \beta_\omega \rho \omega^2 + \underbrace{\sigma_d \frac{\rho}{\omega} 2\left(1 - F_1\right) \max\left(\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 0\right)}_{\mathcal{CD}} \end{matrix} \right\} \tag{3.44}$$

Similar to the k-ω-TNT model, the coefficient β_k is set to $\beta_k = 0.09$. The coefficient of the cross diffusion term, σ_d is set to $\sigma_d \equiv \sigma_{\omega 2} = 0.856$. The rest of the model coefficients, ϕ_c=($\sigma_k,\ \sigma_\omega,\ \alpha_\omega,\ \beta_\omega$) are blended according to

$$\phi_c = F_1 \phi_1 + \left(1 - F_1\right) \phi_2 \tag{3.45}$$

where the coefficients ϕ_1 and ϕ_2 are given in Table 3.1.
The F_1 function, proposed by Menter [9] himself, is given as:

$$F_1 = \tanh\left(\mathfrak{z}_1{}^4\right) \tag{3.46}$$

	σ_k	σ_ω	α_ω	β_ω
ϕ_1	0.856	0.5	5/9	0.075
ϕ_2	1.0	0.856	0.44	0.0828

Table 3.1: k-ω-SST-2003 coefficients

with the argument $\mathfrak{z}_1$ given by:

$$\mathfrak{z}_1 = min\left[max\left(\frac{\sqrt{k}}{\beta_k \omega d}, \frac{500\mu}{\rho\omega d^2}\right), \frac{4\sigma_{\omega 2}\rho k}{max\left(2\sigma_{\omega 2}\frac{\rho}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-10}\right) d^2}\right] \tag{3.47}$$

where d designates the distance from the nearest wall. The turbulent viscosity is defined as

$$\mu_t = \frac{a_1 \rho k}{max\,(a_1\omega, SF_2)} \tag{3.48}$$

where $a_1 = 0.31$ and S is:

$$S = \sqrt{2S_{ij}S_{ij}} \tag{3.49}$$

with

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.50}$$

and the function F_2 is defined as follow:

$$F_2 = \tanh\left({\mathfrak{z}_2}^2\right) \tag{3.51}$$

with

$$\mathfrak{z}_2 = max\left(2\frac{\sqrt{k}}{\beta_k \omega d}, \frac{500\nu}{\omega d^2}\right) \tag{3.52}$$

In the k-ω-SST-2003 version, the production term, P_k is limited as follows

$$P_k = min\,(P_k, 10\beta_k \rho\omega k) \tag{3.53}$$

A description of the boundary conditions appears in Section 3.6.

3.4.2.2 k-ω-SST-SAS Model

The k-ω-SST scale-adaptive-simulation (SAS) model is based on the formulation appearing in the work by Egorov and Menter [10]. The difference between the k-ω-SST model and the k-ω-SST-SAS model is the addition of the Q_{SAS} source term to the equation for ω. Consequently, the k-ω-SST model source term (Equation 3.44) now reads:

$$S = \begin{Bmatrix} P_k - \beta_k \rho k \omega \\ \underbrace{\alpha_\omega \frac{\rho}{\mu_t} P_k}_{P_\omega} - \beta_\omega \rho \omega^2 + \underbrace{\sigma_d \frac{\rho}{\omega} 2 \left(1 - F_1\right) \max\left(\frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 0\right)}_{\mathcal{CD}} + Q_{SAS} \end{Bmatrix} \tag{3.54}$$

where Q_{SAS} is defined as:

$$Q_{SAS} = \max\left[\rho \xi \kappa S^2 \left(\frac{L_l}{L_{vk}}\right)^2 - C \frac{2\rho k}{\sigma_\phi} \max\left(\frac{|\nabla \omega|^2}{\omega^2}, \frac{|\nabla k|^2}{k^2}\right), 0\right] \tag{3.55}$$

The modeled turbulence length scale, L_l, is given by

$$L_l = \sqrt{k} / \left(C_\mu^{1/4} \omega\right) \tag{3.56}$$

and the von Karman length scale, L_{vk}, is given by:

$$L_{vk} = \frac{\kappa S}{\left|\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right)\right|} \tag{3.57}$$

The model constants are $C = 2.0$, $\xi = 3.51$, $\kappa = 0.41$, $\sigma_\phi = 0.67$, and $C_\mu = 0.09$.

3.5 Hybrid RANS/LES Turbulence Models

Recent years have seen the emergence of a new class of turbulence models, Hybrid RANS/LES models. Driven by the need for an accurate description of complex, unsteady, separated flows while limited in computational resources have been the driving force behind the development efforts. In cases where the detailed physics associated with turbulent flows or the aerodynamic noise produced by the flow, the RANS approach is not suited and one has to resort to more accurate approaches like Large-Eddy-Simulation (LES). The hybrid models act in RANS mode near the wall and switch to an LES mode further from the wall. This relieves the need for a high grid resolution near the wall that is required by a pure LES model. The basic motivation and rationale behind this new class of turbulence modeling lies in the argument that RANS models are capable of producing reasonable predictions of attached boundary layers, whereas the strength of LES is in modeling separated flow regions. Note that all hybrid models flow simulations are primarily targeted toward unsteady flow simulations.

Two families of hybrid RANS/LES are available for use in the EZAir suite of codes. The first, the Detached Eddy Simulation (DES) family and the second, the Partially Averaged Navier Stokes (PANS) family. The main difference between the two families is the special treatment of certain terms in the model equations. Whereas in the DES family the destruction term of the turbulent kinetic energy equation ($\beta_k \rho k \omega$) receives a special treatment, the destruction term of the specific dissipation term ($\beta_\omega \rho \omega^2$) receives the special treatment in the PANS family.

Three DES models are available, the X-LES model with the k-ω-TNT as the base model and the Delayed Detached Eddy Simulation (DDES) and the Improved Delayed Detached Eddy Simulation (IDDES) with the k-ω-SST-2003 as the base model. One PANS model is available and it also based on the k-ω-SST-2003 model. The following sections describe each of the available hybrid models.

3.5.1 Subgrid Length Scale

The hybrid models that are implemented in the EZAir suite of codes require the calculation of a subgrid characteristic length which is usually associated with the local grid size and/or the local flow characteristics. The code provides 5 different ways to set the subgrid characteristic length as described in the following sections. From here to forth, the subgrid length scale is denoted by Δ. For further details the user is referred to the paper by Trias *et al* [11].

3.5.1.1 Volume Based Subgrid Scale

The subgrid scale is evaluated as the cube root of the cell volume. For a Cartesian grid, it reads:

$$\Delta_{\mathrm{vol}} = (\Delta x \Delta y \Delta z)^{\frac{1}{3}} \tag{3.58}$$

3.5.1.2 Edge Size Subgrid Scale

The subgrid scale is evaluated as the smallest cell face divided by the cell volume. Let $s_i, i = 1, \ldots, 6$ be the surfaces of all cell faces

$$\Delta_{\mathrm{max}} = \frac{\mathrm{vol}}{\min\left(s_i, i = 1, \ldots, 6\right)} \tag{3.59}$$

3.5.1.3 LSQ Subgrid Scale

The LSQ subgrid scale is based on the velocity gradient stress tensor. Its definition and the rationale behind it are rather complex and are far beyond the scope of this manual (for further details refer to the paper by Trias *et al* [11]). It is defined as follows:

Let $\mathbf{G}$ be the velocity gradient tensor and let Δ be the second-order diagonal tensor: $\boldsymbol{\Delta} \equiv \mathrm{diag}\left(\Delta x, \Delta y, \Delta z\right)$. The LSQ subgrid scale is given by:

$$\Delta_{\mathrm{LSQ}} = \sqrt{\frac{\boldsymbol{\Delta}\mathbf{G}^T\mathbf{G} : \boldsymbol{\Delta}\mathbf{G}^T\mathbf{G}}{\mathbf{G}^T\mathbf{G} : \mathbf{G}^T\mathbf{G}}} \tag{3.60}$$

3.5.1.4 Vorticity Aligned Subgrid Scale

The subgrid scale is evaluated based on flow dependent characteristics as follows: let $\mathbf{l}_n$ be

$$\mathbf{l}_n = \frac{\omega}{|\omega|} \times \mathbf{r}_n \, (n = 1, \ldots, 8) \tag{3.61}$$

where $\omega = \nabla \times \mathbf{V}$ is the local vorticity vector and $\mathbf{r}_n$ is a vector between the cell center and the cell vertices. Therefore, the vorticity based subgrid scale is given by:

$$\tilde{\Delta}_\omega = \frac{1}{\sqrt{3}} \max |\mathbf{l}_n - \mathbf{l}_m|, n, m = 1, \ldots, 8. \tag{3.62}$$

3.5.1.5 Shear Layer Adapted Subgrid Scale

The Shear Layer Adapted (SLA) subgrid-scale is defined as:

$$\Delta_{\text{SLA}} = \tilde{\Delta}_\omega F_{\text{KH}} \, (\text{VTM}) \tag{3.63}$$

where VTM is the vortex tilting measure given by:

$$\text{VTM} = \frac{|(S \cdot \omega) \times \omega|}{\omega^2 \sqrt{-Q_S}} \tag{3.64}$$

where S is the rate-of-strain tensor and Q_S denotes the second invariant of the tensor. The VTM can be viewed as a measure of how much the rate-of-strain tensor tilts the vorticity vector toward another direction. The function F_{KH} is aimed at unlocking the Kelvin-Helmholtz instability in the initial part of shear layers. The function satisfies $0 \leq F_{KH} \, (VTM) \leq 1$, $F_{KH} \, (0) = 0$, and $F_{KH} \, (1) = 1$.

3.5.2 X-LES Hybrid RANS/LES Turbulence Model

The X-LES model is a hybrid model that is based on the k-ω-TNT model (see Section 3.4.1). Developed by Kok *et al* [12], it consists of a composite formulation incorporating both RANS and LES equations; it uses a clearly defined sub grid scale (SGS) model in the LES mode. Namely, in the LES mode, the two-equation turbulence model degenerates into one equation of the turbulence kinetic energy of the SGS model. The form of the convective and diffusive terms of the X-LES model equations is identical to that described in Section 3.4 (Equations (3.37) and (3.38)). A general, unified formulation of the k-ω-TNT/X-LES hybrid turbulence model source term is given by:

$$S = \begin{Bmatrix} P_k - \rho \dfrac{k^{1.5}}{l_k} \\ P_\omega - \beta_\omega \rho \omega^2 + \mathcal{CD} \end{Bmatrix} \tag{3.65}$$

The turbulent length scale is denoted by l_k, given here by the RANS-LES switch as follows:

$$l_k = \min\left(l_{RANS}, l_{LES}\right) \tag{3.66}$$

where the background RANS model length scale, l_{RANS}, is given as

$$l_{RANS} = \frac{k^{1/2}}{\beta_k \omega} \tag{3.67}$$

and l_{LES} is the LES length-scale defined via the sub grid length scale, Δ:

$$l_{LES} = C_{DES} \Delta \tag{3.68}$$

where $C_{DES} = 0.65$ and Δ is evaluated as described in Section 3.5.1 (see Table 10.156). The turbulent viscosity is defined as

$$\mu_t = l_\nu \rho k^{1/2} \tag{3.69}$$

The turbulence length scale, l_ν, is given by:

$$l_\nu = \min\left(l_{\nu_{RANS}}, l_{\nu_{RANS}}\right) \tag{3.70}$$

where

$$l_{\nu_{RANS}} = \frac{k^{1/2}}{\omega} \tag{3.71}$$

and

$$l_{\nu_{LES}} = C_1 \Delta \tag{3.72}$$

The coefficient C_1 is set to $C_1 = 0.06$ and Δ is evaluated as described in Section 3.5.1 (see Table 10.156).
A description of the boundary conditions appears in Section 3.6.

3.5.3 DDES/IDDES Hybrid RANS/LES Turbulence Model

The following two hybrid models, the delayed detached eddy simulation (DDES) and the improved delayed detached eddy simulation (IDDES) models are based on the k-ω-SST turbulence model (see Section 3.4.2). The form of the convective and diffusive terms of theDDES and IDDES model equations is identical to that described in Section 3.4 (Equations (3.37) and (3.38)). For complete details of the models the user is referred to the work by Gritskevich *et al* [13]. A general, unified formulation of the k-ω-SST-DDES/IDDES hybrid turbulence models source term is given by:

$$S = \begin{Bmatrix} P_k - \rho \dfrac{k^{1.5}}{l_k} \\ P_\omega - \beta_\omega \rho \omega^2 + \mathcal{CD} \end{Bmatrix} \tag{3.73}$$

The DDES length scale, l_k in Equation (3.73), is given by:

$$l_k = l_{RANS} - f_d \max\left(0, l_{RANS} - l_{LES}\right) \tag{3.74}$$

whereas the IDDES length scale is given by:

$$l_k = \tilde{f}_d \left(1 + f_e\right) l_{RANS} + \left(1 - \tilde{f}_d\right) l_{LES} \tag{3.75}$$

In both models l_{RANS} is the RANS model length scale

$$l_{RANS} = \frac{k^{1/2}}{\beta_k \omega} \tag{3.76}$$

and

$$l_{LES} = C_{DES} \Delta \tag{3.77}$$

with C_{DES} being:

$$C_{DES} = C_{DES1} F_1 + C_{DES2} (1 - F_1) \tag{3.78}$$

where $C_{DES1} = 0.78$ and $C_{DES2} = 0.61$ and the function F_1 is given in Equation (3.46). The main differences between the models are in the blending functions f_d and $\tilde{f}_d$ and the LES length scale Δ. These are outlined in the following sections.
A description of the boundary conditions appears in Section 3.6.

3.5.3.1 The k-ω-SST-DDES Models

The LES length scale, Δ is evaluated as described in Section 3.5.1 (see Table 10.158). The function f_d is a blending function given by:

$$f_d = 1 - \tanh\left[(C_{d1} \mathbf{r}_d)^{C_{d2}}\right] \tag{3.79}$$

where $C_{d1} = 20$ and $C_{d2} = 3$ and

$$\mathbf{r}_d = \frac{\nu_t + \nu}{\kappa^2 d^2 \sqrt{0.5 (S^2 + \Omega^2)}} \tag{3.80}$$

where d is the distance to the nearest wall, S is the magnitude of the strain stress tensor, Ω is the magnitude of the vorticity vector, and $\kappa = 0.41$.

3.5.3.2 The k-ω-SST-IDDES Models

The LES length scale, Δ is evaluated using:

$$\Delta = \min\left[C_\omega \max(d, \Delta_{gl}), \Delta_{sg}\right) \tag{3.81}$$

The coefficient C_ω is set as $C_\omega = 0.15$, $\Delta_{sg}$is evaluated as described in Section 3.5.1 (any of the choices) while Δ_{gl} is evaluated as described either in Section 3.5.1.1 or Section 3.5.1.2 only (see Table 10.159). The function $\tilde{f}_d$ is a blending function given by:

$$\tilde{f}_d = \max\left[(1 - f_{dt}), f_b\right] \tag{3.82}$$

where

$$f_{dt} = 1 - \tanh\left[\left(C_{dt1} \mathbf{r}_{dt} \right)^{C_{dt2}} \right] \tag{3.83}$$

with $C_{dt1} = 20$ and $C_{dt2} = 3$ and

$$\mathbf{r}_{dt} = \frac{\nu_t}{\kappa^2 d^2 \sqrt{0.5\left(S^2 + \Omega^2\right)}} \tag{3.84}$$

where d is the distance to the nearest wall, S is the magnitude of the strain stress tensor, Ω is the magnitude of the vorticity vector, and $\kappa = 0.41$. The function f_b is given by:

$$f_b = \min\left[2\ exp\left(-9\alpha^2\right), 1.0\right] \tag{3.85}$$

where

$$\alpha = 0.25 - \frac{d}{\Delta_{gl}} \tag{3.86}$$

The function f_e is given by:

$$f_e = f_{e2} \max\left[\left(f_{e1} - 1\right), 0\right] \tag{3.87}$$

where

$$f_{e1} = \begin{cases} 2\ exp\left(-11.09\alpha^2\right) & \alpha \geq 0 \\ 2\ exp\left(-9.0\alpha^2\right) & \alpha < 0 \end{cases} \tag{3.88}$$

and

$$f_{e2} = 1 - \max\left(f_t, f_l\right) \tag{3.89}$$

The function f_t and f_l are given by:

$$f_t = \tanh\left[\left(C_t^2 \mathbf{r}_{dt}\right)^3\right] \tag{3.90}$$

and

$$f_l = \tanh\left[\left(C_l^2 \mathbf{r}_{dl}\right)^3\right] \tag{3.91}$$

where

$$\mathbf{r}_{dl} = \frac{\nu}{\kappa^2 d^2 \sqrt{0.5\left(S^2 + \Omega^2\right)}} \tag{3.92}$$

The coefficients C_l and C_t are set to $C_l = 5.0$ and $C_t = 1.87$
A description of the boundary conditions appears in Section 3.6.

3.5.4 Partially Averaged Navier Stokes Turbulence Model

The governing equations of a k-ω based partially averaged Navier Stokes (PANS) Turbulence Model are given by:

$$\begin{aligned}
\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_j k}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[(\mu + \sigma_{ku}\mu_t)\frac{\partial k}{\partial x_j}\right] + P_k - \beta_k \rho k \omega \\
\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[(\mu + \sigma_{\omega u}\mu_t)\frac{\partial \omega}{\partial x_j}\right] + P_\omega - \beta_{\omega u}\rho\omega^2 + \mathcal{CD}
\end{aligned}$$

The base model of the PANS hybrid model that is implemented in the EZAir suite of codes is the k-ω-SST-2003 model and the therefore the cross diffusion term, $\mathcal{CD}$, is given by:

$$\mathcal{CD} = \sigma_{du}\frac{\rho}{\omega}2\left(1 - F_1\right)\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{3.93}$$

Note that the coefficients β_ω and $\sigma_{(k,\omega,d)}$ have been replaced by $\beta_{\omega u}$ and $\sigma_{(ku,\omega u,du)}$. The new coefficients, called the *unresolved* coefficients, are evaluated as follows:

$$\beta_{\omega u} = \alpha_\omega \beta_k - \alpha_\omega \frac{\beta_k}{f_\omega} + \frac{\beta_\omega}{f_\omega} \tag{3.94}$$

and

$$\sigma_{(ku,\omega u,du)} = \sigma_{(k,\omega,d)}\frac{f_\omega}{f_k} \tag{3.95}$$

where

$$f_\omega = \frac{f_\epsilon}{f_k} \tag{3.96}$$

and

$$f_k = f_k\left(\frac{\Lambda_{k\omega}}{\Delta}\right) \tag{3.97}$$

with

$$f_\epsilon = 1 \tag{3.98}$$

The argument $\Lambda_{k\omega}$ is evaluated using:

$$\Lambda_{k\omega} = \frac{\sqrt{k_{\text{total}}}}{\beta_k \omega} = \frac{\sqrt{k + k_r}}{\beta_k \omega} \tag{3.99}$$

where k_r is given by:

$$k_r = \frac{1}{2}\left(\overline{u_i u_i} - \overline{u}_i \overline{u}_j\right) \tag{3.100}$$

The averaging process starts from the time step where the PANS model has been invoked (saved and maintained after restart as well). However, the averages can be reset using the directive that is described in Table 10.165. The LES length scale (Δ) is evaluated as described in Section 3.5.1 (see Table 10.160).
A description of the boundary conditions appears in Section 3.6.

3.6 Boundary Conditions for the k-ω and Hybrid Turbulence Models

To close the solution of any of the k-ω models, the boundary conditions should be specified. The no-slip wall boundary conditions of k and ω, denoted by k_{wall} and ω_{wall}, respectively, are specified as follows:

$$k_{wall} = 0 \tag{3.101}$$

$$\omega_{wall} = 10 \frac{6\nu}{\beta_1 \left(\Delta d_1\right)^2} \tag{3.102}$$

where Δd_1 denotes the distance between the center of the first cell neighboring the wall and the wall. The inflow boundary condition of k, denoted by k_∞ for external flows is

$$k_\infty = \frac{3}{2}\left(T_u \cdot U_\infty\right)^2 \tag{3.103}$$

where Tu represents the turbulence intensity and U_∞ is the magnitude of the inflow velocity. The inflow boundary condition of ω, denoted as ω_∞ is specified as follows:

$$\omega_\infty = \frac{\rho_\infty k_\infty}{(\mu_t)_\infty} \tag{3.104}$$

with the recommended values of $(\mu_t)_\infty$ for external flows are as follows:

$$0.01 < \frac{(\mu_t)_\infty}{(\mu)_\infty} < 1.0 \tag{3.105}$$

3.7 Evaluation of LEVM Reynolds Stress Tensor

The SA-Edwards, the TNT, and the SST-2003 turbulence models are linear eddy viscosity models (LEVM), and therefore the Reynolds stress tensor that is added to the mean flow equations is defined based on the Boussinesq assumption, namely:

$$\mathfrak{R}_{ij} = \mathfrak{R}_{ij}^{LEVM} \tag{3.106}$$

where

$$\mathfrak{R}_{ij}^{LEVM} = 2\mu_t \left(S_{ij} - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \right) - \frac{2}{3}\rho k \delta_{ij} \tag{3.107}$$

and

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{3.108}$$

Note that the turbulence kinetic energy k is not available with the SA-Edwards model, therefore the term $\frac{2}{3}\rho k \delta_{ij}$ is neglected when using the SA-Edwards model.

3.8 Turbulence-Mean-flow Coupling

In eddy-viscosity-based models, the coefficients of viscosity μ and thermal conductivity κ are replaced by the relations

$$\begin{aligned}\mu &= \mu_l + \mu_t \\ \kappa &= \kappa_l + \frac{C_p \mu_t}{Pr_t}\end{aligned} \tag{3.109}$$

to account for the effects of turbulence on the mean-flow. The turbulent Prandtl number is assumed constant ($Pr_t = 0.9$).

3.9 SSG/LRR-ω Model

The compact, vector conservation-law form of an ω based, Reynolds stress model equations is given as follows:

$$\frac{\partial q}{\partial t} + \frac{\partial (e_c - e_d)}{\partial x} + \frac{\partial (f_c - f_d)}{\partial y} + \frac{\partial (g_c - g_d)}{\partial z} = s \tag{3.110}$$

where t denotes the time and x, y, and z denote the Cartesian coordinates. The vector q denotes the turbulence model dependent variables vector given by:

$$q = \begin{bmatrix} \rho\mathfrak{R}_{xx} \\ \rho\mathfrak{R}_{yy} \\ \rho\mathfrak{R}_{zz} \\ \rho\mathfrak{R}_{xy} \\ \rho\mathfrak{R}_{xz} \\ \rho\mathfrak{R}_{yz} \\ \rho\omega \end{bmatrix} \tag{3.111}$$

where $\mathfrak{R}_{ij} = \overline{\rho u_i'' u_j''}$ are the second moment closure model working variables and ω is the specific turbulent dissipation rate.

The SSG/LRR-ω inviscid fluxes are given by

$$e_c = uq, \qquad f_c = vq, \qquad g_c = wq \tag{3.112}$$

while the diffusive fluxes are given by

$$e_d = \begin{bmatrix} \left[\left(\mu\delta_{1k} + c_{\mathfrak{R}}\frac{\rho\mathfrak{R}_{xx_k}}{\omega}\right)\frac{\partial\mathfrak{R}_{x_ix_j}}{\partial x_k}\right] & m = 1,2,...,6 \\ \left[\left(\mu + c_\omega\mu_t\right)\frac{\partial\omega}{\partial x}\right] & m = 7 \end{bmatrix} \tag{3.113}$$

$$f_d = \begin{bmatrix} \left[\left(\mu\delta_{2k} + c_{\mathfrak{R}}\frac{\rho\mathfrak{R}_{yx_k}}{\omega}\right)\frac{\partial\mathfrak{R}_{x_ix_j}}{\partial x_k}\right] & m = 1,2,...,6 \\ \left[\left(\mu + c_\omega\mu_t\right)\frac{\partial\omega}{\partial y}\right] & m = 7 \end{bmatrix} \tag{3.114}$$

$$g_d = \begin{bmatrix} \left[\left(\mu\delta_{3k} + c_{\mathfrak{R}}\frac{\rho\mathfrak{R}_{zx_k}}{\omega}\right)\frac{\partial\mathfrak{R}_{x_ix_j}}{\partial x_k}\right] & m = 1,2,...,6 \\ \left[\left(\mu + c_\omega\mu_t\right)\frac{\partial\omega}{\partial z}\right] & m = 7 \end{bmatrix} \tag{3.115}$$

where δ_{ij} is the Kronecker delta function and the turbulent viscosity, μ_t is given by:

$$\mu_t = \frac{\rho\left(\mathfrak{R}_{xx} + \mathfrak{R}_{yy} + \mathfrak{R}_{zz}\right)}{2\omega} \tag{3.116}$$

The source term vector of the model and complete details of the model and its coefficients may be found in the original publication. [14]

3.10 Turbulence Model and Scheme Selection

The type of turbulence model may be selected using the "-solve" directive and the "model.tu" parameter (see Tables 10.153, 10.154, 10.157, 10.162, 10.156, 10.158, 10.159, 10.160, and 10.163). The time-marching scheme for the turbulence model equations may differ from that of the mean flow equations. In this case the scheme is set using the "-solve" directive and the "scheme.tu" parameter (see Table 10.172).

Chapter 4

Computational Methods

4.1 Spatial Discretization

A conservative cell-centered finite volume methodology is employed to discretize the governing equations. The computational domain is a structured curvilinear grid that is discretized into N_{cv} non-overlapping control volumes. A control volume, C_v is defined by a grid volume element and $\partial\Gamma$ is the volume control surface, with $\mathbf{n} = [n_x, n_y, n_z]^T$ being the outward-pointing, unit normal vector to $\partial\Gamma$. Therefore, Equation (2.6) for a control volume C_v can be expressed as:

$$\frac{\partial}{\partial t}\int\limits_{C_v} Q dV + \int\limits_{\partial\Gamma} H dS = 0 \tag{4.1}$$

where H is the rotated flux, namely,

$$\begin{aligned} H &= H_c + H_d \\ H_c &= E_c n_x + F_c n_y + G_c n_z \\ H_d &= E_d n_x + F_d n_y + G_d n_z \end{aligned} \tag{4.2}$$

The term H_c is the convective part of the flux while H_d is the diffusive part of the flux. The semi-discrete form of Equation (4.1) for a non-deforming cell i is given by:

$$V_i \frac{dQ_i}{dt} = -\sum_{j \in N(i)} H_{ij} S_{ij} = R_i \tag{4.3}$$

where V_i denotes the cell volume, Q_i is the vector of cell-averaged conservative dependent variables, $N(i)$ denotes the set of cell i neighbors, H_{ij} is the rotated flux vector normal to the interface ij shared by cell i and cell j, and S_{ij} is the interface area. The term R_i signifies the residual of the equations. Note that except at singular points, each cell has exactly $N(i) = 6$ neighbors. In what follows, the subscript "i" is dropped for compactness of the representation.

4.1.1 Geometric Conversation Law

For deforming and/or moving meshes, a geometric conservation law (GCL) is introduced to prevent the violation of the conservation of properties [15]:

$$\frac{\partial \Omega}{\partial t} = \int_{\Gamma} \dot{X} \cdot \hat{n} d\Gamma = 0 \tag{4.4}$$

where Ω is the cell volume , $\dot{X}$ is the cell face velocity and Γ is cell surface. The time derivative from Equation (4.4) is added to the right hand side operator such that the GCL equality that appears in the equation is maintained.

4.2 Flux Approximation Schemes

In flux difference splitting, the problem of computing the cell-face fluxes for a control volume is viewed as a series of one-dimensional Riemann problems along the direction normal to the control-volume faces. Because some of the details of the exact solution, obtained at considerable cost, are lost in the cell-averaged representation of the data, the solution of the full Riemann problem is usually replaced by methods referred to as approximate Riemann solvers. In what follows, the schemes that are used in the

EZAir suite of codes are described in detail.

4.2.1 HLLC

The concept of average-state approximations was introduced by Harten, Lax and van-Leer [16] in 1983. The Harten, Lax and van-Leer (HLL) scheme is attractive because of its robustness, conceptual simplicity, and ease of coding, but it has the serious flaw of a diffusive contact surface. This is mainly because the HLL solver reduces the exact Riemann problem to two pressure waves and therefore neglects the contact surface. Toro *et al* [17] discussed this limitation, and proposed a modified three wave solver, named HLLC, where the contact discontinuity is explicitly present. The HLLC scheme is found to have the following properties:

1. Exact preservation of isolated contact discontinuities and shear waves.

2. Positivity preserving of a scalar quantity.

3. Enforcement of the entropy condition.

The resulting scheme greatly improves contact discontinuity resolution and has been successfully used to compute compressible viscous and turbulent flows [18].

The HLLC approximate Riemann solver that is implemented in the EZAir suite of codes is as proposed by Batten *et al* [18]. The HLLC scheme assumes two intermediate states, $\mathbf{Q}_l^*$ and $\mathbf{Q}_r^*$ within the region bounded by the left moving wave, S_l, and the right moving wave, S_r (the subscripts "l" and "r" denote the left and right states of the approximate Riemann problem, respectively). The states $\mathbf{Q}_l^*$ and $\mathbf{Q}_r^*$ are split by the contact discontinuity, which moves with the velocity S_m (see Figure 4.1).

Two wave speed estimates can be used. In the first, the wave speeds S_l and S_r are computed according to Einfeldt *et al* [19] as follows:

$$S_l = \min[\lambda_{min}, \lambda_{min}^{Roe}] \tag{4.5}$$

$$S_r = \max[\lambda_{max}, \lambda_{max}^{Roe}] \tag{4.6}$$

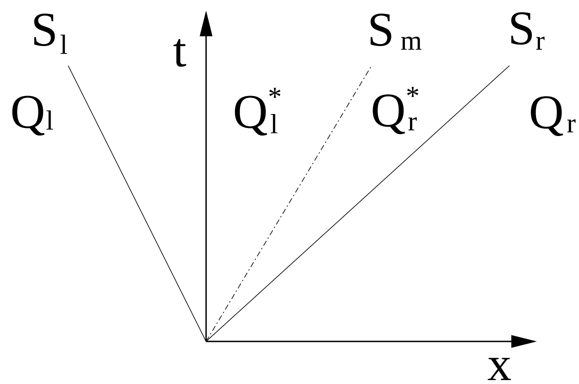


Figure 4.1: Wave structure of the HLLC approximate Riemann solver

where λ_{min} is the smallest eigenvalue and λ_{max} is the largest eigenvalue, evaluated at the interface. Similarly, λ_{min}^{Roe} and λ_{max}^{Roe} are the smallest and largest eigenvalues of Roe's average matrix [20], respectively. In the second, the wave speeds S_l and S_r are computed according to Davis [21] as follows:

$$S_l = \min[\lambda_{min}(L), \lambda_{min}(R)] \tag{4.7}$$

$$S_r = \max[\lambda_{max}(L), \lambda_{max}(R)] \tag{4.8}$$

The normal velocity to the interface, denoted by q, is defined as:

$$q = (u - u_g)\, n_x + (v - v_g)\, n_y + (w - w_g)\, n_z \tag{4.9}$$

The contact discontinuity speed S_m is evaluated according to Batten *et al* [18] by

$$S_m = \frac{\rho_r q_r (S_r - q_r) - \rho_l q_l (S_l - q_l) + p_l - p_r}{\rho_r (S_r - q_r) - \rho_l (S_l - q_l)} \tag{4.10}$$

This choice of S_m enforces the equality of the two star pressures, *i.e.*, $p^* = p_l^* = p_r^*$ which is obtained from

$$p^* = \rho_l (q_l - S_l)(q_l - S_m) + p_l = \rho_r (q_r - S_r)(q_r - S_m) + p_r \tag{4.11}$$

Introducing the intermediate left state vector

$$\mathbf{Q}_l^* = \begin{Bmatrix} \rho_l^* \\ (\rho u)_l^* \\ (\rho v)_l^* \\ (\rho w)_l^* \\ e_l^* \end{Bmatrix} = \Omega_l \begin{Bmatrix} \rho_l (S_l - q_l) \\ (S_l - q_l)(\rho u)_l + (p^* - p_l)\, n_x \\ (S_l - q_l)(\rho v)_l + (p^* - p_l)\, n_y \\ (S_l - q_l)(\rho w)_l + (p^* - p_l)\, n_z \\ (S_l - q_l)\, e_l - p_l q_l + p^* S_m \end{Bmatrix} \tag{4.12}$$

where $\Omega_l \equiv (S_l - S_m)^{-1}$. The left state flux vector becomes

$$\mathbf{H}_{c_l}^* \equiv \mathbf{H}_c (\mathbf{Q}_l^*) = \mathbf{Q}_l^* S_m + \begin{Bmatrix} 0 \\ p^* n_x \\ p^* n_y \\ p^* n_z \\ p^* S_m \end{Bmatrix} \tag{4.13}$$

and the corresponding intermediate right state vector and right flux vector are obtained from Equations (4.12), (4.13) by interchanging the subscripts $l \to r$. Finally, the numerical HLLC flux is defined as follow

$$\mathbf{H}_c (\mathbf{Q}_l, \mathbf{Q}_r) = \begin{Bmatrix} \mathbf{H}_c (\mathbf{Q}_l) & if\ S_l > 0 \\ \mathbf{H}_c (\mathbf{Q}_l^*) & if\ S_l \le 0 < S_m \\ \mathbf{H}_c (\mathbf{Q}_r^*) & if\ S_m \le 0 \le S_r \\ \mathbf{H}_c (\mathbf{Q}_r) & if\ S_r < 0 \end{Bmatrix} \tag{4.14}$$

where $\mathbf{H}_c (\mathbf{Q}_l)$ and $\mathbf{H}_c (\mathbf{Q}_r)$ are the left and right analytic flux vectors, respectively.

4.2.2 AUSM

The advection upstream splitting method (AUSM) was first introduced in the year 1993 by Liou and Steffen [22]. The development of the AUSM was motivated by the desire to combine the efficiency of flux vector splitting methods (FVS) and the accuracy of flux differencing splitting methods (FDS). The key idea behind AUSM schemes is the the fact that the inviscid flux vector consists of two physically distinct parts, namely the *convective* terms and the *pressure* terms. The convective terms can therefore be considered as passive scalar quantities convected by a suitably defined velocity. On the other hand, the pressure flux terms are governed by the acoustics wave speeds.

4.2.2.1 AUSM$^+$-up

Although AUSM schemes enjoy a demonstrated improvement in accuracy, efficiency, and robustness over existing schemes, they have been found to have deficiencies in some cases. In the year 1996, Liou improved the original AUSM, termed now the AUSM$^+$ [23]. Among the improvement features of the original AUSM scheme are the following properties: (1) exact resolution of a one-dimensional contact discontinuity and shock discontinuities, (2) positivity preserving of scalar quantities, (3) free of "carbuncle phenomenon".

In the year 2006, Liou introduced a sequel scheme to the AUSM$^+$ called the AUSM$^+$-up [24] extended for all speed flows. The AUSM$^+$-up is implemented in the EZAir suite of codes and it is given as follows:

$$\mathbf{F}_c\left(\mathbf{Q}_l,\mathbf{Q}_r\right)=\mathbf{p}_{1/2}+\dot{m}_{1/2}\left\{\begin{matrix}\boldsymbol{\psi}_l & \text{if } M_{1/2}>0\\ \boldsymbol{\psi}_r & \text{otherwise}\end{matrix}\right\} \tag{4.15}$$

where

$$\boldsymbol{\psi}_{l/r}=\left\{\begin{matrix}1\\ u_{l/r}\\ v_{l/r}\\ w_{l/r}\\ H_{l/r}\end{matrix}\right\}, \tag{4.16}$$

the mass flux, $\dot{m}_{1/2}$ is defined as

$$\dot{m}_{1/2}=a_{1/2}M_{1/2}\left\{\begin{matrix}\rho_l & \text{if } M_{1/2}>0\\ \rho_r & \text{otherwise}\end{matrix},\right\} \tag{4.17}$$

and the pressure flux, $\mathbf{p}_{1/2}$ is given as

$$\mathbf{p}_{1/2} = \left\{ \begin{matrix} 0 \\ p_{1/2}n_x \\ p_{1/2}n_y \\ p_{1/2}n_z \\ 0 \end{matrix} \right\} \tag{4.18}$$

where

$$p_{1/2} = \mathcal{P}^{+}_{(5)}(M_l)p_l + \mathcal{P}^{-}_{(5)}(M_r)p_r - K_u\mathcal{P}^{+}_{(5)}(M_l)\mathcal{P}^{-}_{(5)}(M_r)(\rho_l + \rho_r)(f_a a_{1/2})(q_r - q_l) \tag{4.19}$$

is the interface pressure. The interface-normal velocity is denoted by q, H denotes the specific total enthalpy, and K_u is a constant that equals 0.75 (may be set by the user; see Table 10.170). The remaining functions are given below. The left/right Mach number at the interface, $M_{l/r}$, is defined as follows:

$$M_{l/r} = \frac{q_{l/r}}{a_{1/2}} \tag{4.20}$$

where $a_{1/2}$ is the speed of sound at the interface and it may be calculated by a simple average of a_l and a_r:

$$a_{1/2} = \frac{a_l + a_r}{2} \tag{4.21}$$

Next, the Mach number at the interface, $M_{1/2}$ is calculated as follows:

$$\begin{aligned} \overline{M}^2 &= \frac{q_l^2 + q_r^2}{2a_{1/2}^2} \\ M_o^2 &= min\left(1, max\left(\overline{M}^2, M_\infty^2\right)\right) \\ f_a\left(M_o\right) &= M_o\left(2 - M_o\right) \\ \rho_{1/2} &= \frac{\rho_l + \rho_r}{2} \\ M_{1/2} &= \mathcal{M}^{+}_{(4)}\left(M_l\right) + \mathcal{M}^{-}_{(4)}\left(M_r\right) - \frac{K_p}{f_a} max\left(1 - \sigma\overline{M}^2, 0\right)\frac{p_r - p_l}{\rho_{1/2}a_{1/2}^2} \end{aligned} \tag{4.22}$$

with the constants $K_p = 0.25$ and $\sigma = 1.0$. The split Mach numbers $\mathcal{M}_m^{+/-}$ are polynomial functions of degree m (=1,2,4) of the Mach number, M, given as follows:

$$\begin{aligned}
\mathcal{M}_1^{\pm} &= \frac{1}{2}\left(M \pm |M|\right) \\
\mathcal{M}_2^{\pm} &= \pm\frac{1}{4}\left(M \pm 1\right)^2 \\
\mathcal{M}_{(4)}^{\pm}(M) &= \begin{Bmatrix} \mathcal{M}_{(1)}^{\pm} & if\ |M| > 0 \\ \mathcal{M}_{(2)}^{\pm}(1 \mp 16\beta\mathcal{M}_{(2)}^{\mp}) & otherwise \end{Bmatrix}
\end{aligned} \tag{4.23}$$

with the constant $\beta = 1/8$. Finally, the pressure polynomials are given as:

$$\mathcal{P}_{(5)}^{\pm}(M) = \begin{Bmatrix} \frac{1}{M}\mathcal{M}_{(1)}^{\pm} & if\ |M| \geq 1 \\ \mathcal{M}_{(2)}^{\pm}[(\pm 2 - M) \mp 16\alpha M\mathcal{M}_{(2)}^{\mp}) & otherwise \end{Bmatrix} \tag{4.24}$$

with the function $\alpha = \frac{3}{16}(-4 + 5f_a^2)$.

4.2.2.2 Extension of the AUSM$^+$-up to Unsteady Low Mach Number Flows: AUSM-JCAM13

The extension is based on modifying the artificial pressure dissipation of the mass flux in the AUSM$^+$-up scheme. Namely, replacing the $M_{1/2}$ term in Equation (4.22) to the following:

$$\begin{aligned}
M_{1/2} &= \mathcal{M}_{(4)}^{+}\left(M_l\right) + \mathcal{M}_{(4)}^{-}\left(M_r\right) \\
&- K_p \max\left(1 - \sigma\overline{M}^2, 0\right) \frac{p_r - p_l}{\rho_{1/2}a_{1/2}\left(f_a a_{1/2} + C\frac{\Delta x_c}{\Delta t} f_t\right)}
\end{aligned} \tag{4.25}$$

where C is a constant Δt is the physical time step, Δx_c is a characteristic length that is based on the local minimal neighboring cell distance, and f_t is a function that is designed to gradually retrieve the AUSM$^+$-up as the Mach number approaches unity. The default of the constant C is $C = 1$ and it can be changed by the user (see Table 10.188). For supersonic flows the scheme reverts to the original AUSM$^+$-up scheme. The function f_t is given by

$$f_t = \max(0.0, 1 - M_f) \tag{4.26}$$

where M_f is the ratio between the cell face velocity magnitude and cell face speed of sound.

4.2.2.3 AUSMDV

The AUSMDV [25] flux uses a blending function to combine the flux-difference (AUSMD) and flux-vector (AUSMV) versions of the basic AUSM flux to obtain a robust scheme that accurately resolves shock waves. The AUSMDV scheme has several properties that are favorable for hypersonic flows (*e.g.*, high-resolution of contact discontinuities, conservation of enthalpy for steady flows and applicability to chemically reacting flows). Therefore, this variant of the AUSM flux is used by default in simulations with the multi-component gas (MCG) physical model.

The AUSMDV flux is given as follows [25]. First, define the splittings of the interface-normal velocity, q, as follows:

$$q_l^+ = \begin{cases} \alpha_l \frac{(q_l+a_m)^2}{4a_m} + (1-\alpha_l)q_l^{pos} & \text{if } |q_l| \le a_m, \\ q_l^{pos} & \text{otherwise} \end{cases} \tag{4.27}$$

$$q_r^- = \begin{cases} -\alpha_r \frac{(q_r-a_m)^2}{4a_m} + (1-\alpha_r)q_r^{neg} & \text{if } |q_r| \le a_m, \\ q_r^{pos} & \text{otherwise} \end{cases} \tag{4.28}$$

Where,

$$q_l^{pos} = \frac{1}{2}(q_l + |q_l|) \tag{4.29}$$

$$q_r^{neg} = \frac{1}{2}(q_r - |q_r|) \tag{4.30}$$

with q_l, q_r denoting the interface-normal velocities evaluated from the left and right states (cells), respectively. Furthermore, the interface speed of sound is given by

$$a_m = max(a_l, a_r); \tag{4.31}$$

and

$$\alpha_l = \frac{2(p_l/\rho_l)}{p_l/\rho_l + p_r/\rho_r} \tag{4.32}$$

$$\alpha_r = \frac{2(p_r/\rho_r)}{p_l/\rho_l + p_r/\rho_r} \tag{4.33}$$

are functions designed to avoid dissipation at contact discontinuities.

Then, the mass flux is simply defined as

$$\dot{m}_{1/2} = \rho_l q_l^+ + \rho_r q_r^- \tag{4.34}$$

Next, define the interface pressure as

$$p_{1/2} = p_l^+ + p_r^- \tag{4.35}$$

where the splittings of the pressure, p_l^+, p_r^- are given as follows:

$$p_l^+ = \begin{cases} p_l \frac{(q_l+a_m)^2}{4a_m^2}(2 - \frac{q_l}{a_m}) & \text{if } |q_l| \leq a_m, \\ p_l \frac{q_l^{pos}}{q_l} & \text{otherwise} \end{cases} \tag{4.36}$$

$$p_r^- = \begin{cases} p_r \frac{(q_r-a_m)^2}{4a_m^2}(2 + \frac{q_r}{a_m}) & \text{if } |q_r| \leq a_m, \\ p_r \frac{q_l^{neg}}{q_r} & \text{otherwise} \end{cases} \tag{4.37}$$

(4.38)

Then, the momentum flux vector of AUSMDV is defined by blending of the AUSMD and AUSMV fluxes, as follows:

$$(\rho \mathbf{u} q)_{1/2} = \frac{1}{2}\left[(1+s)\left(\rho \mathbf{u} q\right)_{AUSMV} + (1-s)\left(\rho \mathbf{u} q\right)_{AUSMD}\right] + p_{1/2}\mathbf{n} \tag{4.39}$$

Where the momentum flux vectors of the AUSMV and AUSMD variants are given by:

$$(\rho \mathbf{u} q)_{AUSMV} = \rho_l q_l^+ \mathbf{u_l} + \rho_r q_r^- \mathbf{u_r} \tag{4.40}$$

$$(\rho \mathbf{u} q)_{AUSMD} = \frac{1}{2}\left[\dot{m}_{1/2}\left(\mathbf{u_l} + \mathbf{u_r}\right) - \left|\dot{m}_{1/2}\right|\left(\mathbf{u_r} - \mathbf{u_l}\right)\right] \tag{4.41}$$

where $\mathbf{u}_{l/r} = (u, v, w)_{l/r}$ denote the velocity vectors evaluated from the left and right states (cells) of the interface, and $s = min\left[1, K\frac{|p_r - p_l|}{min(p_l, p_r)}\right]$ is the blending function. K is typically set equal to 10.

The total energy flux is defined by

$$(\rho H q)_{1/2} = \frac{1}{2}\left[\dot{m}_{1/2}\left(H_l + H_r\right) - \left|\dot{m}_{1/2}\right|\left(H_r - H_l\right)\right] \tag{4.42}$$

And the final expression for the interface-normal AUSMDV flux vector is:

$$\mathbf{F}_c\left(\mathbf{Q}_l, \mathbf{Q}_r\right) = \begin{Bmatrix} \dot{m}_{1/2} \\ (\rho u q)_{1/2} \\ (\rho v q)_{1/2} \\ (\rho w q)_{1/2} \\ (\rho H q)_{1/2} \end{Bmatrix} \tag{4.43}$$

4.2.3 LDFSS

In a similar fashion to the AUSM$^+$-up scheme, the inviscid flux can be split into the convective and pressure contributions as:

$$\begin{aligned} F &= F^c + F^p \\ F &= q\Phi + p_{1/2}\Psi \end{aligned} \tag{4.44}$$

where

$$\Phi = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho H \end{bmatrix}, \ \Psi = \begin{bmatrix} 0 \\ n_x \\ n_y \\ n_z \\ 0 \end{bmatrix} \tag{4.45}$$

The LDFSS scheme [26] applied to the cell interfaces results in the inviscid flux being split into convective and pressure parts as the sum of left and right components,

$$F_{i+1/2} = F^c_{i+1/2} + F^p_{i+1/2} \tag{4.46}$$

where

$$\begin{aligned} F^c_{i+1/2} &= U^+\Phi_L + U^-\Phi_R \\ F^p_{i+1/2} &= p_{1/2}\Psi \end{aligned} \tag{4.47}$$

The split velocities $U^{\pm}$ and the interface pressure $p_{1/2}$ are functions of the Mach number, and are defined as:

$$\begin{aligned} U^+ &= \tilde{a}_{1/2}\left[M^+ - M_{1/2}\left(1 - \frac{p_L - p_R}{2\rho_L V^2_{\text{ref},1/2}}\right)\right] \\ U^- &= \tilde{a}_{1/2}\left[M^- - M_{1/2}\left(1 - \frac{p_L - p_R}{2\rho_R V^2_{\text{ref},1/2}}\right)\right] \end{aligned} \tag{4.48}$$

and

$$p_{1/2} = \frac{1}{2}(p_L + p_R) + \frac{1}{2}\left(P^+ - P^-\right)(p_L + p_R) + \rho_{1/2}V^2_{\text{ref},1/2}\left(P^+ + P^- - 1\right) \tag{4.49}$$

respectively, where

$$\begin{aligned} M^+ &= \alpha^+_L(1+\beta_L)M_L - \beta_L M^+_{(2),L} \\ M^- &= \alpha^+_R(1+\beta_R)M_R - \beta_R M^-_{(2),R} \end{aligned} \tag{4.50}$$

In the above equations, $\rho_{1/2}$ is the interface density, determined using an arithmetic average, and $V_{\text{ref},1/2}$ is the interface reference velocity that is based on

$$V_{\text{ref}}^2 = \min\left[a^2, \max\left(\left|\vec{V}\right|^2, V_\infty^2\right)\right] \tag{4.51}$$

and arithmetic averages of the velocity and sound speed. Other quantities that are used in the preceding definitions are the stagnation and sonic-point switching functions:

$$\begin{aligned} \alpha_{L,R}^{\pm} &= \frac{1}{2}\left[1.0 \pm \text{sign}\left(1.0, M_{L,R}\right)\right] \\ \beta_{L,R} &= -\max\left[0.0, 1.0 - \text{int}\left(\left|M_{L,R}\right|\right)\right] \end{aligned} \tag{4.52}$$

Finally, the pressure splitting is realized using the Leer/Liou polynomial of degree 1 [27]:

$$P^{\pm} = \alpha_{L,R}^{\pm}\left(1 + \beta_{L,R}\right) - \frac{\beta_{L,R}}{2}\left[1.0 \pm M_{L,R}\right] \tag{4.53}$$

4.2.3.1 Extension of the LDFSS to Unsteady Low Mach Number Flows: LDFSS-JCAM13

The extension of the LDFSS scheme to unsteady low Mach number flow is also based on modifying the pressure dissipation term [28]. This is conducted by rewriting Equations (4.48) and (4.49) with two reference velocities,

$$\begin{aligned} U^+ &= \tilde{a}_{1/2}\left[M^+ - M_{1/2}\left(1 - \frac{p_L - p_R}{2\rho_L V_{\text{ref},c,1/2}^2}\right)\right] \\ U^- &= \tilde{a}_{1/2}\left[M^- - M_{1/2}\left(1 - \frac{p_L - p_R}{2\rho_R V_{\text{ref},c,1/2}^2}\right)\right] \end{aligned} \tag{4.54}$$

$$p_{1/2} = \frac{1}{2}\left(p_L + p_R\right) + \frac{1}{2}\left(P^+ - P^-\right)\left(p_L + p_R\right) + \rho_{1/2} V_{\text{ref},p,1/2}^2\left(P^+ + P^- - 1\right) \tag{4.55}$$

where $V_{\text{ref},p,1/2}$ is determined according to Equation (4.51) and $V_{\text{ref},c,1/2}$ is determined according to

$$V^2_{\text{ref},c,1/2} = \min\left\{a^2, \max\left[\left|\vec{V}\right|^2, V^2_{\infty}, \left(\frac{\Delta x_c}{\Delta t}\right)^2\right]\right\} \tag{4.56}$$

where Δt is the physical time step and Δx_c is a characteristic length that is based on the local minimal neighboring cell distance.

4.2.4 The AUFSR and AUFSR+ Schemes

The AUFSR scheme is obtained by hybridizing the AUFS (artificial upstream flux splitting) scheme with the Roe scheme. For ease of derivation and presentation two vectors are defined, the vector of the conservative variables, $\mathbf{U}$,

$$\mathbf{U} = [\rho, \rho u, \rho v, \rho w, E]^T \tag{4.57}$$

and the vector $\mathbf{P}$:

$$\mathbf{P} = [0, pn_x, pn_y, pn_z, pq]^T \tag{4.58}$$

where ρ is the fluid density, u, v, and w, denote the Cartesian velocity vector components, and E denotes the total energy. A cell interface normal velocity is denoted by $q = un_x + vn_y + wn_z$, with $[n_x, n_y, n_z]^T$ being the unit vector normal to the cell interface and p denotes the pressure. The AUFSR flux approximation may be therefore written as:

$$\mathbf{F}^{AUFSR} = (1-M)\left[\frac{1}{2}\left(\mathbf{P}^L + \mathbf{P}^R\right) + \mathbf{D}_{AUFSR}\right] + M\left[\mathbf{U}^{\beta}\left(q^{\beta} - s_2\right) + \mathbf{P}^{\beta}\right] \tag{4.59}$$

where $M = \dfrac{s_1}{s_1 - s_2}$, and s_1, s_2, whose definitions follow below, are two scalars representing two artificial wave speeds. The superscript β distinguishes between information from the left state or the right state depending on the sign of s_1 as follows

$$\beta = \begin{cases} L \quad for \quad s_1 > 0, \\ R \quad for \quad s_1 \leq 0. \end{cases} \tag{4.60}$$

The vector $\mathbf{D}_{AUFSR}$ denotes the AUFSR dissipation defined as follows:

$$\mathbf{D}_{AUFSR} = \frac{1}{2}\left[\left(q^L - s_1\right)\mathbf{U}^L + \left(q^R - s_1\right)\mathbf{U}^R\right] - \frac{1}{2}\sum_{j=1}^{5}\eta_k\left|\tilde{\lambda}_k\right|\widetilde{\mathbf{R}}_k \tag{4.61}$$

where η_k are the wave strengths, $\tilde{\lambda}_k$ are the eigenvalues and $\widetilde{\mathbf{R}}_k$ are the eigenvectors of the Roe dissipation matrix (according to Roe and Pike [29]), and $(\tilde{\ })$ denotes the Roe averages. It should be noted that the eigenvalues within the AUFSR scheme, $\tilde{\lambda}_k$, represent the eigenvalues shifted by the artificial wave speed s_1, namely

$$\tilde{\lambda}_1 = \tilde{q} - s_1 - \tilde{c}, \quad \tilde{\lambda}_{2,3,4} = \tilde{q} - s_1, \quad \tilde{\lambda}_5 = \tilde{q} - s_1 + \tilde{c}, \tag{4.62}$$

where $\tilde{c}$ is the Roe averaged speed of sound. To completely determine the numerical fluxes, the values of the artificial velocities must be known. They are the lower and upper bounds of the physical wave speeds with which the information of the initial discontinuity is transported. To satisfy the entropy and the positivity conditions, Einfeldt *et al* [19] suggested adequate bounds by making use of the Roe-averaged eigenvalues. Then, the approximate wave speeds are simply set to their Roe average

$$s_1 = \tilde{q} \tag{4.63}$$

and s_2 is defined as follow

$$s_2 = \begin{cases} min\left(q^L - c^L,\ \tilde{q} - \tilde{c},\ 0\right) & if \quad s_1 > 0 \\ max\left(q^R + c^R,\ \tilde{q} + \tilde{c},\ 0\right) & otherwise \end{cases} \tag{4.64}$$

Note that thanks to the choice of $s_1 = \tilde{q}$ the eigenvalues become

$$\tilde{\lambda}_1 = -\tilde{c}, \quad \tilde{\lambda}_{2,3,4} = 0, \quad \tilde{\lambda}_5 = \tilde{c}, \tag{4.65}$$

and hence the final form of the AUFSR dissipation vector becomes

$$\mathbf{D}_{AUFSR} = \frac{1}{2}\left[\left(q^L - \tilde{q}\right)\mathbf{U}^L + \left(q^R - \tilde{q}\right)\mathbf{U}^R\right] - \frac{\tilde{c}}{2}\left(\eta_1\widetilde{\mathbf{R}}_1 + \eta_5\widetilde{\mathbf{R}}_5\right) \tag{4.66}$$

where

$$\eta_1 = \frac{\Delta p - \tilde{\rho}\tilde{c}\Delta q}{2\tilde{c}^2}, \quad \eta_5 = \frac{\Delta p + \tilde{\rho}\tilde{c}\Delta q}{2\tilde{c}^2} \tag{4.67}$$

$$\widetilde{\mathbf{R}}_1 = \left[1,\ \tilde{u} - \tilde{c}n_x,\ \tilde{v} - \tilde{c}n_y,\ \tilde{w} - \tilde{c}n_z,\ \widetilde{H} - \tilde{c}\tilde{q}\right]^T \tag{4.68a}$$

$$\widetilde{\mathbf{R}}_5 = \left[1,\ \tilde{u} + \tilde{c}n_x,\ \tilde{v} + \tilde{c}n_y,\ \tilde{w} + \tilde{c}n_z,\ \widetilde{H} + \tilde{c}\tilde{q}\right]^T \tag{4.68b}$$

The operator Δ denotes the difference between the right and the left states (*e.g.*, $\Delta p = p^R - p^L$) and $\widetilde{H}$ is the averaged Roe total enthalpy.

4.2.4.1 Low Mach Number Extension, the AUFSR+ Scheme

Upwind schemes that are designed for compressible flows, fail to handle low Mach number flows [30]. A simple modification to the AUFSR scheme dissipation term is introduced, resulting in an all-speed AUFSR scheme, called the AUFSR+ scheme. Thanks to the explicit form of the AUFSR dissipation part, the required low-Mach extension is easily introduced. The basic idea is to scale the jump of the velocity in the normal direction to the cell interface by a simple function of the Mach number as was proposed by Rieper [31]. According to Rieper, at the lower limit of a zero Mach number, the function should obtain the value of zero. This would result in a vanishing of the normal velocity jump. There are several possibilities to define such a function, called hereafter ζ. Mor-Yossef [32] has chosen the simple function that is used by Shima and Kitamura in their SLAU scheme [33]:

$$\zeta = min\left[1, \frac{1}{c_{1/2}}\sqrt{\frac{u_L^2 + v_L^2 + w_L^2 + u_R^2 + v_R^2 + w_R^2}{2}}\right] \tag{4.69}$$

where $c_{1/2}$ is the averaged speed of sound defined as follows:

$$c_{1/2} = \frac{1}{2}\left(c_L + c_R\right) \tag{4.70}$$

Note that thanks to this choice of the Mach number function, ζ, the AUFSR+ scheme is free from any cut-off reference Mach number, similar to the SLAU scheme.

Caution must be be taken when shock waves are present. In the work by Oßwald *et al* [34], it was noted that at the vicinity of shocks small disturbances were observed when using a scaling Mach number function. To circumvent this anomaly, Owald *et al* suggested to use a shock indicator such that at the vicinity of a shock the Mach number function receives the value of unity. Namely, near a shock there is no scaling of the normal velocity jump. Following Owald *et al*, the Mach number function is therefore modified (denoted as ζ^*) as follow

$$\zeta^* = \begin{cases} 1, & \text{if } s_2 = 0 \\ \zeta, & \text{otherwise} \end{cases} \tag{4.71}$$

Finally, the dissipation part of the AUFSR+ scheme can now be written as follows:

$$\mathbf{D}_{AUFSR+} = \underbrace{\frac{\zeta^*}{2}\left[\left(q^L - \tilde{q}\right)\mathbf{U}^L + \left(q^R - \tilde{q}\right)\mathbf{U}^R\right]}_{\mathbf{D}_{AUFS}} - \underbrace{\frac{\tilde{c}}{2}\left(\eta_1 \widetilde{\mathbf{R}}_1 + \eta_5 \widetilde{\mathbf{R}}_5\right)}_{\mathbf{D}_{Roe}} \tag{4.72}$$

with

$$\eta_1 = \frac{\Delta p - \zeta^* \tilde{\rho}\tilde{c}\Delta q}{2\tilde{c}^2}, \quad \eta_5 = \frac{\Delta p + \zeta^* \tilde{\rho}\tilde{c}\Delta q}{2\tilde{c}^2} \tag{4.73}$$

Note that at the lower limit of zero Mach number, the dissipation part of the $\mathbf{D}_{AUFSR+}$ that originates from the AUFS scheme (labeled as $\mathbf{D}_{AUFS}$ in Equation (4.72)), is vanished. Hence, at the lower limit of zero Mach number, the dissipation part of the AUFSR+ scheme behaves as the dissipation part of the low-Mach Roe scheme, labeled as $\mathbf{D}_{Roe}$, with the shifted eigenvalues by the artificial wave speed s_1.

4.2.5 Passive Scalar Approach

The EZAir suite of codes implements the passive scalar approach [18] in spatial discretization of the various model equations (*e.g.*, turbulence, finite-rate chemistry, etc.). The passive scalar approach enables to treat the extended governing equation set (including the model equations) in a similar manner to that presented for the Navier-Stokes equations. For example, when using any of the $k - \omega$ models with the

HLLC scheme, the left and right state vectors are extended as follows:

$$\mathbf{Q}_l^* = \begin{Bmatrix} \rho_l^* \\ (\rho u)_l^* \\ (\rho v)_l^* \\ (\rho w)_l^* \\ e_l^* \\ (\rho k)^* \\ (\rho\phi)^* \end{Bmatrix} = \Omega_l \begin{Bmatrix} \rho_l\left(S_l - q_l\right) \\ \left(S_l - q_l\right)(\rho u)_l + \left(p^* - p_l\right) n_x \\ \left(S_l - q_l\right)(\rho v)_l + \left(p^* - p_l\right) n_y \\ \left(S_l - q_l\right)(\rho w)_l + \left(p^* - p_l\right) n_z \\ \left(S_l - q_l\right) e_l - p_l q_l + p^* S_m \\ \rho k \\ \rho\phi \end{Bmatrix} \tag{4.74}$$

The HLLC inviscid flux is then easily evaluated using:

$$\mathbf{H}_{c_l}^* \equiv \mathbf{H}_c\left(\mathbf{Q}_l^*\right) = \mathbf{Q}_l^* S_m + \begin{Bmatrix} 0 \\ p^* n_x \\ p^* n_y \\ p^* n_z \\ p^* S_m \\ 0 \\ 0 \end{Bmatrix} \tag{4.75}$$

4.2.6 Choosing the Flux Approximation

As mentioned above, the EZAir suite of codes allows the user to choose the flux approximation to be used for the inviscid terms. This is conducted through the parameter *convrhs*. The reader is referred to Table 10.187 for the details.

4.2.7 Monotone Upstream-Centered Scheme for Conservation Laws (MUSCL)

For a first-order-accurate approximation, the left and right state vectors are simply calculated from cell-center values left and right of the interface, respectively. A higher order of accuracy is achieved by applying a third-order, upwind-biased MUSCL interpolation method [35, 36] to any variables vector $\mathcal{W}$. The vector $\mathcal{W}$ can be for example the mean flow primitive variables vector (as implemented in the single-component, perfect gas (SPG) physical model):

$$\mathcal{W} = \begin{bmatrix} \rho \\ u \\ v \\ w \\ p \end{bmatrix} \tag{4.76}$$

The interpolated values of the variables vector on the left and right sides of face ij are given by:

$$\begin{aligned} \mathcal{W}_L &= \mathcal{W}_i + \frac{\phi\left(r_L\right)}{4}\left[(1+\kappa)\bigtriangleup + (1-\kappa)\bigtriangledown\right]\mathcal{W}_i \\ \mathcal{W}_R &= \mathcal{W}_j - \frac{\phi\left(r_R\right)}{4}\left[(1+\kappa)\bigtriangledown + (1-\kappa)\bigtriangleup\right]\mathcal{W}_j \end{aligned} \tag{4.77}$$

where $\kappa = 1/3$ is used to yield a third-order, upwind-biased, scheme. The difference operators $\bigtriangleup, \bigtriangledown$ operating on the variables vector at a given cell k are defined as follows:

$$\begin{aligned} \bigtriangleup\mathcal{W}_k &= \mathcal{W}_{k+1} - \mathcal{W}_k \\ \bigtriangledown\mathcal{W}_k &= \mathcal{W}_k - \mathcal{W}_{k-1} \end{aligned} \tag{4.78}$$

where indices $k+1$ and $k-1$ indicate neighboring cells located right and left of cell k in the direction perpendicular to face ij, respectively. In the interest of suppressing spurious oscillations in the solution, a limiting function, ϕ, is introduced into Equation (4.77). The default limiter in the EZAir suite is the van-Albada limiter [37], given by:

$$\phi\left(r\right) = \frac{r^2 + r}{1 + r^2} \tag{4.79}$$

where $r_L = \dfrac{\bigtriangleup\mathcal{W}_i}{\bigtriangledown\mathcal{W}_i}$ and $r_R = \dfrac{\bigtriangledown\mathcal{W}_j}{\bigtriangleup\mathcal{W}_j}$ are the ratios of successive gradients of the variables vector around cells i and j, respectively.

4.2.8 High Order Flux Approximation Schemes

Two high order inviscid flux approximation schemes are implemented in the EZAir suite of codes. The first is a high order monotonicity preserving scheme, also known as the monotonicity-preserving (MP) high order scheme developed by Suresh and Huynh [38]. The second is a 5th order WENO interpolation scheme [39].

4.2.8.1 Monotonicity-Preserving (MP) Reconstruction

Let q be a flow variable and let $q^L_{i+1/2}$ be the left face approximation of the variable at $i+1/2$, the 5^{th} order monotonicity-preserving (MP) reconstruction can be written as [38]:

$$q^L_{i+1/2} = \frac{1}{60}\left(2q_{i-2} - 13q_{i-1} + 47q_i + 27q_{i+1} - 3q_{i-2}\right) \tag{4.80}$$

The above reconstruction may result in oscillations near a discontinuity. To suppress these oscillations, a limiter is applied. The necessity of applying the limiter is determined through the following criterion:

$$\left(q^L_{i+1/2} - q_i\right)\left(q^L_{i+1/2} - q^{MP}\right) \leq \epsilon \tag{4.81}$$

where the default is $\epsilon = 10^{-10}$ and

$$q^{MP} = q_i + \text{minmod}\left[\left(q_{i+1} - q_i\right), 4\left(q_i - q_{i-1}\right)\right] \tag{4.82}$$

If the condition is satisfied, then no limiting is required, and therefore the interface value is set as $q_{i+1/2} = q^L_{i+1/2}$. Otherwise, a special accuracy-preserving limiter is applied. Details of this limiter may be found in the original publication [38].

The 5^{th} order MP scheme may be selected as described in Tables 10.90 and 10.178. In addition, an equivalent 4^{th} order MP scheme may be selected as described in Tables 10.89 and 10.176. Furthermore, the MP reconstruction scheme may either be applied directly to the primitive, the conservative, or the characteristic variables. A slight advantage toward characteristics-based reconstruction was identified by [40]. However, a general recommendation may not be given, and the user is advised to compare the performance of the various available schemes for each case.

4.2.8.2 Weighted Essentially Non-Oscillatory (WENO) Reconstruction Scheme

Let q be a flow variable and let $q^L_{i+1/2}$ be the left face approximation of the variable at $i+1/2$, the weighted essentially non-oscillatory (WENO) reconstruction scheme can be written as [39]:

$$q^L_{i+1/2} = w_0 q_0 + w_1 q_1 + w_2 q_2 \tag{4.83}$$

where

$$\begin{aligned} q_0 &= \frac{1}{3}q_{i-2} - \frac{7}{6}q_{i-1} + \frac{11}{6}q_i \\ q_1 &= -\frac{1}{6}q_{i-1} - \frac{5}{6}q_i + \frac{1}{3}q_{i+1} \\ q_2 &= \frac{1}{3}q_i + \frac{5}{6}q_{i+1} - \frac{1}{6}q_{i+2} \end{aligned} \tag{4.84}$$

The weights appearing in Equation 4.84 are defined as follows:

$$\begin{aligned} w_s &= \frac{\alpha_s}{\alpha_0 + ... + \alpha_{r-1}} \\ \alpha_s &= \frac{C_s}{\epsilon + IS_s} \end{aligned} \tag{4.85}$$

where

$$\begin{aligned} C_0 &= 0.1 \\ C_1 &= 0.6 \\ C_2 &= 0.3 \end{aligned} \tag{4.86}$$

and the smoothness indicators IS_s are given by

$$
\begin{aligned}
IS_0 &= \frac{13}{12}\left(q_{i-2} - 2q_{i-1} + q_i\right)^2 + \frac{1}{4}\left(q_{i-2} - 4q_{i-1} + 3q_i\right)^2 \\
IS_1 &= \frac{13}{12}\left(q_{i-1} - 2q_i + q_{i+1}\right)^2 + \frac{1}{4}\left(q_{i-1} - q_{i+1}\right)^2 \\
IS_2 &= \frac{13}{12}\left(q_i - 2q_{i+1} + q_{i+1}\right)^2 + \frac{1}{4}\left(3q_i - 4q_{i+1} + q_{i+1}\right)^2
\end{aligned}
\tag{4.87}
$$

(4.88)

where $\epsilon = 10^{-6}$ by default. Shen *et al* [41] have observed that when there are shock waves or other large flow gradients present in the flowfield, the weights of the original WENO scheme will necessarily oscillate. In such cases, the convergence of the numerical solution may be seriously degraded. In order to avoid this, it is recommended to set ϵ in the range $10^{-4} - 10^{-2}$ to enable better convergence in transonic flows, with minimal impact on accuracy. A non-default value for ϵ may be specified as detailed in Table 10.181.

The 5^{th} order WENO scheme may be selected as described in Tables 10.91 and 10.180. Furthermore, the WENO reconstruction scheme may either be applied directly to the primitive, the conservative, or the characteristic variables. A slight advantage toward characteristics-based reconstruction was identified by [40]. However, a general recommendation may not be given, and the user is advised to compare the performance of the various available schemes for each case.

4.3 Hybrid Convection Flux Scheme

In hybrid flow simulations, specifically in the LES regions, the subgrid-scale (SGS) model has to provide an adequate amount of dissipation, such that the turbulence energy cascade is correctly reproduced and no energy accumulates at the smallest resolved scales. The amount of dissipation provided by the SGS model may become very small with the consequence that any elevated level of numerical dissipation will inevitably contaminate the resolved flow field. Therefore, higher-order central difference schemes with low levels of numerical dissipation are the preferred choice in these regions. However, the use of central difference schemes without any form of artificial dissipation may result in unwanted oscillations and often may hinder the overall stability of the code and the flow simulations.

On the other hand, in general, higher-order upwind or TVD schemes are not recommended for turbulence resolving simulations, since the additional numerical dissipation introduced by these schemes is significant, resulting in a contamination of the resolved flow field. In RANS applications, the source terms and high-order nonlinearities present in the transport equations for turbulence quantities require more robust, yet less accurate, upwind or TVD schemes to ensure numerical stability. The increased levels of numerical dissipation induced by the second-order upwind or TVD schemes are usually smaller than the large diffusivity provided by the RANS model and hence, does not significantly affect the flow predictions.

A robust unified hybrid RANS/LES approach may benefit from a hybrid convection scheme that satisfies the opposed requirements for RANS regions as well as LES regions. A hybrid convection scheme may be obtained by a linear combination of the central-difference scheme and a second-order (or higher) upwind scheme, using a flow-dependent flux-blending function. The EZAir suite of codes provides the choice to use such a hybrid convection scheme. The scheme has the following form:

$$H_c = (1 - \sigma_b)\, H_{c_{CD}} + \sigma_b H_{c_{UP}} \tag{4.89}$$

where H_c is the inviscid flux vector as given in Equation (4.2), $H_{c_{CD}}$ represents the central differencing inviscid flux approximation, and $H_{c_{UP}}$ represents the upwind inviscid flux approximation. The term σ_b is a flow-dependent blending function. Note that it is recommended to use the hybrid convection flux scheme only if a hybrid turbulence model (as described in Section 3.5) is utilized. The hybrid convection flux scheme may be selected as described in Table 10.185.

4.3.1 Hybrid Convection Flux Blending Functions

The EZAir suite of codes provides 5 different blending functions. The following sections describe the blending functions.

4.3.1.1 Detached Eddy Simulation Based Blending Function

The detached eddy simulation blending function works only with the DDES (see Section 3.5.3.1) and IDDES (see Section 3.5.3.2) models. The blending functions in this case are identical to the blending functions that are used for the hybrid turbulence models. Specifically, when using the DDES hybrid model, the function f_d, defined in Equation (3.79)) (see Section 3.5.3.1), is used whereas the function $\tilde{f}_d$, defined in Equation (3.82)), is used when using the IDDES model (see Section 3.5.3.2). Finally, a lower limit is applied and therefore the function becomes:

$$\sigma_b = \max\left(\left\{f_d, \tilde{f}_d\right\}, \sigma_0\right) \tag{4.90}$$

The lower limit σ_0 can be set as described in Table 10.185.

4.3.1.2 Ducros Blending Function

The Ducros blending function is given as:

$$\sigma_{\text{ducros}} = \frac{\left(\nabla \cdot \mathbf{u}\right)^2}{\left(\nabla \cdot \mathbf{u}\right)^2 + \left(\nabla \times \mathbf{u}\right)^2 + \epsilon^2} \tag{4.91}$$

where $\mathbf{u}$ is the velocity vector and ϵ is evaluated using:

$$\epsilon = \frac{1 \times 10^{-8} U_\infty}{\Delta_{gl}} \tag{4.92}$$

The free stream velocity is given by U_∞ and Δ_{gl} is evaluated as described either in Section 3.5.1.1 or Section 3.5.1.2 only. Note that Δ_{gl} may be set as a part of the hybrid model options (see Tables 10.156, 10.158, 10.159, or , 10.160. Finally, a lower limit is applied and therefore the function becomes:

$$\sigma_b = \max\left(\sigma_{\text{ducros}}, \sigma_0\right) \tag{4.93}$$

The lower limit σ_0 can be set as described in Table 10.185.

4.3.1.3 Larsson Blending Function

The Larsson blending function is defined as follows. Let θ be

$$\theta = \frac{\nabla \cdot \mathbf{u}}{\max\left[C_1 \left|\nabla \times \mathbf{u}\right|, C_2 \left(c\sqrt[3]{Vol}\right)\right]} \tag{4.94}$$

where $C_1 = 5$, $C_2 = 0.05$, and c is the speed of sound. The Larsson blending function is therefore

$$\sigma_{\text{larsson}} = \begin{cases} 0 & \theta \leq 1 \\ 1 & \theta > 1 \end{cases} \tag{4.95}$$

Finally, a lower limit is applied and therefore the function becomes:

$$\sigma_b = \max\left(\sigma_{\text{larsson}}, \sigma_0\right) \tag{4.96}$$

The lower limit σ_0 can be set as described in Table 10.185.

4.3.1.4 Travin Blending Function

The Travin blending function is quite complex and its details are far beyond the scope of the current manual. A brief description is brought herein. Let A be defined as:

$$A = \max\left(\frac{C_{DES}\ \Delta_{gl}}{L_{turb}\ g} - 0.5, 0\right) \tag{4.97}$$

where $C_{CDES} = 0.65$ and L_{turb} and g are functions of the velocity vector derivatives (such as the rate of strain tensor etc.) and various time scales. For complete details of the functions the user is referred to the work by Xiao *et al* [42]. The σ_{travin} blending function is given by:

$$\sigma_{\text{travin}} = \tanh\left(A^3\right) \tag{4.98}$$

The grid length scale, Δ_{gl}, is evaluated as described either in Section 3.5.1.1 or Section 3.5.1.2 only. Note that Δ_{gl} may be set as a part of the hybrid model options (see Tables 10.156, 10.158, 10.159, or , 10.160. Finally, a lower limit is applied and therefore the function becomes:

$$\sigma_b = \max\left(\sigma_{\text{travin}}, \sigma_0\right) \tag{4.99}$$

The lower limit σ_0 can be set as described in Table 10.185.

4.3.1.5 Flow Simulation Methodology Based Blending Function

The flow simulation methodology (FSM) blending function is given as follows. Let A be defined as:

$$A = \max\left(f_\eta f_\Delta f_z - 0.6, 0\right) \tag{4.100}$$

where

$$f_\eta = \left[1 - \exp\left(\frac{-\beta\Delta_{gl}}{L_\eta}\right)\right]^n \tag{4.101}$$

with $\beta = 0.001$, $n = 1$, and L_η is given by:

$$L_\eta = \left(\frac{\nu^3}{\beta_k k\omega}\right)^{1/4} \tag{4.102}$$

where $\beta_k = 0.09$, k is the turbulent kinetic energy, and ω is the specific dissipation rate. The function f_Δ is given by:

$$f_\Delta = \frac{C_\alpha \Delta_{gl}}{L_T} \tag{4.103}$$

where $C_\alpha = 0.85$ and

$$L_T = \frac{\sqrt{k}}{\beta_k \omega} \tag{4.104}$$

The function f_z is given by:

$$f_z = \frac{1}{1 - F_z} \tag{4.105}$$

where F_z is the F_1 function by Menter, given in Equation (3.46). Then, the σ_{fsm} blending function is given by:

$$\sigma_{\text{fsm}} = \tanh\left(A^3\right) \tag{4.106}$$

The grid length scale, Δ_{gl}, is evaluated as described either in Section 3.5.1.1 or Section 3.5.1.2 only. Note that Δ_{gl} may be set as a part of the hybrid model options (see Tables 10.156, 10.158, 10.159, or , 10.160. Finally, a lower limit is applied and therefore the function becomes:

$$\sigma_b = \max\left(\sigma_{\mathrm{fsm}}, \sigma_0\right) \tag{4.107}$$

The lower limit σ_0 can be set as described in Table 10.185.

4.4 Diffusive Flux Vector Discretization

The diffusive flux vector normal to the interface, $\mathcal{H}_d$, is a function of the primitive variables vector, $\mathcal{W}$, evaluated at the mid-point of face ij, $\mathcal{W}_{ij}$, and of its derivatives, $\frac{\partial \mathcal{W}_{ij}}{\partial x}$, $\frac{\partial \mathcal{W}_{ij}}{\partial y}$ (in two dimensions). The vector $\mathcal{W}_{ij}$ is calculated by averaging of adjacent cell-center values (*i.e.*, $\mathcal{W}_i$ and $\mathcal{W}_j$). The evaluation of the diffusive flux vector is brought here in two dimensions. The extension to three-dimensions is straightforward.

4.4.1 Full-Viscous Approximation of Face Gradients

The partial derivatives of $\mathcal{W}_{ij}$ are calculated by employing second-order central differencing in all directions.

4.4.2 Thin-Layer Approximation of Face Gradients

Another technique employed by the EZAir suite of codes is the approximation of the partial derivatives of $\mathcal{W}_{ij}$ based on the thin-layer assumption, wherein only viscous diffusion in the direction normal to the body (and flow) is accounted for. This assumption is generally valid for high Reynolds flows without major flow separation. By locally neglecting diffusion in the direction tangent to the interface, the partial derivatives of $\mathcal{W}_{ij}$ may be approximated as follows:

$$\frac{\partial \mathcal{W}_{ij}}{\partial x} = \frac{(\mathcal{W}_j - \mathcal{W}_i)\, n_x}{(x_j - x_i)\, n_x + (y_j - y_i)\, n_y} \tag{4.108}$$

$$\frac{\partial \mathcal{W}_{ij}}{\partial y} = \frac{(\mathcal{W}_j - \mathcal{W}_i)\, n_y}{(x_j - x_i)\, n_x + (y_j - y_i)\, n_y} \tag{4.109}$$

4.5 Special Jacobians for Implicit Time Marching

The EZAir suite of codes provides the capability to use various flux Jacobians. The first one is the Jacobian based on the HLLC approximate Riemann solver. Details may be found in the work by Batten *et al* [18]. There are two flux Jacobians that are based on flux vector splitting schemes. The first are flux Jacobians based on the Van Leer flux [43], and the second are Jacobians that are based on the Steger-Warming flux vector splitting scheme [44], briefly described in the next section. In addition, three variants of the MAPS scheme by Rossow [45] are available. The reader is referred to Table 10.190 for detailed instructions on convective flux Jacobian selection.

4.5.1 Steger-Warming Flux Vector Splitting Jacobians

The Steger-Warming scheme uses the homogeneous property of the inviscid flux vector, namely $E_c(\alpha Q) = \alpha E_c(Q)$, resulting in:

$$E_c(Q) = \frac{\partial E_c}{\partial Q} Q \equiv AQ \tag{4.110}$$

where A denotes the *Jacobian matrix* of the inviscid flux vector. The matrix A may be diagonalized, so that $A = L\Lambda R$, where L, R are the corresponding left- and right-eigenvectors of A, and Λ is a diagonal matrix composed of the eigenvalues of A. To produce an upwind-biased scheme, the inviscid flux is split into upstream (positive) and downstream (negative) contributions as follows:

$$E_{c,SW} = E_c^+ + E_c^- = A_l^+ Q_l + A_r^- Q_r \tag{4.111}$$

where l and r denote values from cells left and right of an interface, and $A_l^+ = (L\Lambda^+ R)_l$ and $A_r^- = (L\Lambda^- R)_r$. The positive and negative parts of the *n-th* eigenvalue of A are given by:

$$\lambda_n^{\pm} = \frac{1}{2}\left(\lambda_n \pm |\lambda_n|\right) \tag{4.112}$$

The discretization of the convective flux using the Steger-Warming scheme may yield non-physical solutions, specifically such that violate the second law of thermodynamics, *i.e.*, result in an impossible increase in entropy [46]. One way that such non-physical phenomena can be avoided is by redefining the eigenvalues in the following manner [47]:

$$\lambda_n^{\pm} = \frac{1}{2}\left(\lambda_n \pm \sqrt{\left(\lambda_n\right)^2 + \left(\epsilon a\right)^2}\right) \tag{4.113}$$

where ϵ is a constant typically set to 0.1-0.2, and a is the local speed of sound. The entropy correction is applied to all the eigenvalues, even though only the acoustic waves need to be corrected to prevent non-physical solutions. Applying the entropy correction to all the eigenvalues does not affect the accuracy of the solution and allows to use larger time steps in the pseudo-time integration of the equations [47].

The Steger-Warming Jacobian are highly recommended for use in supersonic flows, and in other, "numerically stiff" problems where the increased dissipation introduced by the Steger-Warming Jacobians may enhance overall robustness, without compromising accuracy. The selection of the flux Jacobians is detailed in Table 10.190.

4.6 Time Marching Schemes

The EZAir suite of codes provides a large number of possibilities for advancing Equation (4.3) in time. This section contains a brief description of the available schemes. Note that all of these schemes may be used for either time accurate simulations or for convergence to steady state. However, for the most part, all of the schemes described in this section are used for steady state simulations only. The schemes may be classified as follows:

1. Explicit (single and multi stage) schemes

   (a) Explicit Euler
   (b) Third and fourth order Runge-Kutta schemes

2. Implicit schemes
   (a) Point Gauss-Seidel
   (b) Line Gauss-Seidel
   (c) Alternate direction implicit (ADI)
   (d) Diagonally-dominant alternate direction implicit (DDADI), with or without sub-iterations.

3. Multi-stage implicit schemes
   (a) B2 two stage time integration scheme

4.6.1 The Time Step

In what follows, two types of simulations are considered. The first is non-time-accurate flow simulations. These are used when the flow is expected to reach a steady state. The second type is time-accurate flow simulations, for unsteady flows. The time step, Δt, is set to be constant for the whole flow field for time-accurate flow simulations and may locally vary for non-time-accurate flow simulations. In which case, the local pseudo-time step, $\Delta\tau$, is calculated as the minimum of all cell face time step, that is evaluated from a CFL condition:

$$\Delta\tau_{ij} = CFL\frac{L_{ij}}{max(\lambda_c, \lambda_d)_{j\in\mathfrak{M}(i)}} \tag{4.114}$$

where λ_c and λ_d are the maximum eigenvalues of the convective and diffusive flux Jacobians, respectively, defined as:

$$(\lambda_c)_j = \left[\left(\left|q_i\right|_j + a_i\right)\right] \tag{4.115}$$

$$(\lambda_d)_j = \left[\left(\frac{4\left(\bar{\mu} + \bar{\mu}_t\right)_i}{3\bar{\rho}_i L_{ij}}\right)\right] \tag{4.116}$$

where a_i is the speed of sound of cell i, and $|q_i|_j$ is the absolute velocity of cell i in normal direction to the cell interface ij. The term L_{ij} is a typical cell length scale, that is usually calculated as $L_{ij} = V_i/A_{ij}$ where V_i is the cell volume and A_{ij} is the face area. In the following sections, with the exception of dual time-stepping, the physical time step, Δt, may be replaced with the pseudo time step, $\Delta\tau$.

4.6.2 Explicit schemes

4.6.2.1 Explicit Euler Scheme

Consider the semi-discrete equation[1]:

$$V\frac{dQ}{dt} = R \tag{4.117}$$

A simple, first-order Euler explicit time marching scheme is given by:

$$\Delta Q^n = \frac{\Delta t}{V} R^n \tag{4.118}$$

where ΔQ^n is the increment of the solution between time levels , namely,

$$\Delta Q^n = Q^{n+1} - Q^n \tag{4.119}$$

4.6.2.2 Runge-Kutta Schemes

EZAir provides the choice of third or fourth order Runge-Kutta schemes. Consider the semi-discrete formulation as presented in Equation (4.117), the Runge-Kutta scheme formulation is as follows:

$$\begin{aligned} Q^{(0)} &= Q^n \\ Q^{(k)} &= Q^n + \alpha_k \frac{\Delta t}{V} R^{(k-1)}, \quad k = 1, \ldots, K \\ Q^{n+1} &= Q^{(K)} \end{aligned} \tag{4.120}$$

[1] Equation (4.3) without the index

where k is the Runge-Kutta sub-step number, $K = 3$ for third order and $K = 4$ for fourth order, and α_k are the appropriate weights.

4.6.3 Implicit Time Marching Formulation

The fine grid spacing required to resolve the normal viscous terms close to the body surface requires in turn very small time steps and therefore it rules out the use of explicit methods. In explicit time-marching schemes the maximum time step is proportional to the minimum grid spacing. As a result the time-step limit imposed by stability is very small. In contrast, even though the operation count per time step is high, it is more efficient to use implicit methods. The development of a non-iterative implicit algorithm for the solution of the Navier-Stokes equations requires a time linearization of the nonlinear vectors (R). The linearization procedure is simple since the equations are written in conservation-law form. Applying the first order Euler implicit method and utilizing the Delta form of the equations results in the following implicit scheme:

$$\left(\frac{V}{\Delta t} I - \frac{\partial R}{\partial Q}\right)^n \Delta Q^n = R^n \tag{4.121}$$

where R^n is the residual at time level n as define by Equation (4.3), I is the identity matrix, Δt is the time increment between levels n and $n+1$, the term $\frac{\partial R}{\partial Q}$ is the Jacobian matrix. Note that the Jacobian matrix is first order and that it may be altered to improve stability. The type of Jacobians may be set using Table 10.190.

Applying Equation (4.121) at every grid point results in a block-hepta-diagonal matrix in three dimensions. The inversion of the matrix, or its approximation, may be conducted in various manners, resulting in a wide variety of implicit time marching schemes.

4.6.3.1 Point Gauss-Seidel

The exact, first order Jacobian matrix is retained only for the diagonal elements and the off diagonal Jacobian matrices are linearized based on the previous time step as follows:

$$\frac{\partial R}{\partial Q}\Delta Q \approx \left(\frac{\partial R}{\partial Q}\right)^n \Delta Q^n \tag{4.122}$$

Consequently, they can be moved to the right hand side. The resulting PGS (point Gauss-Seidel) scheme is relatively fast but is limited in terms of CFL.

4.6.3.2 Alternating Line Gauss-Seidel

The exact Jacobian matrices are expanded along coordinate lines (ξ, η, and ζ of the computational mesh). Line inversions are conducted along all coordinate lines. All terms along the transverse coordinate lines are moved to the right hand side (in a similar fashion to the PGS scheme). The resulting scheme LGS (line Gauss-Seidel) is much more robust, however it is much slower.

4.6.3.3 ADI and DDADI Schemes

The left hand side block matrix is approximately factored into three one dimensional operators (each of the operators is represented by a block tridiagonal matrix). Inversions are conducted in each coordinate direction and the resulting ADI (alternate direction implicit) scheme is robust, allowing the use of moderate to high CFL numbers, and it is faster than the LGS scheme. However, this comes with the cost of a factorization error that limits the CFL numbers. The DDADI (diagonally dominant alternate direction implicit) scheme is constructed in order to reduce factorization errors. The DDADI scheme allows the use of CFL numbers that are similar (or even higher) to the numbers used by the LGS scheme at a generally reduced cost.

4.6.4 B2 Two Stage Time Integration Scheme

The $B2$ time integration method is a two step scheme that can be utilized in conjunction with any of the above implicit methods. It's formulation is as follows. Consider an Euler implicit integration scheme and an appropriate time marching scheme. Define the single step correction as:

$$\Delta Q^n \quad = \quad B1\left(Q^n, \Delta t\right) \tag{4.123}$$

The $B2$ two-step method is therefore given by:

$$\begin{aligned} \Delta Q^{*^n} &= B1\left(Q^n, \frac{\Delta t}{2}\right) \\ \Delta Q^n &= \frac{1}{2}B1\left(Q^{*^n}, \Delta t\right) \end{aligned} \tag{4.124}$$

Note that the $B2$ method should only be used for non-time-accurate time marching purposes, *i.e.*, for convergence to a steady-state solution. In which case, the pseudo time step, $\Delta\tau$, is used instead of Δt.

4.6.5 Selection of Time Marching Scheme

The time marching scheme is controlled by the user using the "-solve" directive and the "scheme" parameter. Table 10.171 contains the instruction as to the time-marching scheme selection.

4.7 Dual Time Step Formulation

Time accurate flow simulations require a higher order temporal accuracy as well as means to decrease linearization and factorization errors. A second-order dual-time-stepping formulation may be obtained by adding a pseudo time derivative to Equation (4.3) as follows:

$$V\frac{dQ}{d\tau} + V\frac{dQ}{dt} = R \tag{4.125}$$

By applying time linearization to the residual vector, R, and discretizing the real and pseudo time derivatives, a second order accurate scheme is obtained:

$$\frac{V}{\Delta\tau}\left(Q^{k+1} - Q^k\right) + \frac{V}{2\Delta t}\left(3Q^{k+1} - 4Q^n + Q^{n-1}\right) - \left(\frac{\partial R}{\partial Q}\right)^k \Delta Q^k = R^k \tag{4.126}$$

In the dual time step formulation, the superscript n is the time step while the superscript k signifies the sub-iteration. By defining the modified residual such that it contains the physical time derivatives as:

$$R^{*^k} = -\frac{V}{2\Delta t}\left(3Q^k - 4Q^n + Q^{n-1}\right) + R^k \tag{4.127}$$

the following steady-state-like iterative scheme is obtained:

$$\left[\left(\frac{V}{\Delta\tau} + \frac{3V}{2\Delta t}\right) I - \frac{\partial R}{\partial Q}\right]^k \Delta Q^k = R^{*^k} \tag{4.128}$$

The sub-iteration is considered converged if the modified residual, R^{*^k}, has dropped several orders in magnitude (case dependent). At that point, $\Delta\hat{Q}^k \to 0$ and therefore $\hat{Q}^{k+1} \to \hat{Q}^k \to \hat{Q}^{n+1}$. Hence the solution has converged into its second-order accurate solution.

The importance of the dual time formulation is increased when conducting store separation and aeroelastic simulations. Since the geometry changes require that the suite of Chimera routines for hole cutting and interpolation point search is invoked for each time step, using a first-order time-accurate simulation, with a much smaller time step, becomes highly inefficient. The dual time step formulation allows the use of much larger time steps and therefore the overhead of the Chimera routines and other routines, such as cell volume and face area calculations, is greatly reduced. This is true, even when a rather large number of sub-iterations is required for each time step to converge. Temporal discretization is set by the user as described in Tables 10.15 and 10.16.

4.8 Preconditioned Time Marching Formulation

At low Mach numbers, the compressible mean-flow equations may be numerically very stiff. Therefore, it is desired to construct the system of equations eigensystem that is independent of the low Mach conditions. These considerations give rise to the preconditioned system of Equation (4.128). Let the matrix Γ be the transformation matrix defined as:

$$\Gamma = \frac{\partial Q}{\partial Q_p} \tag{4.129}$$

where the vector Q_p is the primitive vector of the mean flow equations, defined as:

$$Q_p = \{p, u, v, w, T\} \tag{4.130}$$

and

$$\Delta Q = \Gamma \Delta Q_p \tag{4.131}$$

Therefore, Equation (4.128) can be written as:

$$\left[\frac{V}{\Delta\tau}\Gamma + \frac{3V}{2\Delta t}\Gamma - \frac{\partial R}{\partial Q}\Gamma\right]^k \Delta Q_p^k = R^{*\,k} \tag{4.132}$$

To alleviate the numerical stiffness, the transformation matrix that multiplies the artificial time derivative is modified to become a preconditioned matrix based on the Weiss and Smith matrix [48], denoted by Γ_p. Moreover, a consistent preconditioned system requires the modification of the eigenvalues of the convective part. Let the vector $\widetilde{R}$ denote the appropriate low-Mach residual operator (the convective flux is designed to handle low Mach number flows) then the preconditioned mean-flow discrete equations become

$$\left[\frac{V}{\Delta\tau}\Gamma_p + \frac{3V}{2\Delta t}\Gamma - \frac{\partial \widetilde{R}}{\partial Q_p}\right]^k \Delta Q_p^k = \widetilde{R}^{*\,k} \tag{4.133}$$

where

$$\widetilde{R}^{*\,k} = \widetilde{R}^{k} - \frac{V}{2\Delta t}\left[3\left(Q^k - Q^n\right) - \left(Q^n - Q^{n-1}\right)\right] \tag{4.134}$$

The matrix Γ is given by:

$$\Gamma = \frac{\partial Q}{\partial Q_p} = \begin{bmatrix} \frac{\partial \rho}{\partial p} & 0 & 0 & 0 & \frac{\partial \rho}{\partial T} \\ u\frac{\partial \rho}{\partial p} & \rho & 0 & 0 & u\frac{\partial \rho}{\partial T} \\ v\frac{\partial \rho}{\partial p} & 0 & \rho & 0 & v\frac{\partial \rho}{\partial T} \\ w\frac{\partial \rho}{\partial p} & 0 & 0 & \rho & w\frac{\partial \rho}{\partial T} \\ \frac{\partial \rho h_t}{\partial p} - 1 & \rho u & \rho v & \rho w & \frac{\partial \rho h_t}{\partial T} \end{bmatrix} \tag{4.135}$$

where h_t is the total enthalpy. The matrix Γ_p is given by:

$$\Gamma_p = \begin{bmatrix} \frac{\partial \rho'}{\partial p} & 0 & 0 & 0 & \frac{\partial \rho}{\partial T} \\ u\frac{\partial \rho'}{\partial p} & \rho & 0 & 0 & u\frac{\partial \rho}{\partial T} \\ v\frac{\partial \rho'}{\partial p} & 0 & \rho & 0 & v\frac{\partial \rho}{\partial T} \\ w\frac{\partial \rho'}{\partial p} & 0 & 0 & \rho & w\frac{\partial \rho}{\partial T} \\ \frac{\partial \rho' h_t}{\partial p} - 1 & \rho u & \rho v & \rho w & \frac{\partial \rho h_t}{\partial T} \end{bmatrix} \tag{4.136}$$

The pseudo-properties derivatives are given as follows:

$$\frac{\partial \rho'}{\partial p} = \frac{1}{M_p^2}\frac{\partial \rho}{\partial p} \tag{4.137}$$

where M_p is the preconditioned Mach number given by

$$M_p = \frac{V_{\text{ref}}}{c} \tag{4.138}$$

and V_{ref} is given in Equation (4.144) below. The eigenvalues of the preconditioned system are given by

$$\lambda(\Gamma_p^{-1}\frac{\partial E_c}{\partial Q_p}) = \{q' - c', q', q', q', q' + c'\} \tag{4.139}$$

where

$$q' = \frac{q}{2}(1 + \beta_p V_{\text{ref}}^2) \tag{4.140}$$

$$c' = \sqrt{\alpha_p^2 q^2 + V_{\text{ref}}^2} \tag{4.141}$$

and

$$\beta_p = \frac{\partial \rho}{\partial p} + \frac{1}{\rho}\frac{\partial \rho}{\partial h} \tag{4.142}$$

$$\alpha_p = \frac{1}{2}(1 - \beta_p V_{\text{ref}}^2) \tag{4.143}$$

Here, the reference velocity, V_{ref}, is defined as

$$V_{\text{ref}} = \min[c, \max(V_{\max}, k_{vr}\ V_\infty)] \tag{4.144}$$

where $V_{\max}$ is the maximum of the velocity magnitude among the local cell and its neighbors and the default value for k_{vr} is $k_{vr} = \sqrt{3}$. For details on how to turn on preconditioning and modifying the reference velocity constant, k_{vr}, the user is referred to Table 10.174 in Chapter 10 .

4.9 Multigrid Convergence Acceleration Method

Turbulent flow simulations employing the Reynolds-Averaged Navier Stokes (RANS) equations may present serious numerical difficulties, including convergence and positivity preservation difficulties. The motivation at hand is accelerating numerical methods for solving the mean-flow and the turbulence model equations, in cases where conventional methods are not optimal. One of the fastest acceleration means is the multigrid class of methods. Multigrid methods accelerate convergence rates of numerical schemes by using a hierarchy of grids, based on the notion that certain numerical error modes are more efficiently treated on a coarse grid than on a fine grid. However, a coarse grid may only be used in conjunction with a finer one, requiring proper data transfer between successive grids. Multigrid methods rely on two basic principles: smoothing and coarse grid correction (CGC). First, standard iterative methods (*e.g.*, Gauss-Seidel) with good smoothing properties (that is, elimination of high spatial frequency modes) are used to treat non-smooth errors in the solution. Pre-smoothing is required because only a smooth error is well represented both on fine and coarse grids, while non-smooth errors exhibit aliasing on coarse grids, significantly reducing the efficiency of CGC. After a smooth error is obtained on the finest grid where a solution is sought, relaxation continues on coarser grids which are achieved by eliminating every other grid line in each coordinate direction. A coarse grid relaxation is substantially cheaper than its fine-grid counterpart, and is also more efficient in eliminating errors that are relatively smooth on a finer grid. Thus, efficiency can be increased by transferring (restricting) some of the fine-grid iterations that are required for convergence, to a coarser grid, and interpolating (prolongating) the results, that is, applying a coarse grid correction to advance the solution on the finest grid.

4.9.1 Full Approximation Storage (FAS) Multigrid

For nonlinear equations such as the RANS equation set, the Full Approximation Storage (FAS) multigrid algorithm [49] is mostly used. The integral form of the RANS equations (including the turbulence model equations) may be rewritten in operator notation for a steady-state flow as:

$$\underbrace{\left[\int_{\partial\Gamma} \left(\boldsymbol{\mathcal{H}}_c - \boldsymbol{\mathcal{H}}_d\right)\, dl - \int_{C_a} \boldsymbol{\mathcal{S}}\, dA\right]}_{\boldsymbol{\mathcal{N}}\left(\boldsymbol{\mathcal{Q}}\right)} = 0 \tag{4.145}$$

where $\boldsymbol{\mathcal{N}}\left(\boldsymbol{\mathcal{Q}}\right)$ is a nonlinear operator representing the RANS and model equations, operating on the dependent-variable vector $\mathcal{Q}$. In geometric multigrid, a hierarchy of spatial discretization (*i.e.,* grids) is constructed based on successive coarsening (*e.g.*, elimination of every other grid line, in each direction) of a given fine grid of mesh size h. The resulting grids are of typical mesh sizes $h, 2h, 4h, 8h, \ldots, 2^{(L+1)}h$, where L denotes the number of grid levels in the hierarchy. Typically, $L \leq 4$ levels are employed, and 4 fine-grid cells compose a single underlying coarse-grid cell (full-coarsening). The numerical solution is sought on the finest grid in the hierarchy

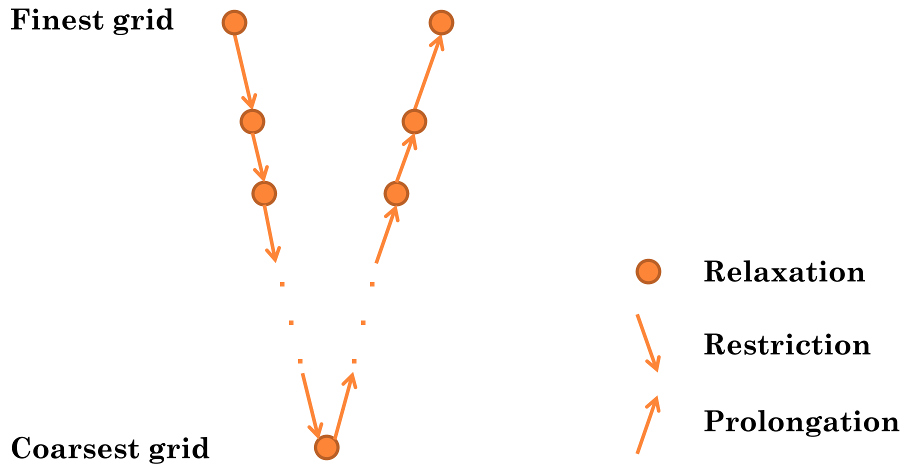


Figure 4.2: A schematic description of the V-Cycle for a four-level hierarchy.

$(l = 1)$, while the underlying coarse meshes $(l \geq 2)$ are only used to accelerate the rate at which a converged solution on the finest grid is obtained. Consequently, the residual is measured on the finest grid in the hierarchy.

The equivalent discrete operator, sampled on the l^{th}-level grid, is denoted by $\boldsymbol{\mathcal{N}_l}\left(\boldsymbol{\mathcal{Q}_l}\right)$. The basic multigrid algorithm employed in the EZAir suite of codes is based on the *V-Cycle*. The multigrid *V-Cycle* is composed of pre-relaxation, restriction, coarse-grid correction, and post-relaxation. Figure 4.2 shows a schematic description of the *V-Cycle*, the execution proceeds from left to right and top (finest grid) to bottom (coarsest grid) and back up again. Relaxation of non-smooth (high-frequency) error components on all grid levels is conducted via Gauss-Seidel iterations. On each grid but the coarsest, pre-relaxations are conducted prior to transferring (restricting) to the next-coarser grid and post-relaxations are conducted after interpolating (prolongating) and adding a coarse-grid correction. Each multigrid cycle is actually based on recursively invoking two-grid sub-cycles to solve the equations on each level of the hierarchy.

4.9.1.1 Fine-Grid Pre-Relaxation

Pre-relaxation sweeps are conducted on the fine-grid to obtain an approximate solution $\bar{\boldsymbol{\mathcal{Q}}}_l$ and a corresponding smooth numerical residual $\boldsymbol{\mathcal{R}}_l$ that satisfies $\boldsymbol{\mathcal{N}}_l\left(\bar{\boldsymbol{\mathcal{Q}}}_l\right) = -\boldsymbol{\mathcal{R}}_l$. One or two pre-relaxation sweeps are necessary to ensure that transferred fine-grid defects (residuals) are correctly sampled on coarser grids in the hierarchy, with relatively small aliasing.

4.9.1.2 Restriction to a Coarser Grid

An area-weighted restriction operator $\boldsymbol{\mathcal{I}}_l^{l+1}$ is employed to transfer the fine-grid approximate solution $\bar{\boldsymbol{\mathcal{Q}}}_l$ to a coarser grid. The transferred fine grid residual, $\widetilde{\mathcal{I}}_l^{l+1}\left(\mathcal{R}_l\right)$, is calculated by summing four equivalent fine-grid residuals (full-weighting). This type of restriction is required to retain the conservation properties of the numerical scheme on coarse grid levels. The coarse-grid operator, $\mathcal{N}_{l+1}$, is constructed in a similar manner to $\mathcal{N}_l$, but for a coarser mesh $(l+1)$. Appropriate coarse-grid equations of the following form are constructed with the aim to reduce low-frequency fine-grid errors:

$$\boldsymbol{\mathcal{N}}_{l+1}\left(\boldsymbol{\mathcal{I}}_l^{l+1}\boldsymbol{\mathcal{Q}}_l\right) - \boldsymbol{\mathcal{N}}_{l+1}\left(\boldsymbol{\mathcal{I}}_l^{l+1}\bar{\boldsymbol{\mathcal{Q}}}_l\right) = \widetilde{\boldsymbol{\mathcal{I}}}_l^{l+1}\left(\boldsymbol{\mathcal{R}}_l\right) \tag{4.146}$$

4.9.1.3 Coarse-Grid Relaxations

On intermediate levels, the coarse-grid equations are solved by recursively invoking another multigrid cycle. Starting from the initial solution, $\bar{\boldsymbol{\mathcal{Q}}}_{l+1}^{\mathbf{0}} = \boldsymbol{\mathcal{I}}_l^{l+1}\bar{\boldsymbol{\mathcal{Q}}}_l$, on the coarsest grid level of the hierarchy, the equations are solved via local iterations (*e.g.*, Gauss-Seidel). An approximate solution, $\bar{\boldsymbol{\mathcal{Q}}}_{l+1}$, and a corresponding coarse-grid residual, $\boldsymbol{\mathcal{R}}_{l+1}$ are formed, satisfying:

$$\boldsymbol{\mathcal{N}}_{l+1}\left(\bar{\boldsymbol{\mathcal{Q}}}_{l+1}\right) = \underbrace{\boldsymbol{\mathcal{N}}_{l+1}\left(\boldsymbol{\mathcal{I}}_l^{l+1}\bar{\boldsymbol{\mathcal{Q}}}_l\right) + \widetilde{\boldsymbol{\mathcal{I}}}_l^{l+1}\left(\boldsymbol{\mathcal{R}}_l\right)}_{\boldsymbol{Fine-grid\ Constants}} - \boldsymbol{\mathcal{R}}_{l+1} \tag{4.147}$$

Note that the first iteration on the coarse-grid is driven only by $\widetilde{\boldsymbol{\mathcal{I}}}_l^{l+1}\left(\boldsymbol{\mathcal{R}}_l\right)$, since $\boldsymbol{\mathcal{N}}_{l+1}\left(\bar{\boldsymbol{\mathcal{Q}}}_{l+1}^{\mathbf{0}}\right) = \boldsymbol{\mathcal{N}}_{l+1}\left(\widetilde{\boldsymbol{\mathcal{I}}}_l^{l+1}\bar{\boldsymbol{\mathcal{Q}}}_l\right)$. Also note that the first two expressions on the right-hand-side of Equation (4.147), which originates from the fine-grid, remain constant throughout coarse-grid relaxations.

4.9.1.4 Coarse-Grid Correction

The difference in the coarse-grid solution $\bar{\mathcal{Q}}_{l+1}$ is interpolated back to the fine-grid (*prolongated*), where it is used to correct the fine-grid solution, $\bar{\mathcal{Q}}_l$, via

$$\Delta\bar{\mathcal{Q}}_l = \mathcal{I}_{l+1}^{l}\left(\bar{\mathcal{Q}}_{l+1} - \mathcal{I}_l^{l+1}\bar{\mathcal{Q}}_l\right) \tag{4.148}$$

The prolongation operator is based on a bi-linear interpolation.

4.9.1.5 Fine-Grid Post-Relaxation

Aside from valuable corrections, application of a coarse-grid correction also introduces new errors to the fine-grid solution, arising from interpolation inaccuracies. Several post-relaxation passes (Gauss-Seidel iterations) are conducted to reduce these new errors. Generally, least four post-relaxation passes on each level are required to efficiently reduce the new error components introduced by interpolating and applying a coarse-grid correction.

4.9.2 Full Multi Grid (FMG) Initialization

To improve the convergence characteristics of non-linear multigrid it is important to begin the solution with a good initial approximation. This may be achieved by a different multigrid initialization procedure, termed *full multigrid*. Let Figure 4.2 describe the classical multigrid 4 level V-cycle. A full multigrid initialization is described in Figure 4.3. Note that the solution starts from the coarsest grid and through the initialization stage (that may span over a few hundred computational steps) the procedure is different than the classical multigrid procedure.

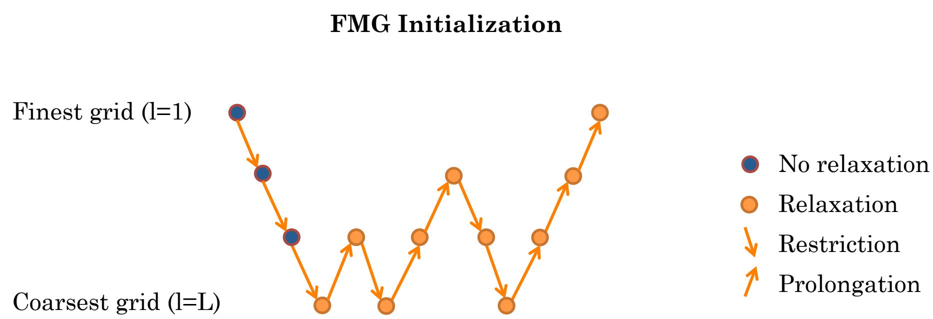


Figure 4.3: Schematic description of Full-Multigrid (FMG) initialization for a four-level hierarchy.

Chapter 5

Boundary Conditions

5.1 Introduction

The EZAir suite of codes contains a wide variety of boundary conditions. Certain boundary conditions are automatically set by the code, based on the identification of the topology. The rest of them are set by the user. Being a finite volume code, the notion of a ghost cell is utilized throughout. In what follows, the subscript "g" signifies a ghost cell, the subscript "r" signifies a real cell where the flow is solved (a "real" cell), and the subscript "f" signifies the face (prescribed) value. The user is referred to Section 10.8 for usage.

5.2 Wall Boundary Conditions

5.2.1 Impermeable Wall Conditions

Let $\hat{n}$ be a unit vector normal to the face of a boundary cell, whose components are (n_x, n_y, n_z), and let t^1 and t^2 be the two unit vectors tangent to the face of a boundary cell, the velocity components in the ghost cell are calculated by solving the system:

$$\begin{bmatrix} n_x & n_y & n_z \\ t_x^1 & t_y^1 & t_z^1 \\ t_x^2 & t_y^2 & t_z^2 \end{bmatrix} \begin{pmatrix} u_g \\ v_g \\ w_g \end{pmatrix} = \begin{pmatrix} 2\vec{V}_f \cdot \hat{n} - \vec{V}_r \cdot \hat{n} \\ \vec{V}_r \cdot \hat{t}^1 \\ \vec{V}_r \cdot \hat{t}^2 \end{pmatrix} \tag{5.1}$$

The prescribed velocity vector $\vec{V}_f$ includes any motion of the boundary surface.

5.2.2 No-Slip Condition

The no-slip condition is much easier to implement:

$$\vec{V}_g = 2\vec{V}_f - \vec{V}_r \tag{5.2}$$

5.2.3 Adiabatic Wall

The temperature is set using:

$$T_g = T_r \tag{5.3}$$

while the pressure is set using

$$p_g = p_r \tag{5.4}$$

The density is evaluated using the equation of state.

5.2.4 Isothermal Wall

The temperature is set using:

$$T_g = 2T_f - T_r \tag{5.5}$$

while the pressure is set using

$$p_g = p_r \tag{5.6}$$

The density is evaluated using the equation of state.

5.2.5 Prescribed Heat Flux

The temperature is set using:

$$T_g = T_r + l\frac{q}{\kappa} \tag{5.7}$$

where q is the prescribed heat flux and l is the distance between the ghost cell and the real cell, while the pressure is set using

$$p_g = p_r \tag{5.8}$$

The density is evaluated using the equation of state.

5.2.6 Radiative Equilibrium

The temperature at the wall is calculated from the surface energy balance equation:

$$\kappa\frac{\partial T_f}{\partial \hat{n}} = \epsilon\sigma T_f^4 \tag{5.9}$$

Where ϵ is the wall emissivity, and σ is the Stefan-Boltzmann constant. The equation is solved numerically, then temperature is set using:

$$T_g = 2T_f - T_r \tag{5.10}$$

5.3 Far Field Conditions

5.3.1 Turkel Type Conditions

The characteristic relations that are due-to Turkel are utilized. For supersonic inflow, the flow quantities at the inflow boundary are set based on the current values of the corresponding boundary:

5.3.1.1 Turkel Inlet

$$\rho_g = \rho_\infty \tag{5.11}$$
$$\vec{V}_g = \vec{V}_\infty \tag{5.12}$$
$$p_g = \frac{\rho_\infty}{\rho_r} p_r \tag{5.13}$$

5.3.1.2 Turkel Outlet

$$\rho_g = \rho_r + \frac{p_\infty - p}{a_\infty^2} \tag{5.14}$$
$$\vec{V}_g = \vec{V}_r \tag{5.15}$$
$$p_g = p_\infty \tag{5.16}$$

5.3.2 Riemann Type Conditions

Let q be the normal to the boundary face velocity. The Riemann invariants are calculated based on the following:

$$\begin{aligned} R^+ &= q_r - \frac{2}{\gamma - 1} a_\infty \\ R^- &= q_\infty - \frac{2}{\gamma - 1} a_r \end{aligned} \tag{5.17}$$

The ghost Riemann invariants are obtained using:

$$R_g = \frac{1}{2}\left(R^+ + R^-\right) \tag{5.18}$$

5.3.2.1 Riemann Inlet

$$\begin{aligned} \rho_g &= \frac{\rho_\infty}{a_\infty^{\frac{1}{\gamma-1}}} \left\{ \underbrace{\left[\frac{\gamma - 1}{4}\left(R^- - R^+\right)\right]^{\frac{1}{\gamma-1}}}_{a_g} \right\}^2 \\ p_g &= \sqrt{a_g} \rho_g \gamma \end{aligned} \tag{5.19}$$

5.3.2.2 Riemann Outlet

Let s be the entropy, the relations for ρ_g and p_g are given by:

$$\begin{aligned} s &= \frac{\rho^\gamma}{\gamma p} \\ \rho_g &= \left(a_g^2 s\right)^{\frac{1}{\gamma-1}} \\ p_g &= \sqrt{a_g}\rho_g\gamma \end{aligned} \tag{5.20}$$

5.3.3 Fixed (Supersonic Inlet)

$$\begin{aligned} \vec{V}_g &= \vec{V}_\infty \\ p_g &= p_\infty \\ T_g &= T_\infty \end{aligned} \tag{5.21}$$

5.3.4 Extrapolation (Supersonic Outlet)

$$\begin{aligned} \vec{V}_g &= \vec{V}_r \\ p_g &= p_r \\ T_g &= T_r \end{aligned} \tag{5.22}$$

5.3.5 Inout

These boundary conditions are specific to low subsonic flows. With the exception of the pressure, these boundary conditions set "Fixed" conditions for inlet and "Extrapolation" conditions for outlet. Namely,

- **Inlet :**

$$\begin{aligned} (\rho, u, v, w)_g &= (\rho, u, v, w)_\infty \\ p_g &= p_r \end{aligned} \tag{5.23}$$

- **Outlet :**

$$
\begin{aligned}
(\rho, u, v, w)_g &= (\rho, u, v, w)_r \\
p_g &= p_\infty
\end{aligned}
\tag{5.24}
$$

5.3.6 Stagnation Inlet

Relevant only for a subsonic inlet, the total pressure and temperature are specified, and the inlet velocity, static pressure, and static temperature are computed from the stagnation conditions.

5.4 Axis Boundary Conditions

The axis boundary condition is used for evaluating velocity gradients in high order spatial discretization for diffusive fluxes (for cells adjacent to the axis). Note that there are no fluxes through the axis.

5.5 Symmetry Boundary Conditions

Symmetric boundary conditions are treated as follows:

$$
\begin{aligned}
\vec{V}_g &= \vec{V}_r - 2\left(\vec{V}_r \cdot \hat{n}\right)\hat{n} \\
p_g &= p_r \\
T_g &= T_r
\end{aligned}
\tag{5.25}
$$

5.6 Intake Conditions

$$\begin{aligned} \vec{V}_g &= \vec{V}_r \\ p_g &= p_\infty \\ T_g &= T_r \end{aligned} \tag{5.26}$$

5.7 E-Mesh Boundary Conditions

Boundary conditions are automatically implemented based on the mesh topology. No action is required by the user. Please refer to Section 6.1.

5.8 Periodic Boundary Conditions

Boundary conditions are automatically implemented based on the mesh topology. No action is required by the user. Please refer to Section 6.1.

5.9 Actuator Disk Model

The EZAir Suite of Codes provides the means to simulate a rotor or a propeller using an actuator disk model. The model that is implemented is the one described in the work by Verweij [50]. For complete details of the model, the reader is referred to the original paper [50].

Chapter 6

Computational Mesh

The grid generation process in itself presents one of the main obstacles in performing numerical simulations. Using a structured mesh, as implemented in the EZAir suite of codes, there are two different approaches for mesh generation. The first is the patched grid approach and the second is the over-set grid approach, known as Chimera [1]. Each of the blocks may have an arbitrary topology as long as it does not contain singularities or negative cell volumes. The EZAir suite of codes has the capability to automatically identify certain topologies.

6.1 Single Mesh Topology

Figure 6.1 shows a computational mesh that contains a geometrically periodic mesh about a body of revolution. This mesh also contains a singular axis. Such a singularity, at the edge of the computational domain is permitted by EZAir. Figure 6.2 shows a computational mesh that is termed in EZAir as an E type mesh. Any mesh that includes a collapsed edge or a cut within the mesh is considered an E type mesh. For example, a C Type or an H type mesh are considered and “e-mesh” in EZAir.

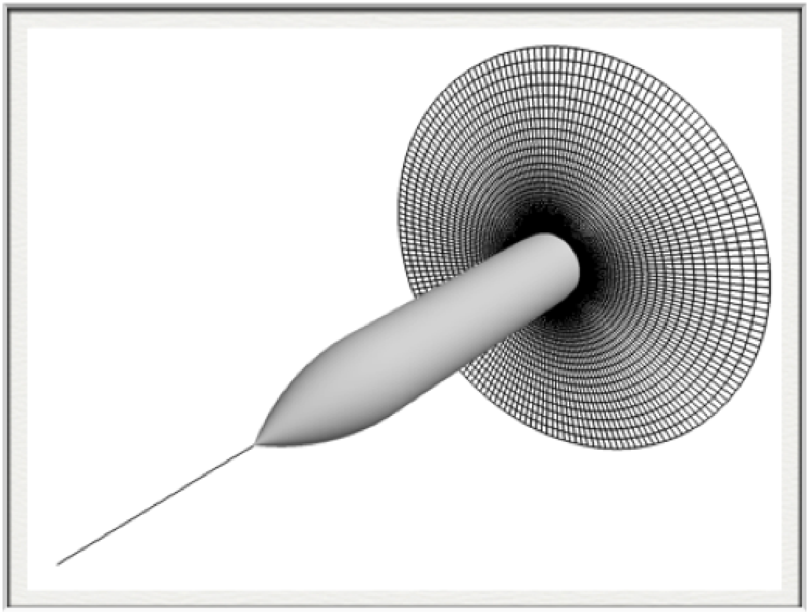


Figure 6.1: O type Mesh with an axis

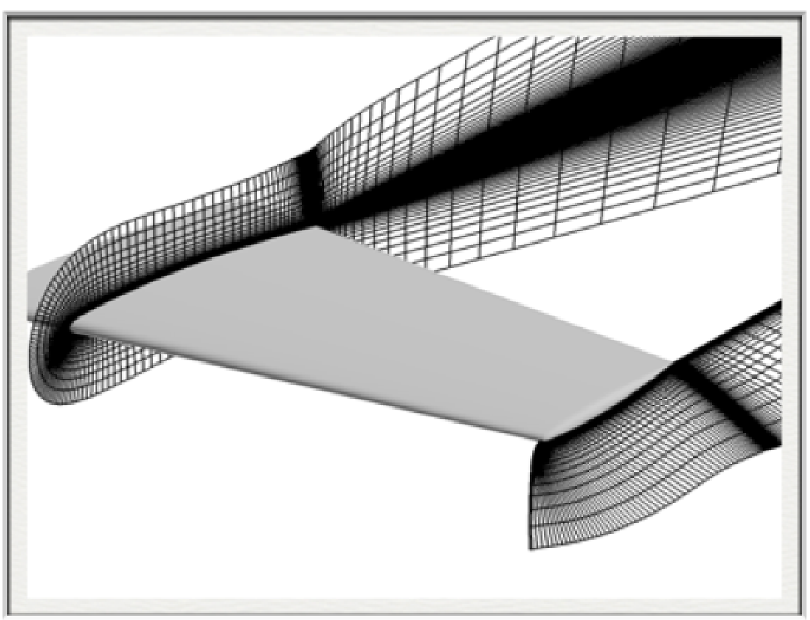


Figure 6.2: E type Mesh

6.2 Definition of a Body

A body in EZAir terminology is composed of a set of zones that define an enclosure or a semi-enclosure (*e.g.,* an open wing that is attached to a fuselage). Body is the quintessential object that cuts holes in zones. Every zone must be contained in one body (even floating grids that do not contain any walls). The zones that comprise a body are not cut by this body. Ultimately, the "body" is comprised of all the walls (impermeable or no-slip boundary conditions) that are defined in all the zones that belong to this body. A body can be a mixture of multi-block grids, overset grids, or even a single grid. Specific walls can be set as not.entity so that they will not be a part of the enclosure.

6.3 Chimera Module

The overset Chimera grid approach is utilized by the EZAir suite of codes to facilitate moving body flow simulations, involving large geometry changes. The Chimera grid approach is also a robust and efficient tool for complex geometry grid generation. In the Chimera grid approach, a separate computational mesh is generated for each component, such as the fuselage or the airfoil of a vehicle, separately. An outer mesh is generated so that it fully includes the meshes of all the components. The domain decomposition in this approach is simplified and the number of zones that are required is relatively small. The generation of grids surrounding each component is performed independently, and therefore points of a certain mesh may be located within the solid boundaries another. Regions located within solid boundaries are called "holes." This is treated by excluding points that are in the holes from the solution process and by using interpolations to update the edges of the holes. The suite of routines to determine the holes and hole boundaries has been added to the EZAir suite of codes so that moving bodies and the holes they created can be tracked (see Sections 10.4.2 and 10.5.4 for usage).

The Chimera module that is implemented in the EZAir suite of codes provides the choice between manual or automatic hole cutting. The module has a few important features that enhance its performance and flexibility. To name a couple, the module allows to define geometrical virtual bodies (called "Phantom") for hole cutting. Another is the ability to cut optimal holes that provide sufficient overlap for interpolation on one hand while minimizing the overlap on the other hand (see Section 6.3.1). The hole minimization process is a part of automatic hole cutting. For a detailed description of the automatic Chimera procedure and the means to control it, the reader is referred to Section 10.4.1.

6.3.1 Automatic Overlap Minimization

This procedure minimizes the overlap between neighboring grids (for usage see Table 10.22). When using the minimization procedure it is recommended to refrain from using manual hole cutting directives as much as possible. It is assumed that the physical holes must be accurate and therefore the bodies and the "surface.cut" must be used properly (see Section 6.3.2 and Figures 6.3-6.7).

The process selects the "best live cell" in every physical space using one of the previously mentioned algorithms and continues to add layers of cells, until there is sufficient overlap for interpolation (or it is not possible to add more cells).

The algorithm may result in isolated grid regions that do not contain boundary conditions of any "real" kind. These regions are termed "enclaved" enclosures and can be eliminated using the options described in Tables 10.28 and 10.117. The "best live cell" is identified using a "Donor Suitability Function" that is a linear combination of the options listed in Tables 10.24 and 10.116.

The automatic minimization procedure allows to cut the necessary holes to the grid without the use of any of the manual means as described above (*e.g*, **envelope**). The minimization process can result in an enclosure that contains "live" grid points. For that matter an enclaved grid part is an isolated grid part (holes all around it), that does not contain any "real" boundary condition. These enclosures can be eliminated using one of two options that are detailed below (see Table 10.28 and 10.117).

6.3.1.1 Recommended Practice

The automatic Chimera module should be used with great care. Experience shows that the automatic minimization may result in jagged Chimera fronts. Jagged edges, in turn, often result in a reduced accuracy of the overall force calculations, especially for low Mach number flows (note that jagged edges may be the result of manual hole cutting as well). Therefore, if minimization and force overlap are both conducted automatically, it is strongly recommended to use a reference pressure (that should be the free stream pressure or close to it) for the purpose of force calculations. It is also strongly recommended to closely examine the "NF" function (see Appendix C). Note that the accuracy level of force summation (regardless of reference values) may be monitored using the log options that start with "CHK" (*e.g.*, "CHK(L)"; for complete details see Appendix B).

6.3.2 The Chimera Procedure

When using manual hole cutting, the Chimera implementation is composed of two main stages: Removing nodes and thereby cells that reside in bodies or "phantoms," and seeking interpolation points for the pseudo boundary cells that comprise the "Chimera front." The first stage of nodes removal takes into account bodies, phantoms, whitelist, blacklist of included, and excluded flags for each body, zone, or phantom. The parallelism mechanism is explained in Chapter 9. In this report only the actual removal algorithm is discussed.

Initially, a body can be declared "external" if every vertex outside its entities (wall boundary conditions) bounding box can be considered in the flow region relative to the body. A body can also be declared "internal" if the flow region must reside within the cells of these body grids. By default, each body enclosure is considered for hole cutting. Next, the user can use envelopes (defined ahead) and "phantoms" for the sake of both improved performance and for the elimination of unwanted grid areas. Every node, of every zone is then checked, considering of course the user's white and black lists, against bodies and virtual bodies ("phantoms"). In addition, a surface cut algorithm has been added to the procedure. The "surface.cut" algorithm is an additional and complementary algorithm to the existing hole cutting one which kills nodes that reside in a body enclosure. The "surface.cut" algorithm kills cells that intersect body surfaces (see Figures 6.3, 6.4, 6.5, 6.6, and 6.7 for an illustrative description of how the surface.cut directive works). Together with the original hole cutting algorithm, they eliminate any cell that resides in the physical space of a different body. The actual hole cutting procedure is composed of the following steps:

1. Remove nodes that reside within the bounds of phantoms (virtual bodies).
2. Remove nodes that reside inside bodies.
3. Remove nodes that reside in envelopes.
4. Remove cell nodes that are cut by other body quads using the surface.cut option (on by default, see Figures 6.3, 6.4, 6.5, 6.6, and 6.7, and Tables 10.38, 10.109, and 10.110).
5. Explicit node removal.
6. Revive nodes explicitly (with -chimera revive) or with phantoms (-revive).

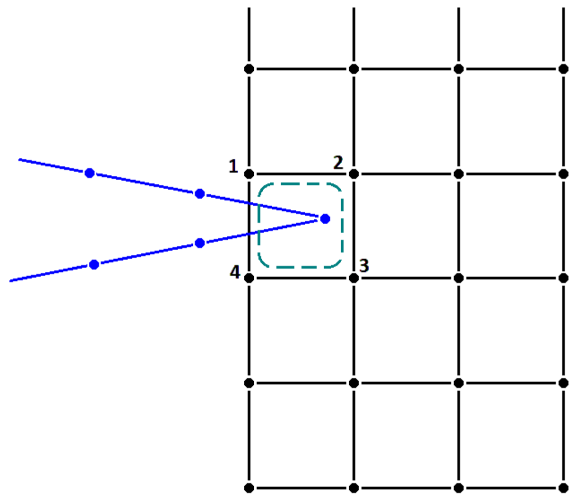


Figure 6.3: The surface.cut directive controls the hole cutting of the cell that is marked by a dashed green curve

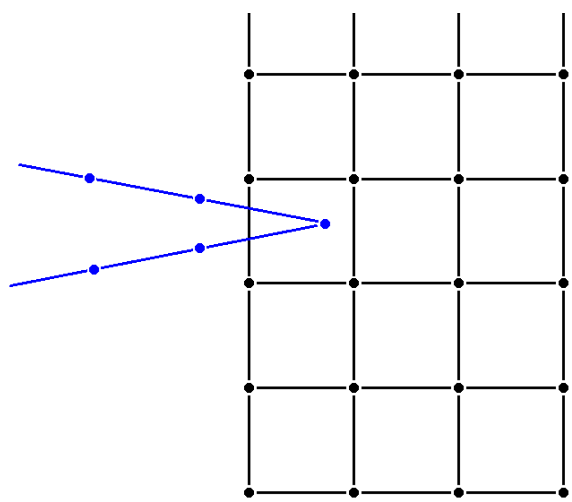


Figure 6.4: Outcome of surface.cut no

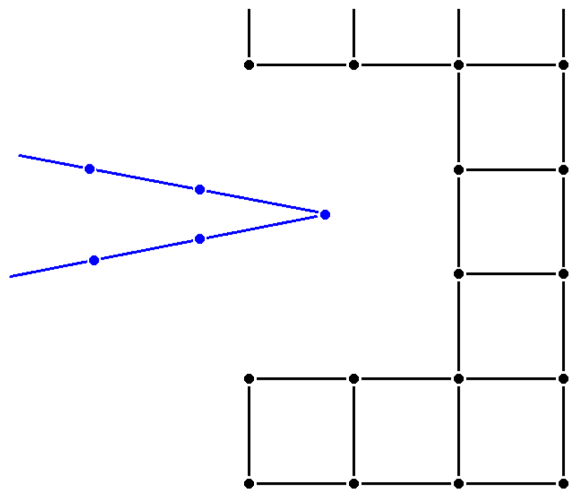


Figure 6.5: Outcome of surface.cut far

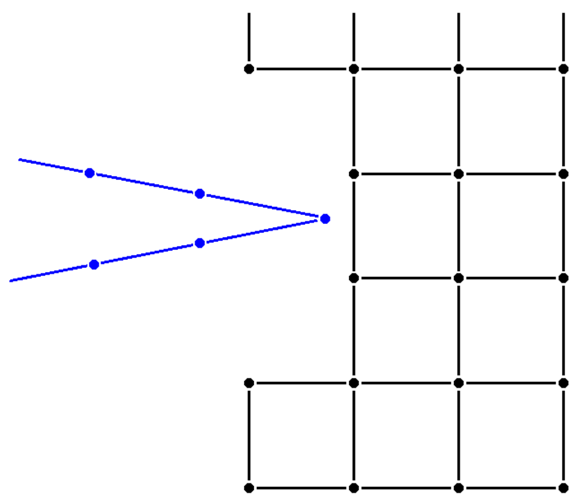


Figure 6.6: Outcome of surface.cut near

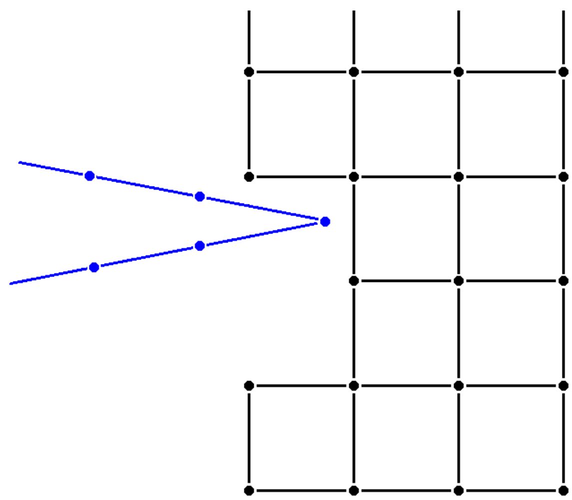


Figure 6.7: Outcome of surface.cut tight

Once all nodes are classified (whether they are a part of the flow field discretization or not), the cells are classified whether they are a part of the computational flow field yes or no, and if they are a part of the “Chimera front.” The default front is comprised of two layers of cells (can be changed to a single layer by the user), in order to be able to simulate first and second order in space transparently. Each cell’s nodes in the Chimera front is then searched in other grids’ live cells. This search forms the interpolation list from which the chimera fronts cell evaluates its properties in every iteration.

6.4 Envelopes

An envelope is a method for quickly defining a region from existing grids. This region is then used to cut holes in selectable zones. Note that the number of envelopes is unlimited, in particular the number of envelopes from a single grid that cuts holes in another specific zone is unlimited. Figure 6.8 provides an example for the use of envelopes to cut a hole in a wing mesh. The figure shows three different envelopes that are defined based on the wing mesh and are used to cut a hole in the fuselage hole (red mesh).

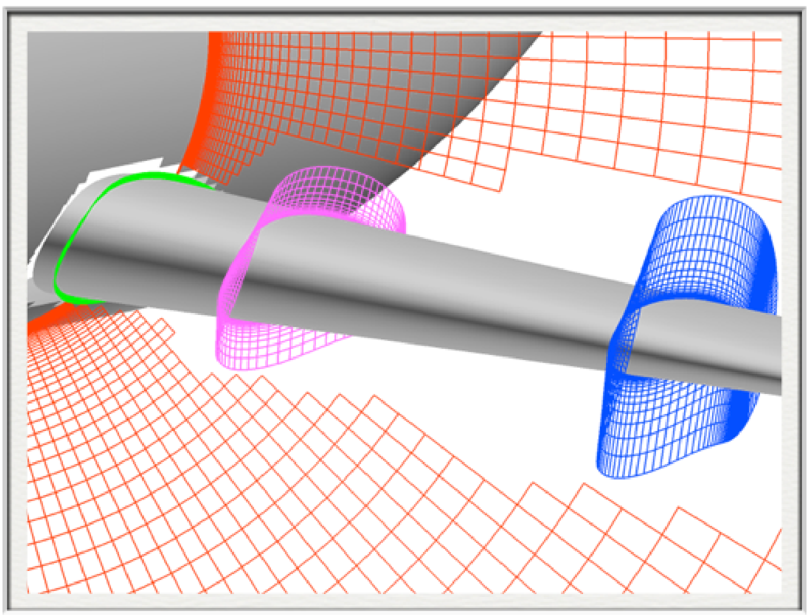


Figure 6.8: Holes cut by envelopes

6.5 Phantoms

This section contains a few examples of "phantoms." Figure 6.9 shows a "region," Figure 6.10 shows an "ellipse," Figure 6.11 shows an "ellipsoid," Figure 6.12 shows a "cylinder," and Figure 6.13 shows a "fan." See Section 10.5.4.3 for usage.

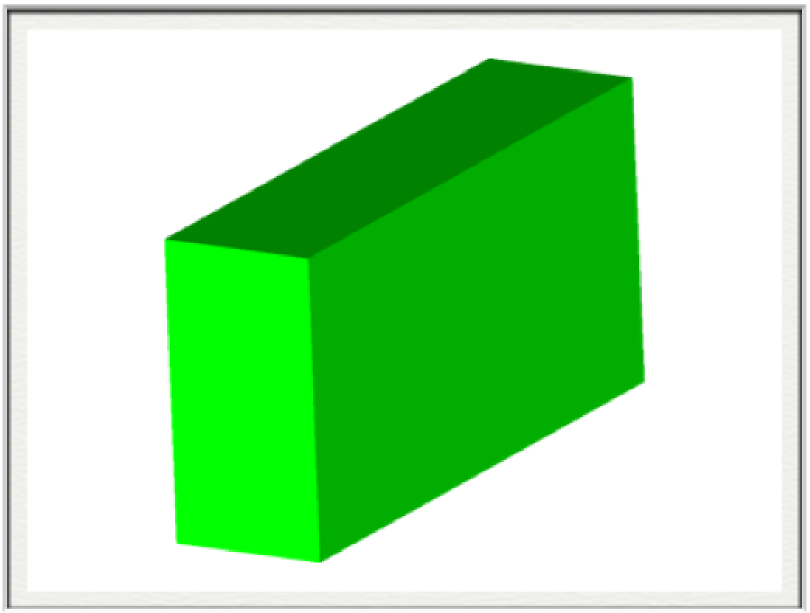


Figure 6.9: Phantom; region

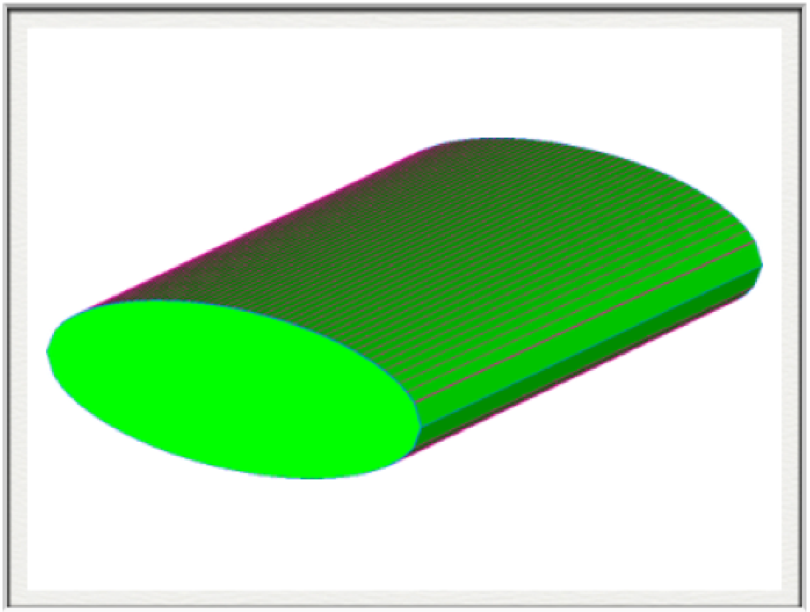


Figure 6.10: Phantom; ellipse

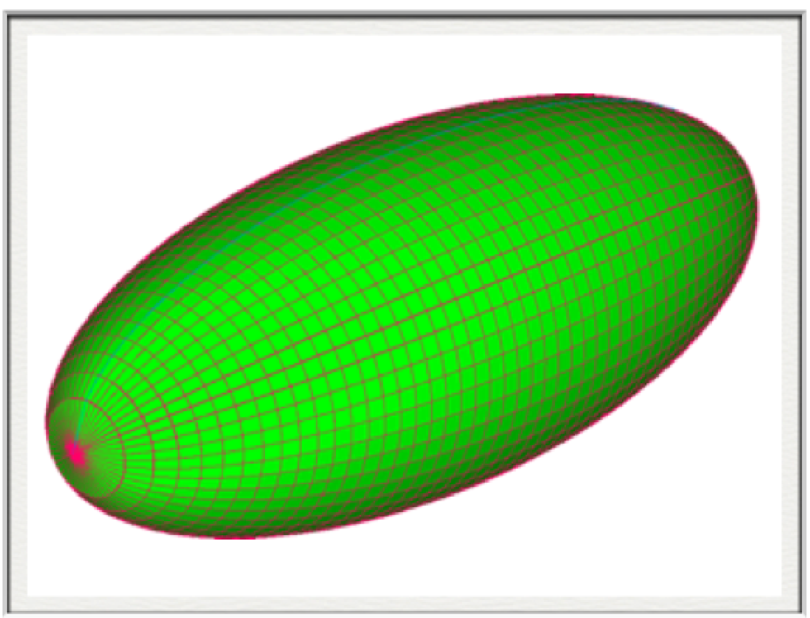


Figure 6.11: Phantom; ellipsoid

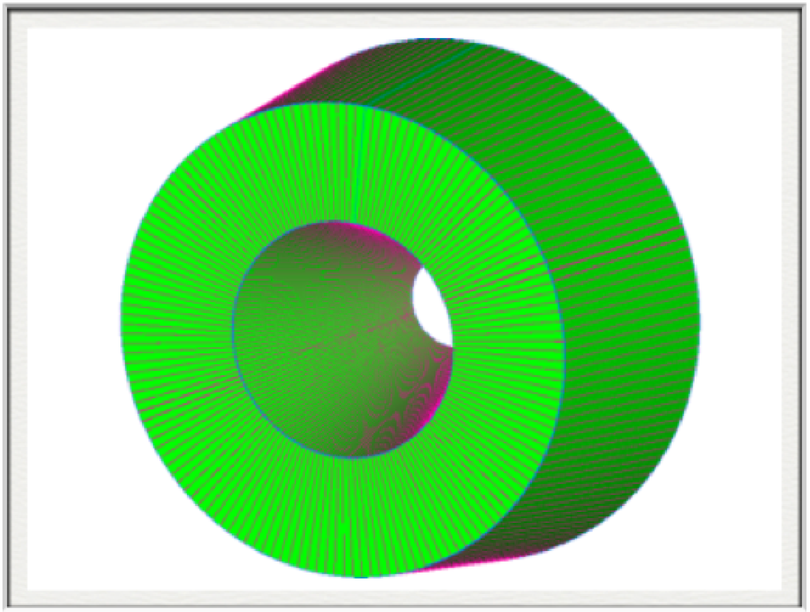


Figure 6.12: Phantom; cylinder

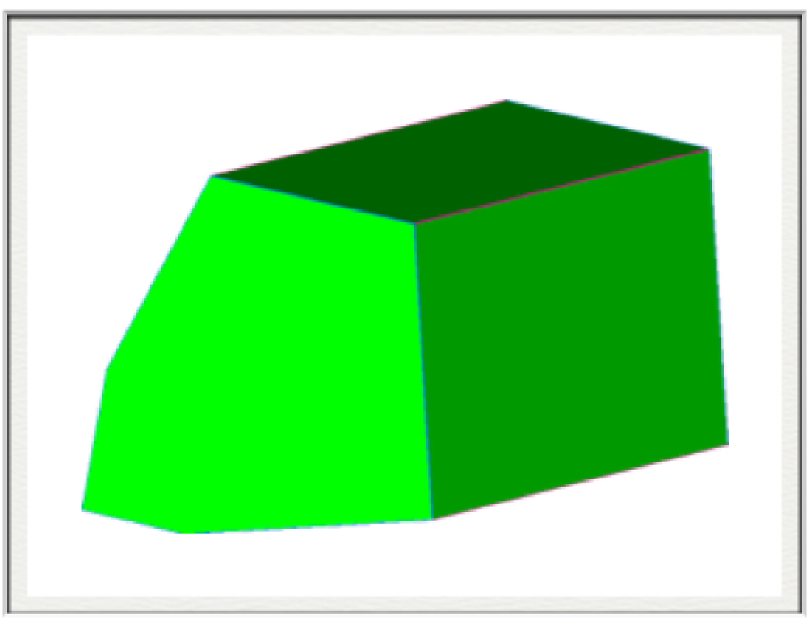


Figure 6.13: Phantom; fan

Chapter 7

Six Degrees of Freedom Simulation

The flow equations solved by the EZAir suite of codes are written in an inertial coordinate system (the computational meshes may move but the observer is set). Therefore, the forces and moments that are calculated about bodies are forces and moments in a Cartesian coordinate system, with its origin set in a predefined static position in space. In the beginning of the flow simulations, it is usually attached to the aircraft tip. For further simplification of the motion calculation, the aerodynamic forces and moments are calculated about a reference point that is the center of gravity of a certain moving entity. Since the aerodynamic coefficients are given in a Cartesian coordinate system, the translational equation of motion is formulated and solved in the Cartesian coordinate system.

Solving the rotational equation of motion in such coordinates is far more complicated due to the need to recalculate the moment of inertia matrix with each rotation of the body. It is then opted to solve the rotational equation of motion in body coordinates. In what follows, the numerical solution procedure of the equation of linear motion and of the rotational equation of motion is described.

See Section 10.9 for usage.

7.1 Translational Equation of Motion

The translational equation of motion is obtained by applying Newton's law of motion. Since it is assumed that the mass of an aircraft or any moving entities remain constant, the law takes the form:

$$\sum \overline{F}^n = m^n \overline{a}^{n+1} \tag{7.1}$$

where $\sum \overline{F}$ is the sum of forces (aerodynamic and other) acting on the body, m is the mass of the entity in question, and $\overline{a}$ is the acceleration vector. The translational velocity of a certain entity at a time step $n+1$ is obtain through integration of the equation:

$$\Delta \overline{V}^{n+1} = \int_{t}^{t+\Delta t} \overline{a}^n dt \tag{7.2}$$

The translation vector of the a certain entity is obtain through integration of the equation:

$$\Delta \overline{X}^{n+1} = \int_{t}^{t+\Delta t} \overline{V}^n dt \tag{7.3}$$

7.2 Rotational Equation of Motion

The rotational equation of motion is obtained by applying Newton's second law for angular motion. Once again, it is assumed that the geometry of a parent aircraft and any of its stores remain constant and therefore, one may assume that the inertia matrix remains unchanged as well. The rotational equation of motion in body coordinates takes the form:

$$\sum \overline{M}^n = \mathbf{I}^n \frac{d\overline{\Omega}^{n+1}}{dt} + \overline{\Omega}^n \times \overline{H}^n \tag{7.4}$$

where $\sum \overline{M}$ is the sum of moments, $\mathbf{I}$ is the moment of inertia matrix, $\overline{\Omega}$ is the angular velocity vector, and $\overline{H} = \mathbf{I}\overline{\Omega}$ is the angular momentum vector. The aerodynamic moments, evaluated in Cartesian coordinates, are transformed into body coordinates using quaternions (see next sub-section). The angular acceleration, $\overline{\alpha} = \frac{d\overline{\Omega}}{dt}$, is calculated by solving Equation (7.4) and the angular velocity is obtained by integrating the equation:

$$\Delta\overline{\Omega} = \int_{t}^{t+\Delta t} \overline{\alpha} dt \tag{7.5}$$

7.2.1 Quaternions

A quaternion consists of four parameters: a scalar (q_0), and a vector (q_1,q_2,q_3). An extra constraint is required, $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. This constraint can be easily enforced through a normalization:

$$\overline{q} = \frac{\overline{q}}{|\overline{q}|} \tag{7.6}$$

Time derivative of the quaternion is obtained through the relation:

$$\dot{\overline{q}} = \frac{1}{2}\left[\overset{\circ}{\overline{\Omega}}\right]\overline{q} \tag{7.7}$$

where the matrix $\overset{\circ}{\overline{\Omega}}$ is composed of the angular velocity components as follows:

$$\overset{\circ}{\overline{\Omega}} = \begin{bmatrix} 0 & \Omega_1 & \Omega_2 & \Omega_3 \\ -\Omega_1 & 0 & -\Omega_3 & \Omega_2 \\ -\Omega_2 & \Omega_3 & 0 & -\Omega_1 \\ -\Omega_3 & -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \tag{7.8}$$

The updated quaternion is obtained by integrating the equation:

$$\Delta\overline{q} = \int_{t}^{t+\Delta t} \dot{\overline{q}} dt \tag{7.9}$$

7.3 Integration

Numerical integration of all the above mentioned integrals may be conducted using the Adams-Bashforth or Adams-Moulton methods of 1^{st} through 5^{th} order. The integration method is set by the user using the "-dof" directive (see Table 10.245 for usage of the directive) and the "DOF" options (See Appendix A for a complete list of the options, including the motion simulation integration method).

7.4 Motion with More than Six Degrees of Freedom

Certain cases require the simulation of a complex motion that involves many degrees of freedom. For example, the deployment of a store from a parent aircraft while its wing open and/or the store has moving canards or flaps etc. The EZAir suite of codes provides two options to conduct such simulations. The first, for an advanced user, through an external program that includes the system equations of motion as well as the integration means and the second through EZAir directives as a part of the input files. The first option is described in Chapter 11 while the second option is described in Section 10.9.1.

Chapter 8

Static and Dynamic Aeroelasticity

Solution of the discrete aeroelastic equation of motion, for a structural large model, can be computationally burdensome. As a result, there is a motivation to reduce the size of the problem without sacrificing significant accuracy. One approach to reduce the computational burden of FEM-based aeroelastic analysis is to represent the structural displacements as a linear combination of a set of its modes of vibration, as given by:

$$\{u\} = \begin{bmatrix} \Phi_R & \Phi_E \end{bmatrix} \begin{Bmatrix} \xi_R \\ \xi_E \end{Bmatrix} \tag{8.1}$$

In this case, $[\Phi_R]$ represents the rigid-body modes, and $[\Phi_E]$ represents a subset of the elastic modes of vibration. Both $[\Phi_R]$ and $[\Phi_E]$, along with their corresponding natural frequencies, ω_i, are solutions to the eigenvalue problem of the free vibration of the structure, which is given by:

$$\left([K] - \omega_i^2 [M]\right) \{\Phi_i\} = \{0\} \tag{8.2}$$

By utilizing such a decomposition, the size of the aeroelastic problem reduces to the number of modes picked up. Typically, a few tens of modes are sufficient to represent displacements of a full aircraft configuration, which, in the FEM, may involve hundred thousands degrees of freedom.

The EZAir suite of codes utilizes a modal structural model for interfacing the flow solver and structural models, for calculating elastic shape deformations, and applying them to the computational mesh. It utilizes two forms of aeroelastic displacements, a simple rigid displacement of the whole mesh and a deformation that is based on trans-finite interpolations. This chapter briefly describes the models for static and dynamic aeroelasticity and the TFI approach.

The reader is referred to Section 10.9.2 for usage instructions.

8.1 Static Aeroelasticity

The modal approach to static aeroelasticity assumes that the elastic deformations of the aircraft structure under external loads can be described as a linear combination of a set of low-frequency elastic mode shapes $[\phi_E]$, typically 10 to 30, namely

$$\{u_E\} = [\phi_E]\{\xi_E\} \tag{8.3}$$

where $\{\xi_E\}$ is the generalized elastic displacement vector. The resulting static equilibrium equation in generalized coordinates is:

$$[K_E]\{\xi_E\} = \{F_E\} \tag{8.4}$$

where $[K_E]$ is the generalized stiffness matrix associated with $[\phi_E]$, and $\{F_E\}$ is the associated generalized aerodynamic force vector. Orthogonality of the rigid-body and elastic modes with respect to the structural mass and stiffness matrices implies that $[K_E]$ is diagonal and that inertia relief effects in the right hand side of Equation (8.4) are taken care of automatically. The rigid-body counterpart of Equation (8.4) is used below in the maneuver trim process.

The generalized forces in Equation (8.4) are obtained by

$$\{F_E\} = [\phi_A]^T \left(\{F_A\} - \{F_C\}\right) \tag{8.5}$$

where $\{F_A\}$ is the aerodynamic force vector at the aerodynamic surface grid points, $\{F_C\}$ is the aerodynamic force vector associated with the reference aerodynamic shape, and $[\phi_A]$ is the elastic modes matrix, expressed at the aerodynamic interface grid points as described below. It is assumed that the reference loads vector $\{F_C\}$ has been defined in a previous aerodynamic analysis. This is the case when the reference aerodynamic shape is the "cruise shape", namely the shape designed for best aerodynamic performance at nominal cruise conditions. The reference aerodynamic shape can also be defined as the "jig shape", namely the unstressed shape, for which $\{F_C\} = 0$.

Using the modal approach, the modal stiffness matrix and the modes matrix are the only structural data required for the maneuver load analysis. These matrices are calculated by the finite element code, and are read in the CFD run as it starts, and after each structural optimization run. Since the number of structural modes used is typically small (in this work 15 structural modes were found to be conservatively sufficient) very little structural data is required to be transferred.

8.2 Equations of Motion for Uncoupled Aeroelastic Motion

The equations of motion corresponding to an uncoupled aeroelastic motion, given in generalized coordinates as:

$$[M_{ii}]\left\{\ddot{\xi}\right\} + \left[2\zeta M_{ii}\sqrt{\Omega_i}\right]\left\{\dot{\xi}\right\} + [M_{ii}\Omega_i]\{\xi\} = \{F(t)\} \tag{8.6}$$

where M is the diagonal, generalized-mass matrix, Ω_i are the eigenvalues, ζ is the structural damping coefficient, ξ is the vector of generalized displacements, and F are the aerodynamic generalized forces.

8.2.1 Numerical Scheme

Since the generalized mass matrix is diagonal, the modes are uncoupled and the time integration of the differential equation can be performed for each mode separately. Finite differences are used to evaluate the time derivatives of ξ. In addition, the equations are put in *delta form* so that only the increments are taken into account. This results in the following set of equations:

$$\Delta\ddot{\xi}_i + 2\zeta\sqrt{\Omega_i}\Delta\dot{\xi}_i + \Omega_i\Delta\xi_i = \frac{\Delta F_i}{M_{ii}} \tag{8.7}$$

one equation for each mode.

8.2.2 Solution Method

Prior to numerically solving the modal equations of motion, the system is rewritten in a state space formulation, as follows:

$$\begin{aligned} \Delta\dot{\xi}_i &= \Delta\eta_i \\ \Delta\dot{\eta}_i &= \frac{\Delta F_i}{M_{ii}} - 2\zeta\sqrt{\Omega_i}\Delta\eta_i - \Omega_i\Delta\xi_i \end{aligned} \tag{8.8}$$

The system can now be integrated using a standard Runge-Kutta method. The EZAir suite of codes utilizes a 4^{th} order Runge-Kutta scheme to advance the solution in time. This formulation has been found very robust.

8.3 Mesh Deformations

EZAir has a single method in which a grid is deformed based on the elastic deformations. Each grid point is deformed exactly as the computationally closest elastically deformed surface point. This method suits grids that have a single elastic surface. Collar grids that can typically have more than a single deformed surface can be reevaluated if they are initially created using the semi-automatic EZAir collar grid generation.

Chapter 9

Parallelization

EZAir adopts a hybrid parallelization model where it uses a shared memory model (OpenMP) within a machine and a distributed memory model (MPI) to facilitate the transfer of data between machines. Nevertheless, each machine must hold the entire geometry and geometry related information, mainly for the purpose of parallelizing the Chimera suite.

Within a machine, the domain (the geometry) is divided between the cores on a “line” level, while the division to the machines is conducted on the “block” level. That means that a single zone is not divided between two or more machines but it is divided between cores sharing the same memory.

See Section 10.12 for usage.

9.1 Section By Section Parallelism

9.1.1 Main Section

The main computational section that includes the block matrix inversions and the flux and Jacobian evaluation. These are the most crucial and computationally intensive parts of the code. These parts are parallelized using a unique and a novel way. As part of the initialization of the Chimera steps, the “live” cells are marked. A “live” cell is a cell that is evaluated during the time integration, as opposed for example to a

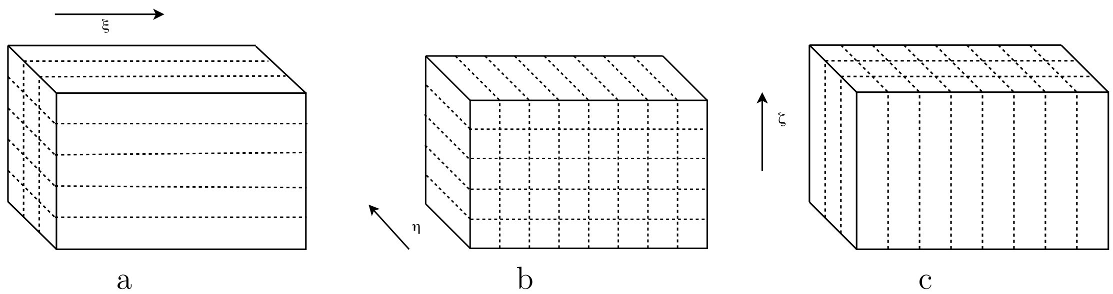


Figure 9.1: Single grid, lists of “live” cells for ξ, η, and ζ directions

“Chimera” cell which takes its value from the interpolation of other “live” cells. The structured grids makes it easy to construct straight lines based on the grid original coordinates. In three dimensions, there would be three lists of lines. A list of lines in the ξ direction, a list of lines in the η direction, and another list of lines in the ζ direction. As shown in Figure 9.1.1. The figure however, contains a single grid where all the cells are alive. It is important to mention that the lists contain the “live” cells only.

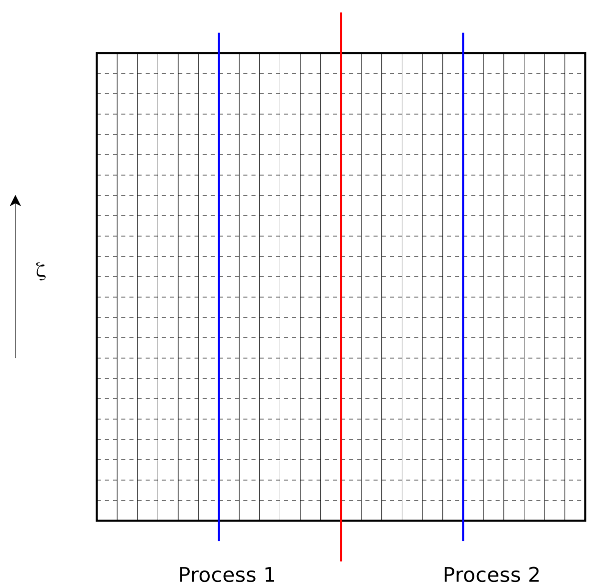


Figure 9.2: Ideal division of a 2D grid into groups of lines for two processes

These lists are then divided between all desired processes to groups so that the number of cells to which each process is responsible are even. Note that the parallel imbalance between the processes is determined by the length of the lines and in essence unlinks the parallel efficiency from the imbalance between grid sizes. However, this cell division does not guarantee that interface cells which have neighbors which are the responsibility of another process, do not have concurrent read and write operations from different processes. In order to achieve that, each one of the groups of lines are split into two parts. The first part of the group does not contain neighbors to the next process groups cells, and the second part does not contain neighbors to the previous process. Figure 9.2 shows an ideal division of a two-dimensional grid into a group of lists of lines in the ζ direction for two processes. The boundary between the processes is colored by red, and the boundary between the parts of each group is colored by blue. It is easily visible that no cell in the first part of the first process group of lines has any neighbor within the first part of the second process group of line. Similarly, no cell in the second part of the first process group of lines has any neighbor within the second part of the second process group of lines.

The desired operation which can be the flux evaluation, the Jacobian evaluation or any other operation is synchronized between the processes after the first part, and therefore no concurrent read and write from different processes to a single cell can occur. This is highly important since it enables the usage of the Gauss-Sidle scheme with no penalty.

9.1.2 Boundary Conditions

Boundary conditions are an essential part of any CFD code. In the present code all the boundary conditions are divided evenly between the processes of the machine regardless of the zone they belong to, and optimal parallelization is achieved.

9.1.3 I/O Operations

Input/output operations such as “file save” must be carried out from a single process per file or the results would be unexpected. The file saving procedure is synchronized between the processes/machines as best as possible. Bear in mind that the logging which is performed every iteration is also a "file save" operation, and it hampers the parallel efficiency. In contrast, the "file read" is parallelized, even if the grids are read from a single file.

9.1.4 Chimera Interpolation Seek

The Chimera module has several routines. They are all parallelized to be very efficient. Most of the operations must be performed sequentially grid after grid, but all the processes share the work. The parallelization scheme varies between task queue of cells to be scanned where each idle process is handled out the next task, or a comb-snake-like scan where each process is given in advance part of a grid.

Chapter 10

Input File, Run Preparation, and Execution

10.1 Preface

The EZAir suite of codes is driven through the command line and/or an input file (or a series of input files) with a certain syntax. An input file may be constructed using a simple text editor and/or the EZAir graphic user interface (GUI), EPPIC. The basic solver is invoked by typing the command (for shared memory architecture): “ezair-shr-icc [-f $includefile]... [options]...”. Alternatively, it can be invoked by typing the command (for distributed memory architecture, using a certain flavor of MPI): “mpirun [mpi options]... ezair-openmpi-mpicc [-f $includefile]... [options]... {-parallel # # }”

The content of the included files is exactly similar to the options that are available in the code. An up-to-date list of all available options may be obtained by invoking the command-line option “-help”. Note that an include file can include additional include files within. Recursive include is not checked and is not allowed. If no option is stated it is assumed that the input file for the basic solver is called ‘ezfile’ and therefore the default ‘-f ezfile’ is used.

Although the input files may be easily generated using the GUI, this chapter provides a description of the structure and building blocks of the input files. A typical input file is composed of 5 sections. “Simulation,” “Geometry,” “Solver,” “Movement,” and “Output” sections.

Listing 10.1 contains a simple, basic input file for a non-moving simulation of the ideal-gas flow about a two-dimensional airfoil. The grid is made of a single zone and the simulation is a turbulent steady state simulation.

```
######### SIMULATION #########
-sim save 1000 ./restart/
-sim detect yes
-sim bc noslip

######### SOLVER ########
-solve equationset laminar
-solve scheme lgs
-solve order uw3
-solve refarea 0.380278
-solve reflong 0.380278
-solve mach 0.734
-solve alpha 2.54
-solve iters 1500
-solve cfl 1 100 exponential 50

########## OUTPUT ##############
-log global per.iter default max(mt)
-p3d global node xyz q mt do end
-table pressure-coefficient node
  airfoil jmin i = 61,283
  vertical x cp do end

########## GEOMETRY ############
-body new rae2822 external
  -zone newp3d grid.x 1 airfoil
  -bc imax riemann
```

```
-bc imin riemann
-bc jmax riemann
```

Listing 10.1: Example of a simple EZAir input file

10.2 Scripting Language

The input file is a simple ASCII file with certain important rules. This chapter contains a detailed description of all the available command-line options. The syntax of each of the input options, or input file lines, follows the proceeding rules:

- The "# " symbol means that there is a comment until the next option. One may use as many comment lines as necessary.
- Each line starts with a directive (all directives start with the "-" sign).
- A directive may have no parameters, one parameter, or a few parameters.
- Parameters may be assigned values.

The proceeding description of the input options, or input file lines, makes use of the following symbols:

- The "$" symbol signifies a string input.
- The "#" symbol signifies a numerical input.
- The "|" symbol signifies a choice selection.
- The "..." symbol signifies an option that can be repeated.
- Input entries that are enclosed by curly brackets, { }, are required.
- Input entries that are enclosed by square brackets, [ ], are optional.

10.3 Auxiliary Options

In addition to the 5 above mentioned sections, there are auxiliary options that allow to control the actual run. They may be used in addition to any options or input files as they control the manner in which the specific run is executed. These are detailed in Table 10.1.

Auxiliary Options

Option	Description
-help	Prints the help (the same result may be obtained by using the options: -h, -H, -help, -HELP , /h , /help , /H , /HELP , /? , -?)
-new	Prints the new updates in the executable.
-n	Prints the commands that would be executed, but does not execute them.
-n1	Prints the boundary conditions and topology inputed and detected and exit.
-n2	Prints the boundary conditions and topology inputed and detected with extended information and exit.
-n3	Does not cut holes.
-n4	Cut chimera holes, without seeking interpolations and exit.
-n5	Cut chimera holes, and seek interpolations and exit.
-post	Skip all steps/iterations and jump to post-processing.
-dryrun	Skips flow-field calculation.

Table 10.1: Auxiliary options

10.4 Simulation Options

The simulation options provide global control options for the simulation. Certain settings serve as the defaults that can be overridden for specific zones using appropriate directives. Except for the “-prob” option, all the options that are related to the “Simulation” section start with the “-sim” directive. The options are listed in the following tables:

Prob	
Syntax	-prob $description
Description	Description of the simulation, The content of $description will be added to the log file.
Parameters	$description Simulation description.
Examples	-prob airfoil

Table 10.2: Simulation prob option

Echo	
Syntax	-echo $description
Description	EPPIC extended information that is added to the log file (For internal use only).
Parameters	$description EPPIC extended information.
Examples	-echo EPPIC

Table 10.3: Simulation echo option

Simulation Detect		
Syntax	-sim detect {yes | no}	
Description	Set the current state for automatic topology detection.	
Parameters	yes	Automatically detect geometry.
	no	Do not automatically detect the geometry (default).
Examples	-sim detect yes	

Table 10.4: Simulation detect option

Simulation Detect.tolerance		
Syntax	-sim detect.tolerance #tolerance	
Description	Set the tolerance [m] to use for automatic topology detection.	
Parameters	#tolerance	Tolerance in standard units (default value is 1.000000e-10 [m]).
Examples	-sim detect.tolerance 1e-10	

Table 10.5: Simulation detect.tolerance option

Simulation Wise.tolerance		
Syntax	-sim wise.tolerance #tolerance	
Description	Set the current state tolerance [m] to use for un/wise search. Use to by bypass errors of the form: "Unable to determine vertex orientation toward cell center."	
Parameters	#tolerance	Tolerance in standard units (default value is 2.220446e-16 [m]).
Examples	-sim wise.tolerance 2e-15	

Table 10.6: Simulation wise.tolerance option

Simulation Load		
Syntax	-sim load [[#specific-step] [$path [#former-step]]	
Description	When this option is activated, the simulation is loaded from $path. If path is not indicated, the files are loaded from the current folder. For dual time stepping, one can load a specific former step.	
Parameters	#specific-step	The specific step to restart from.
	$path	Folder from which to read the restart files.
	#former-step	Specific former step to be loaded (for dual-time stepping).
Examples	-sim load ./restart/	
	-sim load 5 ./restart/	

Table 10.7: Simulation load option

Simulation Save

Syntax	-sim {save | save.sequential | save.overwrite} #i [$path]
Description	Controls the number of iterations between saves of the simulation state. The default is 100. If path is not indicated, the files are saved to the current folder. Using save.sequential or save.overwrite add/discard the position in the filenames. The default is discard position for steady state, and add position for unsteady simulation.
Parameters	#i — Number of steps/iterations between saves. $path — Folder to save the restart files.
Examples	-sim save 1000 ./restart/ -sim save.sequential 1000 ./restart/ -sim save.overwrite 1000 ./restart/

Table 10.8: Simulation save option

Simulation Backup

Syntax	-sim backup #l [$backup-path]
Description	Controls the number of cyclic backup saves. No defaults. Each restart save is carried out, a backup save will be additionaly done to cyclic slot in the path given. The user can restore the backup by loading from the backup path with the #slot number (conducted through the line: -sim load #slot $backup-path).
Parameters	$backup-path — Backup folder.
Examples	-sim backup 2 ./backup/

Table 10.9: Simulation backup option

Store.Velocity.Averages	
Syntax	-sim store.velocity.averages
Description	Store the velocity averages.
Parameters	N/A
Examples	-sim store.velocity.averages

Table 10.10: Simulation store velocity averages option

Store.Resolved.Stress.Averages	
Syntax	-sim store.resolved.stress.averages
Description	Store the resolved stress averages.
Parameters	N/A
Examples	-sim store.resolved.stress.averages

Table 10.11: Simulation store resolved stress averages option

Store.Turbulent.Stress.Averages	
Syntax	-sim store.turbulent.stress.averages
Description	Store the turbulent stress averages.
Parameters	N/A
Examples	-sim store.turbulent.stress.averages

Table 10.12: Simulation store turbulent stress averages option

Store.Statistical.Averages	
Syntax	-sim store.statistical.averages
Description	Store statistical averages.
Parameters	N/A
Examples	-sim store.statistical.averages

Table 10.13: Simulation store statistical averages option

Simulation Wall.distance		
Syntax	-sim wall.distance {save | load} [$path]]	
Description	Save the wall-distance status or load the wall-distance (instead of re-evaluating it). Note that, at this time, there is limited verification that the wall distance that is loaded is consistent with the simulation. It is up to the user to make sure that it is indeed consistent.	
Parameters	save	Save the wall-distance status.
	load	Load the wall-distance.
	$path	Folder path.
Examples	-sim wall.distance save	
	-sim wall.distance load	

Table 10.14: Simulation wall.distance option

Simulation Dualtime		
Syntax	-sim {dualtime | dualtime.1st | dualtime.blend} #initial_rts {#steps [linear | exponential | tangential] #final_rts}...	
Description	Invoke dual-time stepping simulation. No defaults. See Section 4.7.	
Parameters	dualtime	Classical 2^{nd} order temporal accuracy (BDF2).
	dualtime.1st	1^{st} order temporal accuracy.
	dualtime.blend	Blended 1^{st} and 2^{nd} order accuracy in time. Makes use of the dualtime.blend.alpha parameter (see Table 10.196).
	#initial_rts	Initial real time step.
	#steps	Number of steps for which the real time step grows from #initial_rts to #final_rts.
	linear	Linear growth of the real time step.
	exponential	Exponential growth of the real time step.
	tangential	Hyperbolic tangent growth of the real time step.
	#final_rts	Final real time step.
Examples	-sim dualtime 1e-2 500	

Table 10.15: Simulation dualtime option

Simulation Unsteady	
Syntax	-sim unsteady #initial_rts {#iters [linear | exponential | tangential] #final_rts}...
Description	Invoke time-accurate simulation (1^{st} order temporal accuracy). No defaults. When used, all CFL numbers are ignored. See Section 4.6.3.
Parameters	#initial_rts — Initial real time step. #iters — Number of iterations for which the real time step grows from #initial_rts to #final_rts. linear — Linear growth of the real time step. exponential — Exponential growth of the real time step. tangential — Hyperbolic tangent growth of the real time step. #final_rts — Final real time step.
Examples	-sim unsteady 1e-2 500

Table 10.16: Simulation unsteady option

Simulation Format

Syntax	-sim format [{fmt| formatted} | {fmt-ms | formatted-ms} | {bin | binary} | {unf | unformatted} | {unf2 | unf8}] [{r8 | double} | {r4 | single}] [{i8 | i4}] [{ne | native.endian} | {re | reverse.endian} | {le | little.endian} | {be | big.endian}]	
Description	Sets the plot3d output format for xyz, q, and function files. Default is Plot3D, Binary, Single precision, Native endian.	
Parameters	fmt | formatted	Unix text format.
	fmt-ms | formatted-ms	MS-Dos text format.
	bin | binary	Binary format.
	unf | unformatted	Unformatted.
	unf2 | unf8	Obsolete unformatted.
	r8 | double	Real double precision.
	r4 | single	Real single precision.
	i4/i8	32/64 bit integer, use with care (default is 32 bit).
	ne | native.endian	Native endianness.
	re | reverse.endian	Reverse endianness.
	le | little.endian	Little endian.
	be | big.endian	Big endian.
Examples	-sim format r8	

Table 10.17: Simulation format option

Simulation BC

Syntax	-sim bc {noslip | impermeable | slip #sigma.u #sigma.T | none}	
Description	Selects the current wall boundary condition when "emesh" is automatically detected. An emesh is a mesh where a cut condition is identified or imposed, *e.g.*, a C type mesh about an airfoil (see Section 6.1).	
Parameters	noslip	no-slip boundary conditions at automatically-detected walls.
	impermeable	Impermeable boundary conditions at automatically-detected walls.
	slip	Isothermic, slip boundary conditions at all walls.
	#sigma.u	Slip boundary condition parameter affecting the velocity condition (σ_u). Suggested value: $\sigma_u = 1$.
	#sigma.T	Slip boundary condition parameter affecting the temperature condition (σ_T). Suggested value: $\sigma_u = 1$.
	none	No boundary conditions assigned to automatically-detected walls (default).
Examples	-sim bc noslip	
	-sim bc slip 1 1	

Table 10.18: Simulation bc option

Simulation Turbulent BC		
Syntax	-sim bc {omega.menter | omega.wilcox | automatic.wall.func.incompressible | nichols.wall.func}	
Description	Set the default ω boundary condition for k-ω turbulence models.	
Parameters	omega.menter	Menter style (classical) ω boundary condition (default).
	omega.wilcox	Wilcox style boundary conditions.
	auto...incompressible	Use automatic (incompressible) wall function.
	nichols.wall.func	Use Nichols wall function (compressible and isothermic).
Examples	-sim bc omega.menter	
	-sim bc automatic.wall.func.incompressible	

Table 10.19: Simulation turbulent bc option

Simulation Sigma BC		
Syntax	-sim bc {sigma.u sigma.T} #sigma	
Description	Set the default slip parameters.	
Parameters	#sigma.u	Slip boundary condition parameter affecting the velocity condition (σ_u). Suggested value: $\sigma_u = 1$.
	#sigma.T	Slip boundary condition parameter affecting the temperature condition (σ_T). Suggested value: $\sigma_u = 1$.
	#sigma	Value.
Examples	-sim bc sigma.u 1	

Table 10.20: Simulation Sigma bc option

Simulation Turbulent.wall.distance		
Syntax	-sim bc turbulent.wall.distance #turbulent-wall-distance	
Description	Set the turbulent wall distance (relevant only to all k-ω and Reynolds stress turbulence models).	
Parameters	#turbulent-wall-distance	Explicit wall distance.
Examples	-sim bc turbulent.wall.distance 1.0e-4	

Table 10.21: Simulation turbulent.wall.distance bc option

10.4.1 Automatic Chimera Module

The automatic Chimera module that is implemented in the EZAir suite of codes is composed of three key elements. The first is the automatic overlap grid minimization procedure (see Section 6.3.1). The second pertains to the automatic logical interpolation. And last, the automatic force overlap procedure. All three can be used together (see Table 10.22) or independently (see Table 10.23). This section contains a description of the directives that are used to control the automatic Chimera module.

Simulation Chimera Auto/Manual		
Syntax	-sim chimera {auto | manual}	
Description	Selects automatic or manual Chimera algorithms. Use with care: for recommended practice the reader is referred to Section 6.3.1.1.	
Parameters	auto	Automatic hole cutting.
	manual	Manual hole cutting (default).
Examples	-sim chimera auto	

Table 10.22: Simulation Chimera auto/manual option

Simulation Partial Automatic Chimera		
Syntax	-sim chimera {auto.overlap.minimization | auto.force.overlapping}	
Description	Selects partial automatic algorithms: automatic minimization or automatic force overlap.	
Parameters	auto.overlap.minimization	Enable automatic overlap minimization.
	auto.force.overlapping	Enable automatic exact force overlap.
Examples	-sim chimera auto.force.overlapping	

Table 10.23: Simulation automatic Chimera option

Simulation Chimera Suitability Options		
Syntax	-sim chimera {auto.suitability.distance | auto.suitability.volume | auto.suitability.max.diag | auto.suitability.min.diag | auto.suitability.live | auto.suitability.constant} #weight	
Description	Controls the suitability function (default is auto.suitability.max.diag 1).	
Parameters	auto.suitability.distance	Automatic hole cutting based on wall distance.
	auto.suitability.volume	Automatic hole cutting based on cell volume.
	auto.suitability.max.diag	Automatic hole cutting based on maximal cell diagonal.
	auto.suitability.min.diag	Automatic hole cutting based on minimal cell diagonal.
	auto.suitability.live	Add the 'live cell' flag (0/1) to the DSF (donor suitability function).
	auto.suitability.constant	Add a constant to the DSF.
	#weight	Weight of hole cutting criterion.
Examples	-sim chimera auto.suitability.distance 1	

Table 10.24: Simulation Chimera suitability option

Simulation Chimera Auto.wall.distance		
Syntax	-sim chimera auto.wall.distance {exact | quick | naive}	
Description	Select the algorithm for the wall-distance based suitability function calculation (default is naive).	
Parameters	exact	Use exact wall distance calculation for evaluating the suitability function.
	quick	Use regular wall distance calculation for evaluating the suitability function.
	naive	Use mesh based wall distance calculation (fastest) for evaluating the suitability function.
Examples	-sim chimera chimera auto.wall.distance exact	

Table 10.25: Simulation Chimera auto.wall.distance option

Simulation Chimera Logical Interpolation Options		
Syntax	-sim chimera {auto.length.tolerance | auto.angle.tolerance.deg | auto.angle.tolerance.rad} #tolerance [by.height | by.dsf]	
Description	Controls the tolerances for logical interpolation and force overlapping (becomes the default for all walls).	
Parameters	auto.length.tolerance	Defines tolerances for automatic force overlap and logical interpolation.
	auto.angle.tolerance.deg	Defines tolerances for automatic force overlap and logical interpolation.
	auto.angle.tolerance.rad	Defines tolerances for automatic force overlap and logical interpolation.
	by.height	Set interpolation strategy based on height.
	by.dsf	Set interpolation strategy based on donor suitability function.
	#tolerance	Defines tolerances for automatic force overlap and logical interpolation.
Examples	-sim chimera auto.length.tolerance 0.0005	
	-sim chimera auto.angle.tolerance.deg 90	

Table 10.26: Simulation Chimera global logical interpolation options

Simulation Chimera Force Overlap Options		
Syntax	-sim chimera {auto.force.distance | auto.force.surface | auto.force.index}	
Description	Sets the default force overlap criteria for all walls, (default is auto.force.distance).	
Parameters	auto.force.distance	Priority is given to the face closest to its' zone center.
	auto.force.surface	Priority is given to the face with the smaller area.
	auto.force.index	Priority is given to the face whose zone index is lower.
Examples	-sim chimera auto.force.surface	

Table 10.27: Simulation Chimera global force overlap options

Simulation Chimera Enclave Options

Syntax	-sim chimera {auto.enclave.match | auto.enclave.differ}	
Description	Select the elimination method of enclaved grid points that are created by the minimization algorithm.	
Parameters	auto.enclave.match	Remove enclaved grid points after the inner-formation hole cutting ("matching formation"). Therefore, it removes enclaved grid parts that are the result of inner-formation zones. Invoked only once.
	auto.enclave.differ	Remove enclaved grid points after outer-formation hole-cutting ("formation differ"). Therefore it removes all enclaved parts (this option is more expensive in terms of computation time). Invoked every time the Chimera module is invoked.
Examples	-sim chimera auto.enclave.differ	

Table 10.28: Simulation Chimera enclave option

Simulation Chimera Iterations Options		
Syntax	-sim chimera {auto.iters | auto.iters.match | auto.iters.differ} #iterations	
Description	Directly set the number of automatic minimization iterations. More iteration results in increased overlap. Do not use without guidance.	
Parameters	auto.iters	Perform for "match" and "differ" (see below).
	auto.iters.match	Perform only once (between zones belonging to the same formation).
	auto.iters.differ	Perform at every time step.
Examples	-sim chimera auto.iters.differ 2	

Table 10.29: Simulation Chimera iterations option

Simulation Chimera Algorithm		
Syntax	-sim chimera {auto.alg.global | auto.alg.front | auto.alg.front.tree}	
Description	Selects different search algorithm for the hole minimization process.	
Parameters	auto.alg.global	Uses all cell search algorithm (slowest but checks all the cells).
	auto.alg.front	Use "flood fill" type algorithm, starting from the Chimera front.
	auto.alg.front.tree	Use "flood fill" type algorithm that utilizes a pre-calculated alternating digital tree (ADT). It is the fastest but it requires additional memory (default).
Examples	-sim chimera auto.alg.global	

Table 10.30: Simulation Chimera algorithm option

Simulation Chimera Leaf size		
Syntax	-sim chimera auto.leaf.size #size	
Description	Selects the maximal leaf size on the tree based algorithm. The lower it is, the faster is the algorithm but it requires more memory. Must be larger to or equal to 1 (default is 3).	
Parameters	#size	Leaf size.
Examples	-sim chimera auto.leaf.size 4	

Table 10.31: Simulation Chimera leaf option

Simulation Chimera Opt.dsf		
Syntax	-sim chimera opt.dsf {tight | near | far}	
Description	Select the optimized dsf cell eligibility. For backward compatibility default is tight; Recommended use: near or far.	
Parameters	tight	Consider only local cell dsf (donor suitability function).
	near	Direct neighbors must be live cells (for the current cell to be considered).
	far	Two neighboring cells must be live (for the current cell to be considered.
Examples	-sim chimera opt.dsf near	

Table 10.32: Simulation Chimera opt.dsf option

10.4.2 Simulation Chimera Options

Chimera options can be globally set using the directives that are described in the current section. Some of the settings can be changed for each zone by using the directives that are described in Section 10.5.4.

Simulation Chimera.tolerance		
Syntax	-sim chimera.tolerance #tolerance	
Description	Set the tolerance [m] to use for Chimera search.	
Parameters	#tolerance	Tolerance in standard units (default value is 2.220668e-12 [m]).
Examples	-sim chimera.tolerance 2e-11	

Table 10.33: Simulation chimera.tolerance option

Simulation Chimera		
Syntax	-sim chimera [minimum # | layers # | allow.1st | force.zombie.freestream | leave.zombie.unhandled | allow.inner.interpolation | allow.freestream.interpolation | allow.unhandled.interpolation | disable.interpolation]	
Description	Define Chimera options.	
Parameters	minimum #	Minimum cell cloud for interpolation
	layers #	The number of additional chimera fronts. If 1 is entered then it assumes also allow.1st.
	allow.1st	Allows for a single cell chimera front, where face values are evaluated in 1st order.
	force.zombie...	Force zombies to initialize with free-stream values.
	leave.zombie...	Leave the zombies un-initialized (with free-stream initial initialization).
	allow.inner...	Allow interpolation from inner neigbhouring cells.
	allow.free...	Allow interpolation from free stream.
	allow.unhandled...	Allow unhandled interpolations.
	disable.interpolation	Disables all interpolations and keep Chimera front cells unchanged.
Examples	-sim chimera allow.inner.interpolation -sim chimera allow.freestream.interpolation -sim chimera layers 1	

Table 10.34: Simulation chimera options

Simulation Interpolation Options

Syntax	-sim chimera [shepard1 | shepard2 | microsphere1 | microsphere2]	
Description	Set the interpolation scheme.	
Parameters	shepard1	Shepard weighted interpolation scheme based on the inverse of the distance (default).
	shepard2	Shepard weighted interpolation scheme based on the inverse of the distance squared.
	microsphere1	Micro-sphere weighted interpolation scheme based on the inverse of the distance (designed to minimize the bias of the Shepard scheme).
	microsphere2	Micro-sphere weighted interpolation scheme based on the inverse of the distance squared.
Examples	-sim chimera microsphere1	

Table 10.35: Simulation Chimera interpolation options

Simulation Chimera Auto.logical.interpolation

Syntax	-sim chimera auto.logical.interpolation [by.height | by.dsf]	
Description	Select logical interpolation and priority strategy (default is by.height).	
Parameters	by.height	Set interpolation strategy based on height.
	by.dsf	Set interpolation strategy based on donor suitability function.
Examples	-sim chimera auto.logical.interpolation by.dsf	

Table 10.36: Simulation Chimera auto.logical.interpolation option

Simulation Complementary Interpolation Options		
Syntax	-sim chimera [tri.linear | tri.linear.or.logical]	
Description	Set the complementary interpolation scheme.	
Parameters	tri.linear	Tri-linear interpolation scheme (attempts to use trilinear interpolation where available).
	tri.linear.or.logical	Tri-linear interpolation scheme or logical (use either trilinear interpolation or logical interpolation but not both; trilinear has precedence).
Examples	-sim chimera tri.linear	

Table 10.37: Simulation Chimera complementary interpolation options

Simulation Chimera Surface.cut		
Syntax	-sim chimera surface.cut {yes | no | far | near | tight}	
Description	Add surface hole cutting.	
Parameters	yes	Enable surface hole cutting (same as far, default).
	no	Disable surface hole cutting. See Figures 6.3 and 6.4 for a description of the resulting holes.
	far	See Figures 6.3 and 6.5 for a description of the resulting holes.
	near	See Figures 6.3 and 6.6 for a description of the resulting holes.
	tight	See Figures 6.3 and 6.7 for a description of the resulting holes.
Examples	-sim chimera surface.cut no	

Table 10.38: Simulation Chimera surface.cut option

Simulation Chimera Save/Load		
Syntax	-sim chimera {save | load | reload} [$path]	
Description	Save or load Chimera holes. Saving the Chimera holes provides the means to view the exact polygons of the automatic force overlap in EPPIC. Note that, at this time, there is limited verification that the Chimera data that is loaded is consistent with the simulation. It is up to the user to make sure that it is indeed consistent.	
Parameters	save	Save the Chimera holes.
	load	Load the Chimera holes. If the information (or a part of it) is missing then the program stops and issues an error.
	reload	Load the Chimera holes. If the information (or a part of it) is missing then the program recalculates the missing information and continues the run.
	$path	Path for saved/loaded Chimera hole information.
Examples	-sim chimera save ./restart	
	-sim chimera load ./restart	

Table 10.39: Simulation chimera save/load options

Simulation Save Reconstruct		
Syntax	-sim save.reconstruct $path	
Description	Save 'reconstruction' files, so the simulation can start with different grids; pay attention: the 'reconstruction folder' will be overwritten.	
Parameters	$path	Save reconstruct path.
Examples	-sim save.reconstruct /my_path	

Table 10.40: Simulation save.reconstruct option

Simulation Load Reconstruct		
Syntax	-sim load.reconstruct $path	
Description	Load 'reconstruction' files, and reconstruct the solution so the simulation can start with different grid and solution; pay attention: the 'save folder' will be overwritten with restart & load files.	
Parameters	$path	Load reconstruct path.
Examples	-sim load.reconstruct /my_path	

Table 10.41: Simulation load.reconstruct option

10.5 Geometry Options

The geometry options section are divided into subsections. These subsections are: "Formation," "Body," "Zone," "Chimera," "Phantom," "Shape," and "Vertex" options.

10.5.1 Formation Options

New		
Syntax	-formation new $formation	
Description	Creates a new formation with the name $formation. It starts the scope of a newly created formation. All subsequent bodies will be added to this formation. Formations are required for DOF simulations (see Section 10.9.1).	
Parameters	$formation	Name of formation.
Examples	-formation new aircraft	

Table 10.42: Geometry formation new option

Refarea		
Syntax	-formation refarea #area	
Description	Set formation reference area.	
Parameters	#area	Formation reference area.
Examples	-formation refarea 12.3	

Table 10.43: Geometry formation refarea option

Relen		
Syntax	-formation reflen #x #y #z	
Description	Set formation reference lengths (x, y, and z directions are in body coordinates).	
Parameters	#x	Formation reference length x coordinate direction (body coordinates).
	#y	Formation reference length y coordinate direction (body coordinates).
	#z	Formation reference length z coordinate direction (body coordinates).
Examples	-formation reflen 1.0 0.5 2.3	

Table 10.44: Geometry formation reflen option

Refvel		
Syntax	-formation refvel #vel	
Description	Set formation reference velocity.	
Parameters	#vel	Formation reference velocity.
Examples	-formation refvel 230	

Table 10.45: Geometry formation refvel option

Refrho		
Syntax	-formation refrho #rho	
Description	Set formation reference density.	
Parameters	#rho	Formation reference density.
Examples	-formation refrho 0.95	

Table 10.46: Geometry formation refrho option

Refpoint		
Syntax	-formation refpoint $vertex	
Description	Set formation reference point (for moments calculations).	
Parameters	$vertex	Formation reference point.
Examples	-formation refpoint p1	

Table 10.47: Geometry formation refpoint option

Clear	
Syntax	-formation clear
Description	Ends the current formation scope.
Parameters	N/A
Examples	-formation clear

Table 10.48: Geometry formation clear option

Offset		
Syntax	-formation offset #ox #oy #oz	
Description	Performs a simple translation to the nodes of the formation bodies as well as the vertexes (as if the corresponding input grid file zones were translated). If a node is (x, y, z) it will result in ($x + ox$, $y + oy$, $z + oz$).	
Parameters	#ox	x coordinate direction translation.
	#oy	y coordinate direction translation.
	#oz	z coordinate direction translation.
Examples	-formation offset 10.0 0.0 0.0	

Table 10.49: Geometry formation offset option

Rotate		
Syntax	-formation {rotate.x.rad | rotate.y.rad | rotate.z.rad | rotate.x.deg | rotate.y.deg | rotate.z.deg} #a	
Description	Rotates nodes for the formation zones as well as the vertexes about a main axis (as if the corresponding input grid file zones were rotated). Each of the options pertains to one coordinate direction.	
Parameters	#a	Rotation angle
Examples	-formation rotate.y.deg 90	
	-formation rotate.z.rad 0.5	

Table 10.50: Geometry formation rotate options

Scale		
Syntax	-formation scale #sx #sy #sz	
Description	Performs simple scaling to the nodes of the formation bodies as well as the vertexes as if the corresponding input grid file zones were scaled. If a node is (x, y, z) it will result in ($sx * x$, $sy * y$, $sz * z$)	
Parameters	#sx	x coordinate direction scaling factor.
	#sy	y coordinate direction scaling factor.
	#sz	z coordinate direction scaling factor.
Examples	-formation scale 2.0 2.0 2.0	

Table 10.51: Geometry formation scale option

Matrix		
Syntax	-formation matrix #a11 #a12 #a13 #a21 #a22 #a23 #a31 #a32 #a33	
Description	Performs matrix multiplication to the nodes of the formation bodies as well as the vertexes (as if the corresponding input grid file zones were transformed accordingly). If a node is $(x,\ y,\ z)$ and $A = [a_{ij}]$, it will result in $A * (x, y, z)$	
Parameters	#a11	Transformation matrix a_{11} element
	#a12	Transformation matrix a_{12} element
	#a13	Transformation matrix a_{13} element
	#a21	Transformation matrix a_{21} element
	#a22	Transformation matrix a_{22} element
	#a23	Transformation matrix a_{23} element
	#a31	Transformation matrix a_{31} element
	#a32	Transformation matrix a_{32} element
	#a33	Transformation matrix a_{33} element
Examples	-formation matrix 1 0 0 0 1 0 0 0 1	

Table 10.52: Geometry formation matrix option

Dilute		
Syntax	-formation {dilute.i | dilute.j | dilute.k}	
Description	Dilutes all formation zones in a specific dimension. Namely, remove every other grid point in the corresponding direction.	
Parameters	dilute.i	Dilute the grid points in the ξ coordinate direction.
	dilute.j	Dilute the grid points in the η coordinate direction.
	dilute.k	Dilute the grid points in the ζ coordinate direction.
Examples	-formation dilute.i	

Table 10.53: Geometry formation dilute options

Formation redistribute.equal		
Syntax	-formation {redistribute.equal.i | redistribute.equal.j | redistribute.equal.k} #n	
Description	Redistribute all formation zones in a specific dimension using equal spacing.	
Parameters	redistribute.equal.i	Redistribute the grid points in the ξ coordinate direction.
	redistribute.equal.j	Redistribute the grid points in the η coordinate direction.
	redistribute.equal.k	Redistribute the grid points in the ζ coordinate direction.
	#n	New dimension.
Examples	-formation redistribute.equal.i 101	

Table 10.54: Geometry formation redistribute options (equal spacing)

Formation redistribute.tanh (single sided)		
Syntax	-formation {redistribute.tanh.imin | redistribute.tanh.jmin | redistribute.tanh.kmin | redistribute.tanh.imax | redistribute.tanh.jmax | redistribute.tanh.kmax} #n #d	
Description	Redistribute all formation zones in a specific dimension in a single-sided hyperbolic tangent distribution.	
Parameters	redistribute.tanh.imin	Redistribute the grid points in the ξ coordinate direction.
	redistribute.tanh.jmin	Redistribute the grid points in the η coordinate direction.
	redistribute.tanh.kmin	Redistribute the grid points in the ζ coordinate direction.
	redistribute.tanh.imax	Redistribute the grid points in the ξ coordinate direction.
	redistribute.tanh.jmax	Redistribute the grid points in the η coordinate direction.
	redistribute.tanh.kmax	Redistribute the grid points in the ζ coordinate direction.
	#n	New dimension.
	#d	New edge spacing.
Examples	-formation redistribute.tanh.imin 101 0.0001	

Table 10.55: Geometry formation redistribute tanh options (single sided)

Formation redistribute.tanh (two sided)		
Syntax	-formation {redistribute.tanh.i | redistribute.tanh.j | redistribute.tanh.k} #n #d-min #d-max	
Description	Redistribute all formation zones in a specific dimension in a single-sided or two-sided hyperbolic tangent distribution.	
Parameters	redistribute.tanh.i	Redistribute the grid points in the ξ coordinate direction (two-sided).
	redistribute.tanh.j	Redistribute the grid points in the η coordinate direction (two-sided).
	redistribute.tanh.k	Redistribute the grid points in the ζ coordinate direction (two-sided).
	#n	New dimension.
	#d-min	New edge spacing.
	#d-max	New edge spacing.
Examples	-formation redistribute.tanh.i 101 0.001 0.000012	

Table 10.56: Geometry formation redistribute tanh options (two sided)

CNV		
Syntax	-formation {cnv.i | cnv.j | cnv.k}	
Description	Reverse the grid point ordering of all formation zones in a specific dimension.	
Parameters	cnv.i	Reverse the grid points in the ξ coordinate direction.
	cnv.j	Reverse the grid points in the η coordinate direction.
	cnv.k	Reverse the grid points in the ζ coordinate direction.
Examples	-formation cnv.i	

Table 10.57: Geometry formation cnv options

10.5.2 Body Options

The definition of a body is brought in Section 6.2. The reader is referred to this section for better understanding of the role of bodies in the EZAir suite of codes.

New		
Syntax	-body new $body {external | internal} [multiblock]	
Description	Creates a new body with the name $body. A body is declared external if the flow is external to its entities A body is declared internal if the flow is internal to its cells. When a body is declared multiblock then search for patched zones. If declared outside explicit formation scope, is added to 'default' formation. This option starts the newly added body scope. Passive Chimera directives (*e.g.*, "cut.me.with" or "do.not.cut.me.with") are usually invoked in the scope of bodies.	
Parameters	$body	Name of the body
	external	The body cuts holes assuming that the flow is external to the body.
	internal	The body cuts holes assuming that the flow is internal to the body. The body cuts holes in everything that is not contained within the body grid.
	multiblock	Search for multi-block grids inside the body
Examples	-body new fuselage external	

Table 10.58: Geometry body new option

Clear	
Syntax	-body clear
Description	Ends the current body scope.
Parameters	N/A
Examples	-body clear

Table 10.59: Geometry body clear option

Set		
Syntax	-body set $body	
Description	Reopens a body scope.	
Parameters	$body	The name of the body.
Examples	-body set airfoil-body	

Table 10.60: Geometry body set option

Offset		
Syntax	-body offset #ox #oy #oz	
Description	Performs a simple translation to the nodes of the body zones (as if the corresponding input grid file zones were translated). If a node is (x, y, z) it will result in $(x + ox, y + oy, z + oz)$.	
Parameters	#ox	x coordinate direction translation.
	#oy	y coordinate direction translation.
	#oz	z coordinate direction translation.
Examples	-body offset 10.0 0.0 0.0	

Table 10.61: Geometry body offset option

Rotate		
Syntax	-body {rotate.x.rad | rotate.y.rad | rotate.z.rad | rotate.x.deg | rotate.y.deg | rotate.z.deg} #a	
Description	Rotates nodes of the body zones about a main axis (as if the corresponding input grid file zones were rotated). Each of the options pertains to one coordinate direction.	
Parameters	#a	Rotation angle
Examples	-body rotate.y.deg 90	
	-body rotate.z.rad 0.5	

Table 10.62: Geometry body rotate options

Scale		
Syntax	-body scale #sx #sy #sz	
Description	Performs simple scaling to the nodes of the body zones (as if the corresponding input grid file zones were scaled). If a node is (x, y, z) it will result in ($sx * x$, $sy * y$, $sz * z$)	
Parameters	#sx	x coordinate direction scaling factor.
	#sy	y coordinate direction scaling factor.
	#sz	z coordinate direction scaling factor.
Examples	-body scale 2.0 2.0 2.0	

Table 10.63: Geometry body scale option

Matrix		
Syntax	-body matrix #a11 #a12 #a13 #a21 #a22 #a23 #a31 #a32 #a33	
Description	Performs matrix multiplication to the nodes of the body zones (as if the corresponding input grid file zones were transformed accordingly). If a node is (x, y, z) and $A = [a_{ij}]$, it will result in $A*(x, y, z)$	
Parameters	#a11	Transformation matrix a_{11} element
	#a12	Transformation matrix a_{12} element
	#a13	Transformation matrix a_{13} element
	#a21	Transformation matrix a_{21} element
	#a22	Transformation matrix a_{22} element
	#a23	Transformation matrix a_{23} element
	#a31	Transformation matrix a_{31} element
	#a32	Transformation matrix a_{32} element
	#a33	Transformation matrix a_{33} element
Examples	-body matrix 1 0 0 0 1 0 0 0 1	

Table 10.64: Geometry body matrix option

Dilute		
Syntax	-body {dilute.i | dilute.j | dilute.k}	
Description	Dilutes all body zones in a specific dimension. Namely, remove every other grid point in the corresponding direction.	
Parameters	dilute.i	Dilute the grid points in the ξ coordinate direction.
	dilute.j	Dilute the grid points in the η coordinate direction.
	dilute.k	Dilute the grid points in the ζ coordinate direction.
Examples	-body dilute.i	

Table 10.65: Geometry body dilute options

Body redistribute.equal		
Syntax	-body {redistribute.equal.i | redistribute.equal.j | redistribute.equal.k} #n	
Description	Redistribute all body zones in a specific dimension using equal spacing.	
Parameters	redistribute.equal.i	Redistribute the grid points in the ξ coordinate direction.
	redistribute.equal.j	Redistribute the grid points in the η coordinate direction.
	redistribute.equal.k	Redistribute the grid points in the ζ coordinate direction.
	#n	New dimension.
Examples	-body redistribute.equal.i 101	

Table 10.66: Geometry body redistribute options (equal spacing)

Body redistribute.tanh (single sided)		
Syntax	-formation {redistribute.tanh.imin | redistribute.tanh.jmin | redistribute.tanh.kmin | redistribute.tanh.imax | redistribute.tanh.jmax | redistribute.tanh.kmax} #n #d	
Description	Redistribute all body zones in a specific dimension in a single-sided hyperbolic tangent distribution.	
Parameters	redistribute.tanh.imin	Redistribute the grid points in the ξ coordinate direction.
	redistribute.tanh.jmin	Redistribute the grid points in the η coordinate direction.
	redistribute.tanh.kmin	Redistribute the grid points in the ζ coordinate direction.
	redistribute.tanh.imax	Redistribute the grid points in the ξ coordinate direction.
	redistribute.tanh.jmax	Redistribute the grid points in the η coordinate direction.
	redistribute.tanh.kmax	Redistribute the grid points in the ζ coordinate direction.
	#n	New dimension.
	#d	New edge spacing.
Examples	-body redistribute.tanh.imin 101 0.0001	

Table 10.67: Geometry body redistribute tanh options (single sided)

Body redistribute.tanh (two sided)

Syntax	-body {redistribute.tanh.i | redistribute.tanh.j | redistribute.tanh.k} #n #d-min #d-max
Description	Redistribute all body zones in a specific dimension in a single-sided or two-sided hyperbolic tangent distribution.
Parameters	redistribute.tanh.i — Redistribute the grid points in the ξ coordinate direction (two-sided). redistribute.tanh.j — Redistribute the grid points in the η coordinate direction (two-sided). redistribute.tanh.k — Redistribute the grid points in the ζ coordinate direction (two-sided). #n — New dimension. #d-min — New edge spacing. #d-max — New edge spacing.
Examples	-body redistribute.tanh.i 101 0.001 0.000012

Table 10.68: Geometry body redistribute tanh options (two sided)

CNV		
Syntax	-body {cnv.i | cnv.j | cnv.k}	
Description	Reverse the grid point ordering of all body zones in a specific dimension.	
Parameters	cnv.i	Reverse the grid points in the ξ coordinate direction.
	cnv.j	Reverse the grid points in the η coordinate direction.
	cnv.k	Reverse the grid points in the ζ coordinate direction.
Examples	-body cnv.i	

Table 10.69: Geometry body cnv options

10.5.3 Zone Options

Newp3d		
Syntax	-zone newp3d $plot3dfile #index $zone	
Description	Detects the binary/ascii format and loads a single zone from a Plot3D file. Index is a single based index of the zone in the Plot3D file (therefore, the first zone is #1). Performance is extremely better for non-formatted (and hence binary or unformatted) files. This directive starts a zone scope.	
Parameters	$plot3dfile	Full path of the Plot3D file.
	#index	The number of the zone within the Plot3D file.
	$zone	Name of the zone.
Examples	-zone newp3d grid.x 3 left_wing	

Table 10.70: Geometry zone newp3d option

Collar		
Syntax	-zone collar {project $bodyzone {i | j | k} = #min, #max} {domain $wingzone {i | j | k} = #min, #max} {range {i | j | k} = #min, #max} {align {i | j | k} = #min, #max} [extrap # | elongate #]	
Description	Creates a collar grid based on a body-wing configuration (see Figure 10.1). It also starts a zone scope.	
Parameters	project	The ranges of domain to project onto.
	$bodyzone	Zone name of the body.
	i, j, k	Keywords for ξ, η and ζ coordinates.
	#min, #max	Index values.
	domain	The domain of the wing zone that is projected onto the body.
	$wingzone	Zone name of the wing.
	align	Set the outward range of the wing zone domain.
	extrap	Uses extrapolation if the wing zone domain does not contain all the body zone domain.
	elongate	Extrapolates the body zone edges.
Examples	-zone collar c_f-w project fuselage k = 1,60 domain wing i = 1,30 range j = 1,61 align k = 1,60 extrap 50 elongate 50	

Table 10.71: Geometry zone collar option

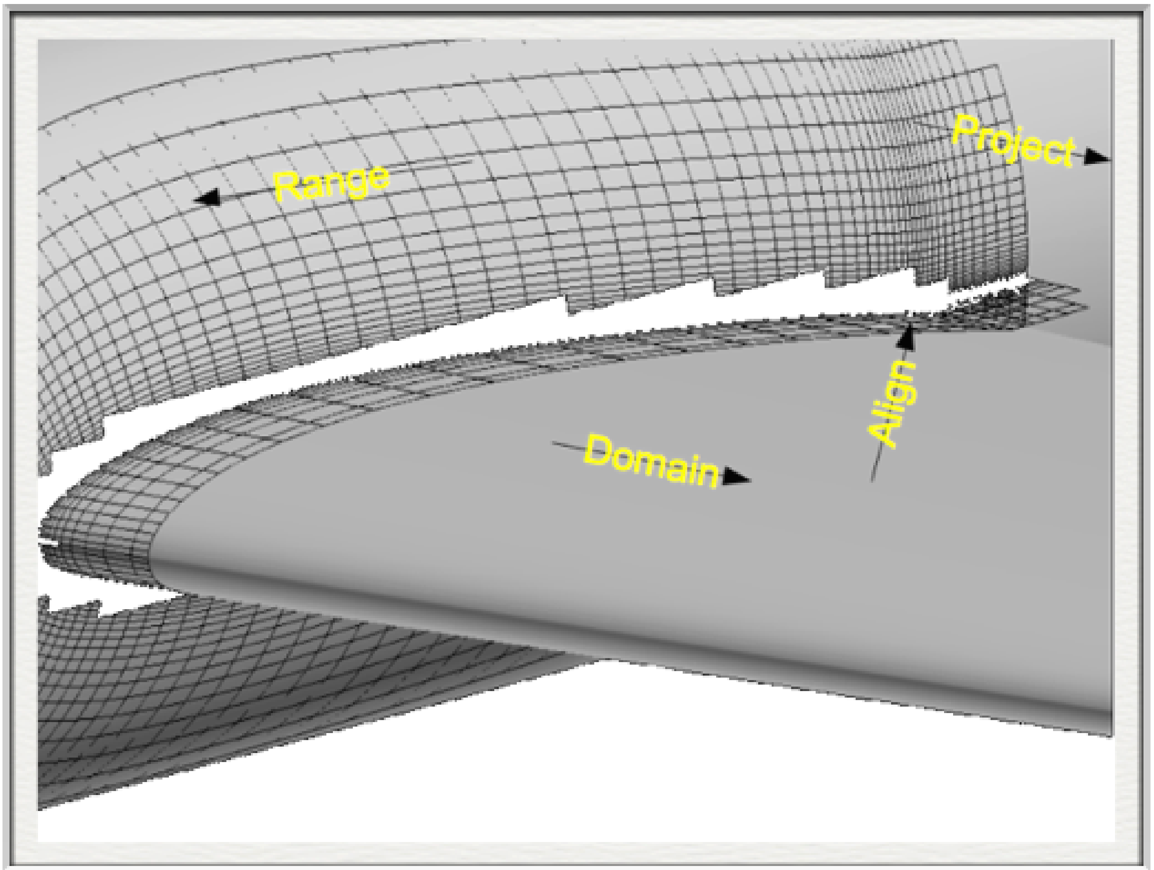


Figure 10.1: Example of collar grid generation
See Table 10.71 for a description of the parameters

Set		
Syntax	-zone set $zone	
Description	Reopens a zone scope.	
Parameters	$zone	The name of the zone.
Examples	-zone set wing1	

Table 10.72: Geometry zone set option

Crop		
Syntax	-zone crop #imin #imax #jmin jmax #kmin #kmax	
Description	Crops a zone (as if the corresponding input grid file zone was cropped, #imin #imax #jmin #jmax #kmin #kmax define the edges of the remaining part).	
Parameters	#imin, #imax	Crop i direction limits.
	#jmin, #jmax	Crop j direction limits.
	#kmin, #kmax	Crop k direction limits.
Examples	-zone crop 1 57 1 61 1 44	

Table 10.73: Geometry zone crop option

Dilute		
Syntax	-zone {dilute.i | dilute.j | dilute.k}	
Description	Dilutes all zone zones in a specific dimension. Namely, remove every other grid point in the corresponding direction.	
Parameters	dilute.i	Dilute the grid points in the ξ coordinate direction.
	dilute.j	Dilute the grid points in the η coordinate direction.
	dilute.k	Dilute the grid points in the ζ coordinate direction.
Examples	-zone dilute.i	

Table 10.74: Geometry zone dilute options

Zonal redistribute.equal		
Syntax	-zone {redistribute.equal.i | redistribute.equal.j | redistribute.equal.k} #n	
Description	Redistribute a zone in a specific dimension using equal spacing.	
Parameters	redistribute.equal.i	Redistribute the grid points in the ξ coordinate direction.
	redistribute.equal.j	Redistribute the grid points in the η coordinate direction.
	redistribute.equal.k	Redistribute the grid points in the ζ coordinate direction.
	#n	New dimension.
Examples	-zone redistribute.equal.i 101	

Table 10.75: Geometry zonal redistribute options (equal spacing)

Zonal redistribute.tanh (single sided)		
Syntax	-zone {redistribute.tanh.imin | redistribute.tanh.jmin | redistribute.tanh.kmin | redistribute.tanh.imax | redistribute.tanh.jmax | redistribute.tanh.kmax} #n #d	
Description	Redistribute a zone in a specific dimension in a single-sided hyperbolic tangent distribution.	
Parameters	redistribute.tanh.imin	Redistribute the grid points in the ξ coordinate direction.
	redistribute.tanh.jmin	Redistribute the grid points in the η coordinate direction.
	redistribute.tanh.kmin	Redistribute the grid points in the ζ coordinate direction.
	redistribute.tanh.imax	Redistribute the grid points in the ξ coordinate direction.
	redistribute.tanh.jmax	Redistribute the grid points in the η coordinate direction.
	redistribute.tanh.kmax	Redistribute the grid points in the ζ coordinate direction.
	#n	New dimension.
	#d	New edge spacing.
Examples	-zone redistribute.tanh.imin 101 0.0001	

Table 10.76: Geometry zonal redistribute tanh options (single sided)

Zonal redistribute.tanh (two sided)		
Syntax	-zone {redistribute.tanh.i | redistribute.tanh.j | redistribute.tanh.k} #n #d-min #d-max	
Description	Redistribute a zone in a specific dimension in a single-sided or two-sided hyperbolic tangent distribution.	
Parameters	redistribute.tanh.i	Redistribute the grid points in the ξ coordinate direction (two-sided).
	redistribute.tanh.j	Redistribute the grid points in the η coordinate direction (two-sided).
	redistribute.tanh.k	Redistribute the grid points in the ζ coordinate direction (two-sided).
	#n	New dimension.
	#d-min	New edge spacing.
	#d-max	New edge spacing.
Examples	-zone redistribute.tanh.i 101 0.001 0.000012	

Table 10.77: Geometry zonal redistribute tanh options (two sided)

CNV		
Syntax	-zone {cnv.i | cnv.j | cnv.k}	
Description	Reverse the grid point ordering of all zone zones in a specific dimension.	
Parameters	cnv.i	Reverse the grid points in the ξ coordinate direction.
	cnv.j	Reverse the grid points in the η coordinate direction.
	cnv.k	Reverse the grid points in the ζ coordinate direction.
Examples	-zone cnv.i	

Table 10.78: Geometry zone cnv options

Offset		
Syntax	-zone offset #ox #oy #oz	
Description	Performs a simple translation to the nodes of the zone (as if the corresponding input grid file zone was translated). If a node is (x, y, z) it will result in ($x + ox$, $y + oy$, $z + oz$).	
Parameters	#ox	x coordinate direction translation.
	#oy	y coordinate direction translation.
	#oz	z coordinate direction translation.
Examples	-zone offset 10.0 0.0 0.0	

Table 10.79: Geometry zone offset option

Rotate		
Syntax	-zone {rotate.x.rad | rotate.y.rad | rotate.z.rad | rotate.x.deg | rotate.y.deg | rotate.z.deg} #a	
Description	Rotates nodes of the zone about a main axis (as if the corresponding input grid file zone was rotated). Each of the options pertains to one coordinate direction.	
Parameters	#a	Rotation angle
Examples	-zone rotate.y.deg 90	
	-zone rotate.z.rad 0.5	

Table 10.80: Geometry zone rotate options

Scale		
Syntax	-zone scale #sx #sy #sz	
Description	Performs simple scaling to the nodes of the zone (as if the corresponding input grid file zone was scaled). If a node is (x, y, z) it will result in ($sx * x$, $sy * y$, $sz * z$)	
Parameters	#sx	x coordinate direction scaling factor.
	#sy	y coordinate direction scaling factor.
	#sz	z coordinate direction scaling factor.
Examples	-zone scale 2.0 2.0 2.0	

Table 10.81: Geometry zone scale option

Matrix		
Syntax	-zone matrix #a11 #a12 #a13 #a21 #a22 #a23 #a31 #a32 #a33	
Description	Performs matrix multiplication to the nodes of the zone (as if the corresponding input grid file zone was transformed accordingly). If a node is (x, y, z) and $A = [a_{ij}]$, it will result in $A * (x, y, z)$	
Parameters	#a11	Transformation matrix a_{11} element
	#a12	Transformation matrix a_{12} element
	#a13	Transformation matrix a_{13} element
	#a21	Transformation matrix a_{21} element
	#a22	Transformation matrix a_{22} element
	#a23	Transformation matrix a_{23} element
	#a31	Transformation matrix a_{31} element
	#a32	Transformation matrix a_{32} element
	#a33	Transformation matrix a_{33} element
Examples	-zone matrix 1 0 0 0 1 0 0 0 1	

Table 10.82: Geometry zone matrix option

Detect		
Syntax	-zone detect {yes | no}	
Description	Sets the current zone detection mode. Overrides "-sim detect"	
Parameters	yes	Use automatic topology detection for this zone.
	no	Do not use automatic topology detection for this zone.
Examples	-zone detect yes	

Table 10.83: Geometry zone detect option

Detect.tolerance	
Syntax	-zone detect.tolerance #tolerance
Description	Sets the current zone tolerance for automatic topology detection. Overrides -sim tolerance.
Parameters	#tolerance
Examples	-zone detect.tolerance 2e-7

Table 10.84: Geometry zone detect.tolerance option

Chimera.tolerance	
Syntax	-zone chimera.tolerance #tolerance
Description	Set the current zone tolerance for Chimera search. Overrides -sim chimera.tolerance (see Table 10.33).
Parameters	#tolerance
Examples	-zone chimera.tolerance 2e-7

Table 10.85: Geometry zone chimera.tolerance option

Wise.tolerance	
Syntax	-zone wise.tolerance #tolerance
Description	Set the current zone tolerance for un/wise cell search. Use to by bypass errors of the form: "Unable to determine vertex orientation toward cell center." Overrides -sim wise.tolerance (see Table 10.6).
Parameters	#tolerance
Examples	-zone wise.tolerance 2e-14

Table 10.86: Geometry zone wise.tolerance option

BC	
Syntax	-zone bc {noslip | impermeable | slip | none}
Description	Selects the current wall boundary condition when "emesh" is automatically detected. An emesh is a mesh where a cut condition is identified or imposed, *e.g.*, a C type mesh about an airfoil. Overrides "-sim bc" (see Table 10.18 and Section 6.1)
Parameters	noslip — Adiabatic, no-slip boundary conditions at all walls. impermeable — Adiabatic, impermeable boundary conditions at all walls. slip — Isothermic, slip boundary conditions at all walls. none — No boundary conditions assigned to all walls.
Examples	-zone bc noslip

Table 10.87: Geometry zone bc option

Order		
Syntax	-zone order {1st | uw2 | uw3}	
Description	Defines the order to use for the specific zone (default is determined by the solve option). The zone order must be lower than the solve order (see Table 10.175 for setting of the solve order). The reader is referred to Section 4.2.7 for a description of high order schemes.	
Parameters	1st	1^{st} order (default).
	uw2	Upwind 2^{nd} order.
	uw3	Upwind 3^{rd} order biased.
Examples	-zone order uw3	

Table 10.88: Geometry zone order option

Order (MP-4)		
Syntax	-zone order {mp4.cons | mp4.prim | mp4.char}	
Description	Use MP-4 high order approximation for the specific zone (default is determined by the solve option). The zone order must be lower than the solve order (see Table 10.176 for setting of the solve order). The reader is referred to Section 4.2.8.1 for a description of high order approximations.	
Parameters	mp4.cons	Use MP 4^{th} order approximation for variables reconstruction; apply using the conservative variables.
	mp4.prim	Use MP 4^{th} order approximation for variables reconstruction; apply using the primitive variables (recommended).
	mp4.char	Use MP 4^{th} order approximation for variables reconstruction; apply using the characteristic variables.
Examples	-zone order mp4.cons	

Table 10.89: Geometry zone order option (MP-4)

Order (MP-5)

Syntax	-zone order {mp5.cons | mp5.prim | mp5.char}	
Description	Use MP-5 high order approximation for the specific zone (default is determined by the solve option). The zone order must be lower than the solve order (see Table 10.178 for setting of the solve order). The reader is referred to Section 4.2.8.1 for a description of high order approximations.	
Parameters	mp5.cons	Use MP 5^{th} order approximation for variables reconstruction; apply using the conservative variables.
	mp5.prim	Use MP 5^{th} order approximation for variables reconstruction; apply using the primitive variables (recommended).
	mp5.char	Use MP 5^{th} order approximation for variables reconstruction; apply using the characteristic variables.
Examples	-zone order mp5.cons	

Table 10.90: Geometry zone order option (MP-5)

Order (WENO-5)

Syntax	-zone order {weno5.cons | weno5.prim | weno5.char}
Description	Use WENO-5 high order approximation for the specific zone (default is determined by the solve option). The zone order must be lower than the solve order (see Table 10.180 for setting of the solve order). The reader is referred to Section 4.2.8.2 for a description of high order approximations.
Parameters	weno5.cons — Use WENO 5^{th} order approximation for variables reconstruction; apply using the conservative variables.
	weno5.prim — Use WENO 5^{th} order approximation for variables reconstruction; apply using the primitive variables (recommended).
	weno5.char — Use WENO 5^{th} order approximation for variables reconstruction; apply using the characteristic variables.
Examples	-zone order weno5.cons

Table 10.91: Geometry zone order option (WENO-5)

Tu		
Syntax	-zone tu {inviscid | no.source | baldwin.lomax}	
Description	Sets different modes for zonal-turbulence: No turbulence source, Baldwin-Lomax, or fully inviscid.	
Parameters	inviscid	Current zone is treated as fully inviscid.
	baldwin.lomax	Current zone uses the Baldwin-Lomax turbulence model.
	no.source	Turn off turbulence model source term for current zone.
Examples	-zone tu no.source	

Table 10.92: Geometry zone tu option

Zonal TNT X-LES

Syntax	-zone tnt.xles {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla}	
Description	Set X-LES mode for this zone (overrides setting from Table 10.156).	
Parameters	sg.delta.volume	Use volume based subgrid scale (see Section 3.5.1.1).
	sg.delta.max	Use edge length based subgrid scale (see Section 3.5.1.2).
	sg.delta.lsq	Use the LSQ subgrid scale (see Section 3.5.1.3).
	sg.delta.omega	Use vorticity aligned subgrid scale (see Section 3.5.1.4).
	sg.delta.sla	Use shear layer adapted subgrid scale (see Section 3.5.1.5).
Examples	-zone tnt.xles sg.delta.volume	
	-zone tnt.xles sg.delta.sla	

Table 10.93: Geometry zone tnt.xles option

Zonal SST DDES		
Syntax	-zone sst.ddes {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla}	
Description	Set DDES mode for this zone (overrides setting from Table 10.158).	
Parameters	sg.delta.volume	Use volume based subgrid scale (see Section 3.5.1.1).
	sg.delta.max	Use edge length based subgrid scale (see Section 3.5.1.2).
	sg.delta.lsq	Use the LSQ subgrid scale (see Section 3.5.1.3).
	sg.delta.omega	Use vorticity aligned subgrid scale (see Section 3.5.1.4).
	sg.delta.sla	Use shear layer adapted subgrid scale (see Section 3.5.1.5).
Examples	-zone sst.ddes sg.delta.volume	
	-zone sst.ddes sg.delta.sla	

Table 10.94: Geometry zone sst.ddes option

Zonal SST IDDES		
Syntax	-zone sst.iddes {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla}	
Description	Set IDDES mode for this zone (overrides setting from Table 10.159).	
Parameters	sg.delta.volume	Use volume based subgrid scale (see Section 3.5.1.1).
	sg.delta.max	Use edge length based subgrid scale (see Section 3.5.1.2).
	sg.delta.lsq	Use the LSQ subgrid scale (see Section 3.5.1.3).
	sg.delta.omega	Use vorticity aligned subgrid scale (see Section 3.5.1.4).
	sg.delta.sla	Use shear layer adapted subgrid scale (see Section 3.5.1.5).
Examples	-zone sst.iddes sg.delta.volume -zone sst.iddes sg.delta.sla	

Table 10.95: Geometry zone sst.iddes option

Zonal SST PANS

Syntax	-zone sst.pans {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla}
Description	Set PANS mode for this zone (overrides setting from Table 10.160).
Parameters	sg.delta.volume — Use volume based subgrid scale (see Section 3.5.1.1). sg.delta.max — Use edge length based subgrid scale (see Section 3.5.1.2). sg.delta.lsq — Use the LSQ subgrid scale (see Section 3.5.1.3). sg.delta.omega — Use vorticity aligned subgrid scale (see Section 3.5.1.4). sg.delta.sla — Use shear layer adapted subgrid scale (see Section 3.5.1.5).
Examples	-zone sst.pans sg.delta.volume -zone sst.pans sg.delta.sla

Table 10.96: Geometry zone sst.pans option

<table>
<tr><th colspan="3">Zonal Hybrid.central.upwind</th></tr>
<tr><td>Syntax</td><td colspan="2">-zone hybrid.central.upwind {des.[momentum.]{2nd|4th} | ducros.[momentum.]{2nd|4th} | larsson.[momentum.]{2nd|4th} travin.[momentum.]{2nd|4th} | fsm.[momentum.]{2nd|4th} | none} #sigma_0 [#sigma_0_me]</td></tr>
<tr><td>Description</td><td colspan="2">Hybrid central differencing with upwind approximations. Use together with k-ω hybrid models only!</td></tr>
<tr><td>Parameters</td><td>des</td><td>Use the DES based blending function (see Section 4.3.1.1). Use together with DDES or IDDES hybrid models only!</td></tr>
<tr><td></td><td>ducros</td><td>Use the ducros blending function (see Section 4.3.1.2).</td></tr>
<tr><td></td><td>larsson</td><td>Use the Larsson blending function (see Section 4.3.1.3).</td></tr>
<tr><td></td><td>travin</td><td>Use the Travin blending function (see Section 4.3.1.4).</td></tr>
<tr><td></td><td>fsm</td><td>Use the FSM blending function (see Section 4.3.1.5).</td></tr>
<tr><td></td><td>2nd</td><td>2^{nd} order central differencing.</td></tr>
<tr><td></td><td>4th</td><td>4^{th} order central differencing.</td></tr>
<tr><td></td><td>momentum</td><td>Apply hybrid central upwind to the momentum equation only.</td></tr>
<tr><td></td><td>none</td><td>No central upwind blending for this zone. Note, when using the “none” option, there is no need to enter a value for #sigma_0[_me].</td></tr>
<tr><td></td><td>#sigma_0</td><td>Lower limit of the blending function.</td></tr>
<tr><td></td><td>#sigma_0_me</td><td>When added, lower limit of the continuity and energy equations blending function.</td></tr>
<tr><td>Examples</td><td colspan="2">-zone hybrid.central.upwind des.momentum.4th 0.1</td></tr>
</table>

Table 10.97: Geometry zone hybrid.central.upwind option

CFL.factor		
Syntax	-zone cfl.factor #factor	
Description	Set the zone CFL number to be a factor times the global CFL.	
Parameters	#factor	CFL factor.
Examples	-zone cfl.factor 0.5	

Table 10.98: Geometry zone cfl.factor option

Board		
Syntax	-zone board #board	
Description	Explicitly set the board on which the zone is solved.	
Parameters	#board	Board number.
Examples	-zone board 2	

Table 10.99: Geometry zone board option

Multigrid (Zonal)

Syntax	-zone multigrid #levels [[all.levels | level #] [do.not.coarse.in.i | do.not.coarse.in.j | do.not.coarse.in.k]...] [prolong.lin | prolong.log] [damp]
Description	Set the number of multigrid levels (should be equal to or less than the declared multigrid levels) and enable directional multigrid per level (optional). Overrides global multigrid settings for this zone (see Table 10.228 for global settings). You may disable coarsening in a specific grid direction via do.not.coarse.in.i|j|k , especially useful when the number of grid cells in that direction is odd.
Parameters	#levels — Multigrid levels. all.levels — Directional directives apply to all levels. level — Specific level. # — Level number. do.not.coarse.in.i — Directional coarsening directive. do.not.coarse.in.j — Directional coarsening directive. do.not.coarse.in.k — Directional coarsening directive. prolong.lin — (Tri)-linear prolongation (interpolation) of coarse-grid corrections (default). prolong.log — Positivity-preserving, log-based prolongation (interpolation) of coarse-grid corrections. damp — Damping of coarse-grid defect and time-step near discontinuities (shock/interface).
Examples	-zone multigrid 3 -zone multigrid 4 level 2 do.not.coarse.in.k

Table 10.100: Geometry zone multigrid option

10.5.4 Zonal Chimera Options

The Chimera options inherit the default of the "-sim chimera"directives (see Section 10.5.4). They can be subsequently changed for each zone using the directives in this section. As mentioned in Section 6.3, the "Chimera" process is composed of two main parts, namely, hole cutting, and interpolation stencil search. The hole cutting can be conducted in one of two ways. The first, manually, using bodies, envelopes, and phantom bodies, and second, automatically. The following options pertain to both ways.

Do.not.cut.holes.in		
Syntax	-chimera do.not.cut.holes.in [all | $entity...]	
Description	Excludes the current entity from the Chimera cut list of the listed entities. The list may contain zones and bodies. This is the first step in the Chimera-whitelist/blacklist. Must be in a body/phantom scope and out of a zone scope (see Section 6.3.2).	
Parameters	all	All entities.
	$entity	Entity can be a body, a zone.
Examples	-chimera do.not.cut.holes.in all.zones	

Table 10.101: Geometry Chimera do.not.cut.holes.in option

Cut.holes.in		
Syntax	-chimera cut.holes.in [all | $entity...]	
Description	Includes the current entity in the chimera cut list of the listed entities. The list can contain zones and bodies This is the second step in the Chimera-whitelist/blacklist. Must be in a body/phantom scope and out of a zone scope (see Section 6.3.2).	
Parameters	all	All entities.
	$entity	Entity can be a body, a zone.
Examples	-chimera cut.holes.in all	

Table 10.102: Geometry Chimera cut.holes.in option

Do.not.cut.me.with		
Syntax	-zone chimera do.not.cut.me.with [all | all.bodies | all.phantoms | $entity...]	
Description	Excludes the entity list from the current entity's Chimera cut list The list can contain zones, bodies and phantoms. This is the third step in Chimera-whitelist/blacklist. Must be in body or zone scope (see Section 6.3.2).	
Parameters	all	All entities.
	all.bodies	All bodies.
	all.phantoms	All phantoms.
	$entity	Entity can be a body, a zone, or a phantom.
Examples	-zone chimera do.not.cut.me.with all.phantoms	

Table 10.103: Geometry Chimera do.not.cut.me.with option

Cut.me.with

Syntax	-zone chimera cut.me.with [all | all.bodies | all.phantoms | $entity...]	
Description	Includes the current entity in the Chimera cut list of the listed entities. The list can contain zones and bodies. This is the fourth step in Chimera-whitelist/blacklist. Must be in body or zone scope (see Section 6.3.2).	
Parameters	all	All entities.
	all.bodies	All bodies.
	all.phantoms	All phantoms.
	$entity	Entity can be a body, a zone, or a phantom.
Examples	-zone chimera cut.me.with all.bodies	

Table 10.104: Geometry Chimera cut.me.with option

Zonal Chimera Options

Syntax	-zone chimera [minimum # | layers # | allow.1st | force.zombie.freestream | leave.zombie.unhandled | allow.inner.interpolation | allow.freestream.interpolation | allow.unhandled.interpolation | disable.interpolation]
Description	Per zone Chimera options.
Parameters	minimum # — Minimum cell cloud for interpolation layers # — The number of additional chimera fronts. If 1 is entered then it assumes also allow.1st. allow.1st — Allows for a single cell chimera front, where face values are evaluated in 1st order. force.zombie... — Force zombies to initialize with free-stream values. leave.zombie... — Leave the zombies un-initialized (with free-stream initial initialization). allow.inner... — Allow interpolation from inner neigbhouring cells. allow.free... — Allow interpolation from free stream. allow.unhandled... — Allow unhandled interpolations. disable.interpolation — Disables all interpolations and keep Chimera front cells unchanged.
Examples	-zone chimera allow.inner.interpolation -zone chimera allow.freestream.interpolation -zone chimera allow.1st -zone chimera layers 1

Table 10.105: Geometry zone Chimera option

Chimera Interpolation Options

Syntax	-zone chimera [shepard1 | shepard2 | microsphere1 | microsphere2]	
Description	Set the interpolation scheme.	
Parameters	shepard1	Shepard weighted interpolation scheme based on the inverse of the distance (default).
	shepard2	Shepard weighted interpolation scheme based on the inverse of the distance squared.
	microsphere1	Micro-sphere weighted interpolation scheme based on the inverse of the distance (designed to minimize the bias of the Shepard scheme).
	microsphere2	Micro-sphere weighted interpolation scheme based on the inverse of the distance squared.
Examples	-zone chimera shepard2	

Table 10.106: Geometry Chimera interpolation options

Chimera Complementary Interpolation Options		
Syntax	-zone chimera [tri.linear | tri.linear.or.logical]	
Description	Set the complementary interpolation scheme.	
Parameters	tri.linear	Tri-linear interpolation scheme (attempts to use trilinear interpolation where available).
	tri.linear.or.logical	Tri-linear interpolation scheme or logical (use either trilinear interpolation or logical interpolation but not both; trilinear has precedence).
Examples	-zone chimera tri.linear	

Table 10.107: Geometry Chimera complementary interpolation options

10.5.4.1 Manual Hole Cutting

Envelope		
Syntax	-zone chimera envelope {[{i | j | k} {$>$ | $<$} #]...} [all | $entity...]	
Description	Defines an envelope to be treated as a body for the listed entities. The list can contain zones and bodies. Can be used in zone scope only (see Section 6.4).	
Parameters	i	i (ξ) computational coordinate direction.
	j	j (η) computational coordinate direction.
	k	k (ζ) computational coordinate direction.
	$>$	Grid point index more than.
	$<$	Grid point index less than.
	#	Grid point index.
	all	All entities.
	$entity	Entity can be a body, a zone, or a phantom.
Examples	-zone chimera envelope i $>$ 69 i $<$ 221 k $<$ 25 fuselage	

Table 10.108: Geometry Chimera envelope option

Surface.cut		
Syntax	-zone chimera surface.cut {yes | no | far | near | tight}	
Description	In zone scope, select the default surface cutting.	
Parameters	yes	Enable surface cutting to this zone (same as far).
	no	Disable surface hole cutting. See Figures 6.3 and 6.4 for a description of the resulting holes.
	far	See Figures 6.3 and 6.5 for a description of the resulting holes.
	near	See Figures 6.3 and 6.6 for a description of the resulting holes.
	tight	See Figures 6.3 and 6.7 for a description of the resulting holes.
Examples	-zone chimera surface.cut no	

Table 10.109: Geometry Chimera surface.cut option

Surface.cut override		
Syntax	-zone chimera {do.not.surface.cut.with | surface.cut.far | surface.cut.near | surface.cut.tight} $entity...	
Description	In zone scope, override the surface cutting of zone per entity.	
Parameters	do.not.surface.cut.with	Disable surface cutting to this zone by this entity. See Figures 6.3 and 6.4 for a description of the resulting holes.
	far	See Figures 6.3 and 6.5 for a description of the resulting holes.
	near	See Figures 6.3 and 6.6 for a description of the resulting holes.
	tight	See Figures 6.3 and 6.7 for a description of the resulting holes.
	$entity	Entity.
Examples	-zone chimera do.not.surface.cut.with wing	

Table 10.110: Geometry Chimera surface.cut override option

Kill/revive		
Syntax	-zone chimera {kill | revive} i = #min, #max j = #min, #max k = #min, #max	
Description	In zone scope, kill or revive specific nodes.	
Parameters	kill	Kill specific nodes
	revive	Revive specific nodes
	i	i (ξ) computational coordinate direction.
	j	j (η) computational coordinate direction.
	k	k (ζ) computational coordinate direction.
	#min	Grid point minimum index
	#max	Grid point minimum index
Examples	-zone chimera kill i = 110,148 j = 5,11 k = 56,66	

Table 10.111: Geometry Chimera kill/revive option

Interpolation.priority		
Syntax	-zone chimera interpolation.priority $entity...	
Description	In zone scope, changes the priority of interpolation seeking.	
Parameters	$entity...	List of entity IDs.
Examples	-zone chimera interpolation.priority wing pylon	

Table 10.112: Geometry Chimera interpolation.priority option

Do.not.interpolate.from		
Syntax	-zone chimera do.not.interpolate.from $entity...	
Description	In zone scope, exclude the entity from interpolation seeking.	
Parameters	$entity...	List of entity IDs.
Examples	-zone chimera do.not.interpolate.from wing pylon	

Table 10.113: Geometry Chimera do.not.interpolate.from option

Chimera Auto/Manual		
Syntax	-zone chimera {auto | manual}	
Description	In zone scope, selects automatic or manual Chimera algorithms.	
Parameters	auto	Automatic hole cutting to this zone.
	manual	Manual hole cutting to this zone.
Examples	-zone chimera auto	

Table 10.114: Geometry Chimera auto/manual option

Zonal Partial Automatic Chimera		
Syntax	-zone chimera {auto.overlap.minimization | auto.force.overlapping}	
Description	In zone scope, selects partial automatic algorithms: automatic minimization or automatic force overlap.	
Parameters	auto.overlap.mini	Enable automatic overlap minimization.
	auto.force.overlap	Enable automatic exact force overlap.
Examples	-zone chimera auto.force.overlapping	

Table 10.115: Geometry zonal automatic Chimera option

Suitability		
Syntax	-zone chimera {auto.suitability.distance | auto.suitability.volume | auto.suitability.max.diag | auto.suitability.min.diag} #weight	
Description	In zone scope controls the suitability function cross cells for overlap minimization (default is the simulation's choice or, if not set, auto.suitability.max.diag 1).	
Parameters	auto.suitability.di	Automatic hole cutting based on wall distance.
	auto.suitability.vo	Automatic hole cutting based on cell volume.
	auto.suitability.ma	Automatic hole cutting based on maximal cell diagonal.
	auto.suitability.mi	Automatic hole cutting based on minimal cell diagonal.
	#weight	Weight of hole cutting criterion.
Examples	-zone chimera auto.suitability.distance 0.3	

Table 10.116: Geometry Chimera suitability option

Enclave		
Syntax	-zone chimera {auto.enclave.match | auto.enclave.differ}	
Description	In zone scope, selects the elimination method of enclaved grid points that are created by the minimization algorithm.	
Parameters	auto.enclave.match	Remove enclaved grid points after the inner-formation hole cutting ("matching formation"). Therefore, it removes enclaved grid parts that are the result of inner-formation zones. Invoked only once.
	auto.enclave.differ	Remove enclaved grid points after outer-formation hole-cutting ("formation differ"). Therefore it removes all enclaved parts (this option is more expensive in terms of computation time). Invoked every time the Chimera module is invoked.
Examples	-zone chimera auto.enclave.differ	

Table 10.117: Geometry Chimera enclave option

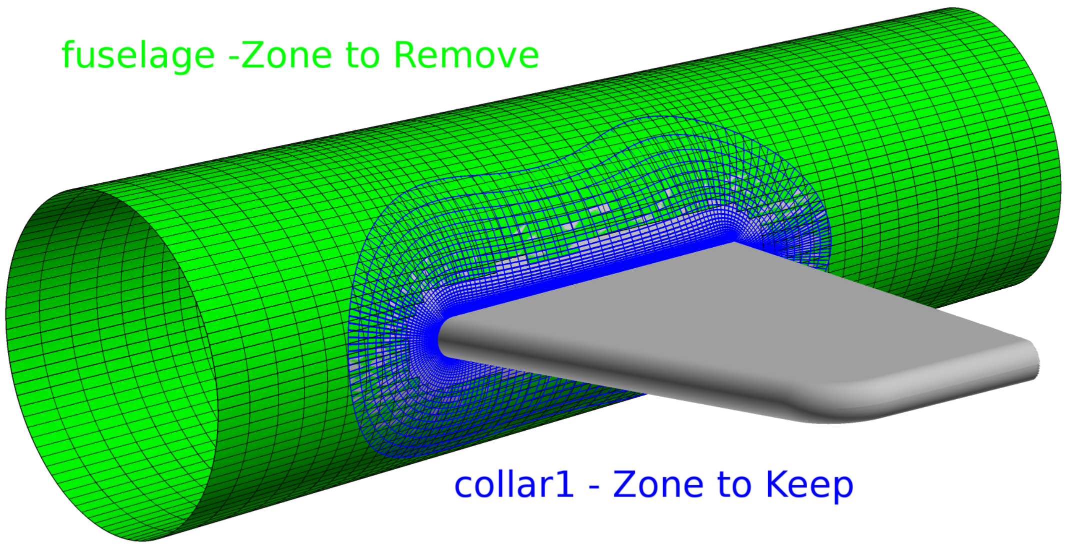


Figure 10.2: Example of force overlap
see Table 10.118

10.5.4.2 Force Overlap

The force overlap/seek/project command is intended to address the issue of overlapping wall surfaces. By default, EZAir summates the forces on all the live wall cells. In case of two (or more) wall surfaces which have an area of overlap, the force in this area may be summated twice. To resolve this issue and to prevent the duplicate summation, one of the overlapping surfaces should be subtracted from the other. In Figure 10.2 the blue wall surface, belonging to the zone collar1 (*jmin*), overlaps the green wall surface, belonging to the zone fuselage (*kmin*). In this case, the collar surface can be subtracted from the fuselage surface. This means that the force on all the collar surface live cells will be summated and the fuselage surface cells which overlap the collar surface will either be omitted from the summation or partially summated according to the prescribed scheme (overlap/seek/project). For example, the input file line would look as follows: “-chimera force.seek collar1 jmin fuselage kmin”

Force.overlap

Syntax	-chimera force.overlap $zone-to-keep {imin | imax | jmin | jmax | kmin | kmax [i, j, k = #min, #max]...} $zone-to-remove surface [logi = #]
Description	Seek/Overlap algorithm for force overlap removal. Results in in-/out flag for force surfaces (as well as logical interpolation). See Figure 10.2 for an example.
Parameters	$zone-to-keep — Surface that the overlapping area is subtracted from.
	i, j, k — Keywords for ξ, η and ζ coordinates.
	imin, imax — The surface edge.
	jmin, jmax — The surface edge.
	kmin, kmax — The surface edge.
	#min, #max — Index values.
	$zone-to-remove — Surface that has precedence.
	logi = # — Number of grid points to use for logical interpolation
Examples	-chimera force.overlap c_fus-frng_a-3 jmin fuselage kmin

Table 10.118: Geometry Chimera force.overlap option

Force.project	
Syntax	-chimera force.project $zone-to-keep {imin | imax | jmin | jmax | kmin | kmax [i, j, k = #min, #max]...} $zone-to-subtract surface [lmax = #l] [logi #n]
Description	Projection algorithm for force overlap removal. Results in partial surface calculation (as well as logical interpolation).
Parameters	$zone-to-subtract: Surface that the overlapping area is subtracted from.
	i, j, k: Keywords for ξ, η and ζ coordinates.
	imin, imax: The surface edge.
	jmin, jmax: The surface edge.
	kmin, kmax: The surface edge.
	#min, #max: Index values.
	$zone-to-stay: Surface that has precedence.
	logi = #n: Number of grid points to use for logical interpolation.
	lmax= #l: Length tolerance for manual force-overlapping (the maximal length allowed).
Examples	-chimera force.project c_fus-frng_a-3 jmin fuselage kmin logi 30

Table 10.119: Geometry Chimera force.project option

Forcing		
Syntax	-chimera forcing $zone-to-keep {imin | imax | jmin | jmax | kmin | kmax [i, j, k = #min, #max]...} $zone-to-remove surface logi = #	
Description	Apply logical interpolation.	
Parameters	$zone-to-keep	Surface that the overlapping area is subtracted from.
	i, j, k	Keywords for ξ, η and ζ coordinates.
	imin, imax	The surface edge.
	jmin, jmax	The surface edge.
	kmin, kmax	The surface edge.
	#min, #max	Index values.
	$zone-to-remove	Surface that has precedence.
	logi = #	Number of grid points to use for logical interpolation
Examples	-chimera forcing c_fus-frng_a-3 jmin fuselage kmin logi 11	

Table 10.120: Geometry Chimera forcing option

10.5.4.3 Phantom Options

Phantom is a body that is used only for the purpose of manual hole cutting. The reader is referred to Section 6.5 for the description of Phantoms.

Entity.box		
Syntax	-phantom $phantom entity.box $body...	
Description	Defines a bounding box based on body entities.	
Parameters	$phantom	Phantom name.
	$body...	Body name.
Examples	-phantom my_phantom entity.box my_body	

Table 10.121: Geometry phantom entity.box option

<table>
<tr><th colspan="3">Region</th></tr>
<tr><td>Syntax</td><td colspan="2">-phantom $phantom region [{{min | max} $vertex} | {x | y | z} {> | <} $vzv}...] [inside | outside]</td></tr>
<tr><td>Description</td><td colspan="2">Defines a Cartesian body region.</td></tr>
<tr><td>Parameters</td><td>$phantom</td><td>Phantom name.</td></tr>
<tr><td></td><td>$vzv</td><td>{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.</td></tr>
<tr><td></td><td>min</td><td>Region lower bound.</td></tr>
<tr><td></td><td>max</td><td>Region upper bound.</td></tr>
<tr><td></td><td>$vertex</td><td>Vertex name.</td></tr>
<tr><td></td><td>x</td><td>x axis.</td></tr>
<tr><td></td><td>y</td><td>y axis.</td></tr>
<tr><td></td><td>z</td><td>z axis.</td></tr>
<tr><td></td><td>></td><td>Greater than.</td></tr>
<tr><td></td><td><</td><td>Lower than.</td></tr>
<tr><td></td><td>inside</td><td>Cut holes inside the region.</td></tr>
<tr><td></td><td>outside</td><td>Cut holes outside the region.</td></tr>
<tr><td>Examples</td><td colspan="2">-phantom symm-vb region y > -0.3</td></tr>
</table>

Table 10.122: Geometry phantom region option

Ellipse		
Syntax	-phantom $phantom ellipse a1 $vzv a2 $vzv b $vzv [span $vzv]} [inside | outside]	
Description	Defines an elliptic cylindrical body region.	
Parameters	$phantom	Phantom name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	a1	First apex on main axis.
	a2	Second apex on main axis.
	b	Apex on secondary axis.
	span	Span point for the cylinder, if not given the cylinder spans to infinity.
	inside	Cut holes inside the ellipse.
	outside	Cut holes outside the ellipse.
Examples	-phantom nozzle-ph ellipse a1 0 0 0 a2 1 0 0 b 0.25 span 1	

Table 10.123: Geometry phantom ellipse option

Ellipsoid		
Syntax	-phantom $phantom ellipsoid a1 $vzv a2 $vzv b $vzv [inside | outside]	
Description	Defines an ellipsoid body region.	
Parameters	$phantom	Phantom name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	a1	First apex on main axis.
	a2	Second apex on main axis.
	b	Apex on secondary axis.
	inside	Cut holes inside the ellipsoid.
	outside	Cut holes outside the ellipsoid.
Examples	-phantom nozzle-ph ellipsoid a1 0 0 0 a2 1 0 0 b 0.25 inside	

Table 10.124: Geometry phantom ellipsoid option

Cylinder		
Syntax	-phantom $phantom cylinder c1 $vzv c2 $vzv {rmax $vzv | rmax1 $vzv rmax2 $vzv} {rmin $vzv | rmin1 $vzv rmin2 $vzv} [inside | outside]	
Description	Defines a cylindrical body (optionally hollow, or conical) region.	
Parameters	$phantom	Phantom name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	c1, c2	Center points of the circles.
	rmin1, rmin2	inner radii of the circles.
	rmin	Sets both rmin1 and rmin2.
	rmax1, rmax2	outer radii of the circles.
	rmax	Sets both rmax1 and rmax2.
	inside	Cut holes inside the cylinder.
	outside	Cut holes outside the cylinder.
Examples	-phantom ring-vb cylinder c1 ring1-vx c2 ring2-vx rmin 0.072 rmax 0.09	

Table 10.125: Geometry phantom cylinder option

Fan.cylinder		
Syntax	-phantom $phantom fan.cylinder [{np $vzv}...] span $vzv [inside | outside]	
Description	Defines a fan cylinder body region. The cross-section is a Triangle-Fan geometry element where the first vertex is the apex. EPPIC converts arbitrary polygons to a sequence of fan cylinders.	
Parameters	$phantom	Phantom name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	span	Span point for the cylinder, if not given the cylinder spans to infinity.
	inside	Cut holes inside the fan.
	outside	Cut holes outside the fan.
Examples	-phantom pylon3_l-vb fan.cylinder np pylon3_l1-vx np pylon3_l2-vx np pylon3_l3-vx np pylon3_l4-vx np pylon3_l5-vx span -0.0731	

Table 10.126: Geometry phantom fan.cylinder option

Sphere		
Syntax	-phantom $phantom sphere o $vzv radius #r [inside | outside]	
Description	Define a spherical body region. The center is marked by o and the radius by r.	
Parameters	$phantom	Phantom name.
	o	Sphere origin.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	#r	Radius.
	inside	Cut holes inside the fan.
	outside	Cut holes outside the fan.
Examples	-phantom my_sphere o center1 r = 1	

Table 10.127: Geometry phantom sphere option

Project	
Syntax	-phantom $phantom {project.extent | project.entity} $bodyname [inside | outside]
Description	Defines a phantom from the analytical body representation.
Parameters	project.extent — Around the body grid extents. project.entity — Around the body's entities. $bodyname — Body name. inside — Cut holes inside the phantom. outside — Cut holes outside the phantom.
Examples	-phantom fus-ph project.entity fuselage-body inside

Table 10.128: Geometry phantom project option

Revive	
Syntax	-revive
Description	Converts a phantom into a revive phantom (performs an action opposite of a hole cutter), *i.e.*, cells within the phantom are not cut (must be within a phantom scope).
Parameters	N/A
Examples	-revive

Table 10.129: Geometry phantom revive option

10.5.5 Shape Options

The shape options are similar to the phantom options with the exception that shapes do not generate holes. The shape options may be used to initialize a certain flow variable within a certain region or to restrict boundary conditions. It requires the definition of the shape, described herein, followed by the initialization, "-init", directive. It can also be utilized to confine the turbulence model source term to a certain shape. The description of the initialization appears in the following table. The remainder of the section contains a description of the shape options.

Init		
Syntax	-init [p | T | mt | u | v | w | ti] =#...	
Description	Initialize the flow field defined by the shape with various primitives (pertains to a certain shape scope).	
Parameters	p	Pressure.
	T	Temperature.
	mt	Turbulent viscosity.
	u, v, w	Velocity components in Cartesian coordinates.
	ti	Turbulent intensity.
	#	Value of primitive.
Examples	-init T 300.0	

Table 10.130: Geometry shape init option

Entity.box		
Syntax	-shape $shape entity.box $body... [restrict.trb]	
Description	Defines a bounding box based on body entities	
Parameters	$shape	Shape name.
	$body...	Body name.
	restrict.trb	Restricts the operation of the turbulent source term solely to within the shape. Can be used to simulate forced transition.
Examples	-shape test-entity entity.box eft-body	

Table 10.131: Geometry shape entity.box option

<table>
<tr><th colspan="3">Region</th></tr>
<tr><td>Syntax</td><td colspan="2">-shape $shape region [{{min | max} $vertex} | {x | y | z} {> | <} $vzv}...] [inside | outside] [restrict.trb]</td></tr>
<tr><td>Description</td><td colspan="2">Defines a Cartesian body region.</td></tr>
<tr><td>Parameters</td><td>$shape</td><td>Shape name.</td></tr>
<tr><td></td><td>$vzv</td><td>{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.</td></tr>
<tr><td></td><td>min</td><td>Region lower bound.</td></tr>
<tr><td></td><td>max</td><td>Region upper bound.</td></tr>
<tr><td></td><td>$vertex</td><td>Vertex name.</td></tr>
<tr><td></td><td>x</td><td>x axis.</td></tr>
<tr><td></td><td>y</td><td>y axis.</td></tr>
<tr><td></td><td>z</td><td>z axis.</td></tr>
<tr><td></td><td>></td><td>Greater than.</td></tr>
<tr><td></td><td><</td><td>Lower than.</td></tr>
<tr><td></td><td>inside</td><td>Refers to the inside segment of the region.</td></tr>
<tr><td></td><td>outside</td><td>Refers to the outside segment of the region.</td></tr>
<tr><td></td><td>restrict.trb</td><td>Restricts the operation of the turbulent source term solely to within the shape. Can be used to simulate forced transition.</td></tr>
<tr><td>Examples</td><td colspan="2">-shape symm-vb region y > -0.3

-shape trb-shape region x > 4.3 restrict.trb</td></tr>
</table>

Table 10.132: Geometry shape region option

Ellipse

Syntax	-shape $shape ellipse a1 $vzv a2 $vzv b $vzv [span $vzv]} [inside | outside] [restrict.trb]	
Description	Defines an elliptic cylindrical body region.	
Parameters	$shape	Shape name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	a1	First apex on main axis.
	a2	Second apex on main axis.
	b	Apex on secondary axis.
	span	Span point for the cylinder, if not given the cylinder spans to infinity.
	inside	Refers to the inside segment of the ellipse.
	outside	Refers to the outside segment of the ellipse.
	restrict.trb	Restricts the operation of the turbulent source term solely to within the shape. Can be used to simulate forced transition.
Examples	-shape nozzle-ph ellipse a1 0 0 0 a2 1 0 0 b 0.25 span 1 inside	

Table 10.133: Geometry shape ellipse option

Ellipsoid		
Syntax	-shape $shape ellipsoid a1 $vzv a2 $vzv b $vzv [inside | outside] [restrict.trb]	
Description	Defines an ellipsoid body region.	
Parameters	$shape	Shape name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	a1	First apex on main axis.
	a2	Second apex on main axis.
	b	Apex on secondary axis.
	inside	Refers to the inside segment of the ellipsoid.
	outside	Refers to the outside segment of the ellipsoid.
	restrict.trb	Restricts the operation of the turbulent source term solely to within the shape. Can be used to simulate forced transition.
Examples	-shape nozzle-ph ellipsoid a1 0 0 0 a2 1 0 0 b 0.25 inside	

Table 10.134: Geometry shape ellipsoid option

Cylinder		
Syntax	-shape $shape cylinder c1 $vzv c2 $vzv {rmax $vzv | rmax1 $vzv rmax2 $vzv} {rmin $vzv | rmin1 $vzv rmin2 $vzv} [inside | outside] [restrict.trb]	
Description	Defines a cylindrical body (optionally hollow, or conical) region.	
Parameters	$shape	Shape name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	c1, c2	Center points of the circles.
	rmin1, rmin2	inner radii of the circles.
	rmin	Sets both rmin1 and rmin2.
	rmax1, rmax2	outer radii of the circles.
	rmax	Sets both rmax1 and rmax2.
	inside	Refers to the inside segment of the cylinder.
	outside	Refers to the outside segment of the cylinder.
	restrict.trb	Restricts the operation of the turbulent source term solely to within the shape. Can be used to simulate forced transition.
Examples	-shape ring-vb cylinder c1 ring1-vx c2 ring2-vx rmin 0.072 rmax 0.09	

Table 10.135: Geometry shape cylinder option

Fan.cylinder		
Syntax	-shape $shape fan.cylinder [{np $vzv}...] span $vzv [inside | outside] [restrict.trb]	
Description	Defines a fan cylinder body region. The cross-section is a Triangle-Fan geometry element where the first vertex is the apex. EPPIC converts arbitrary polygons to a sequence of fan cylinders.	
Parameters	$shape	Shape name.
	$vzv	{$vertex | $zone $i = j = k = \#$ | #val} (either vertex name, grid point, or value when applicable). See Section 10.5.7 for the definition of a vertex.
	span	Span point for the cylinder, if not given the cylinder spans to infinity.
	inside	Refers to the inside segment of the fan.cylinder.
	outside	Refers to the outside segment of the fan.cylinder.
	restrict.trb	Restricts the operation of the turbulent source term solely to within the shape. Can be used to simulate forced transition.
Examples	-shape pylon3__l-vb fan.cylinder np pylon3__l1-vx np pylon3__l2-vx np pylon3__l3-vx np pylon3__l4-vx np pylon3__l5-vx span -0.0731	

Table 10.136: Geometry shape fan.cylinder option

Project		
Syntax	-shape $shape {project.extent | project.entity} $bodyname [inside | outside] [restrict.trb]	
Description	Defines a shape from the analytical body representation.	
Parameters	project.extent	Around the body grid extents.
	project.entity	Around the body's entities.
	$bodyname	Body name.
	inside	Refers to the inside segment of the shape.
	outside	Refers to the outside segment of the shape.
	restrict.trb	Restricts the operation of the turbulent source term solely to within the shape. Can be used to simulate forced transition.
Examples	-shape fus-sh project.entity fuselage-body inside	

Table 10.137: Geometry shape project option

10.5.6 Disturb Options

The directive “-disturb” allows to override flow variables in given zones or free stream values with user prescribed values.

Import		
Syntax	-disturb import $file {R | U | V | W | P | RU | RV | RW | ET | T | RPG | RPV | PG | PV | RG | RV | K | O | RK | | tR | tRU | tRV | tRW | tET | tRPG | tRPV | tRK | tRO}...	
Description	Imports a Plot3D solution file, following the filenames are the variables in the file in their exact order. The values in the file are used to override the flow variables.	
Parameters	$file	Plot3D file name.
Examples	-disturb import params-p.f p	

Table 10.138: Geometry disturb import option

Freestream	
Syntax	[-disturb | -disturb.all] [u.freestream | v.freestream | w.freestream | p.freestream | T.freestream]
Description	Set the specified primitive to the free-stream value. The directive "-disturb" refers to the current zone only while the directive "-disturb.all" refers to all zones.
Parameters	u.freestream — Set x coordinate direction velocity. v.freestream — Set y coordinate direction velocity. w.freestream — Set z coordinate direction velocity. p.freestream — Set the pressure. T.freestream — Set the temperature.
Examples	-disturb u.freestream -disturb.all p.freestream

Table 10.139: Geometry disturb freestream option

Uniform		
Syntax	[-disturb | -disturb.all] [u.uniform | v.uniform | w.uniform | p.uniform | T.uniform | mt.uniform] #	
Description	Set specified primitive to a specified value. The directive "-disturb" refers to the current zone only while the directive "-disturb.all" refers to all zones.	
Parameters	u.uniform	Set x coordinate direction velocity.
	v.uniform	Set y coordinate direction velocity.
	w.uniform	Set z coordinate direction velocity.
	p.uniform	Set the pressure.
	T.uniform	Set the temperature.
	mt.uniform	Set the turbulent viscosity.
	#	Value.
Examples	-disturb u.uniform 50	
	-disturb p.uniform 100000	

Table 10.140: Geometry disturb uniform option

Shift		
Syntax	[-disturb | -disturb.all] [u.shift | v.shift | w.shift | p.shift] #delta	
Description	Galilean transformation to the velocity and a simple shift to the pressure. The directive "-disturb" refers to the current zone only while the directive "-disturb.all" refers to all zones.	
Parameters	u.shift	Shift the x coordinate direction velocity.
	v.shift	Shift the y coordinate direction velocity.
	w.shift	Shift the z coordinate direction velocity.
	p.shift	Shift the pressure.
	#delta	Shift value.
Examples	-disturb w.shift 10	
	-disturb.all u.shift -5	

Table 10.141: Geometry disturb shift option

Rotate		
Syntax	[-disturb | -disturb.all] [rotate.x.rad | rotate.x.deg | rotate.y.rad | rotate.y.deg | rotate.z.rad | rotate.z.deg] #angle	
Description	Rotate the velocity vector around an axis.	
Parameters	rotate.x.rad	Rotate around x coordinate direction (angle given in radians).
	rotate.x.deg	Rotate around x coordinate direction (angle given in degrees).
	rotate.y.rad	Rotate around y coordinate direction (angle given in radians).
	rotate.y.deg	Rotate around y coordinate direction (angle given in degrees).
	rotate.z.rad	Rotate around z coordinate direction (angle given in radians).
	rotate.z.deg	Rotate around z coordinate direction (angle given in degrees).
	#angle	Rotation angle.
Examples	-disturb rotate.x.deg 90	
	-disturb.all rotate.y.rad 1	

Table 10.142: Geometry disturb rotate option

Manipulation based on X	
Syntax	[-disturb | -disturb.all] {u.cos.x.shift | v.cos.x.shift | w.cos.x.shift | p.cos.x.shift | T.cos.x.shift} #a #b #c #d
Description	Manipulate the corresponding primitive using the function: a+b*cos(c*x+d).
Parameters	u.cos.x.shift — Manipulate the x coordinate direction velocity.
	v.cos.x.shift — Manipulate the y coordinate direction velocity.
	w.cos.x.shift — Manipulate the z coordinate direction velocity.
	p.cos.x.shift — Manipulate the pressure.
	T.cos.x.shift — Manipulate the temperature.
	#a — Set the #a parameter.
	#b — Set the #b parameter.
	#c — Set the #c parameter.
	#d — Set the #d parameter.
Examples	-disturb v.cos.x.shift 0 1 1 0

Table 10.143: Geometry disturb manipulation based on the X coordinate option

Manipulation based on Y		
Syntax	[-disturb | -disturb.all] {u.cos.y.shift | v.cos.y.shift | w.cos.y.shift | p.cos.y.shift | T.cos.y.shift} #a #b #c #d	
Description	Manipulate the corresponding primitive using the function: a+b*cos(c*y+d).	
Parameters	u.cos.y.shift	Manipulate the x coordinate direction velocity.
	v.cos.y.shift	Manipulate the y coordinate direction velocity.
	w.cos.y.shift	Manipulate the z coordinate direction velocity.
	p.cos.y.shift	Manipulate the pressure.
	T.cos.y.shift	Manipulate the temperature.
	#a	Set the #a parameter.
	#b	Set the #b parameter.
	#c	Set the #c parameter.
	#d	Set the #d parameter.
Examples	-disturb v.cos.y.shift 0 1 1 0	

Table 10.144: Geometry disturb manipulation based on the Y coordinate option

Manipulation based on Z		
Syntax	[-disturb | -disturb.all] {u.cos.z.shift | v.cos.z.shift | w.cos.z.shift | p.cos.z.shift | T.cos.z.shift} #a #b #c #d	
Description	Manipulate the corresponding primitive using the function: a+b*cos(c*z+d).	
Parameters	u.cos.z.shift	Manipulate the x coordinate direction velocity.
	v.cos.z.shift	Manipulate the y coordinate direction velocity.
	w.cos.z.shift	Manipulate the z coordinate direction velocity.
	p.cos.z.shift	Manipulate the pressure.
	T.cos.z.shift	Manipulate the temperature.
	#a	Set the #a parameter.
	#b	Set the #b parameter.
	#c	Set the #c parameter.
	#d	Set the #d parameter.
Examples	-disturb v.cos.z.shift 0 1 1 0	

Table 10.145: Geometry disturb manipulation based on the Z coordinate option

10.5.7 Vertex Options

$vertex		
Syntax	-vertex $vertex #x #y #z [after.trans]	
Description	Defines a named vertex linked to the current formation. Use after.trans if the given position should be considered after the initial formation transformation.	
Parameters	$vertex	Vertex name.
	#x	Vertex x coordinate.
	#y	Vertex y coordinate.
	#z	Vertex z coordinate.
Examples	-vertex p1 -13.7 -14.1 -8.1	

Table 10.146: Geometry vertex $vertex option

10.5.8 Force Options

Impermeable/noslip/slip	
Syntax	-force $force {impermeable \| noslip \| slip} [$zone {imin | imax | jmin | jmax | kmin | kmax} [{i | j | k}=#min,#max]]... [$zone]... [$body]... [$formation]... [refarea #] [reflen # # #] [refrho #] [refvel #] [refpoint vertexname]
Description	A force is the sum of the aerodynamic forces acting on a collection of wall surfaces. It is used as an input in a 6DOF directive. It can also be used for outputing forces and moments in a log file. Forces can have specific reference values which may be different from the global reference values. Default forces are calculated for the entire computational domain based on the defined boundary conditions. Note, when using complete aerodynamic summation ([$zones]...), the wall type in the force option is ignored. The reader is also referred to Section 10.5.4.2 for an explanation on force overlap. This Table continues on the next page.

Parameters	$force	Force name.
	impermeable	Add pressure forces.
	noslip	Add pressure and viscous forces.
	slip	Add slip boundary condition forces.
	$zone	Zone name
	imin, imax	The surface edge.
	jmin, jmax	The surface edge.
	kmin, kmax	The surface edge.
	i, j, k	Keywords for ξ, η, and ζ coordinates.
	#min	Direction lower bound.
	#max	Direction upper bound.
	$zone	Zone.
	$body	Body.
	$formation	Formation.
	refarea #	Specific reference area.
	reflen # # #	Specific reference lengths for C_M calculation.
	refrho #	Specific reference density for dynamic pressure calculation.
	refvel #	Specific reference velocity for dynamic pressure calculation.
	refpoint vertexname	# Specific reference point for moment calculation.
Examples	-force wing1 noslip wing3 kmin i 1 257 j 1 91	

Table 10.147: Geometry force impermeable/noslip option

Back.pressure		
Syntax	-force $force back.pressure $body factor #nx #ny #nz	
Description	Specific to back pressure, should be a part of 6 DOF (obsolete, can use actual force in body frame using the regular 6 DOF options).	
Parameters	$force	Force name.
	back.pressure	The type "Back pressure".
	$body	Body name
	factor	Force factor.
	#nx	x direction component of the force.
	#ny	y direction component of the force.
	#nz	z direction component of the force.
Examples	-force bp-f back.pressure fuselage-body factor 0.5 normal -1 0 0	

Table 10.148: Geometry force back.pressure option

10.5.9 Special Summation Options

Mass Flow		
Syntax	-mass.flow $name [$zone {imin | imax | jmin | jmax | kmin | kmax} [{i | j | k}=#min,#max]]...	
Description	Integrate the mass flow rate on the specified surface.	
Parameters	$name	Surface summation name.
	$zone	Zone name
	imin, imax	The surface edge.
	jmin, jmax	The surface edge.
	kmin, kmax	The surface edge.
	i, j, k	Keywords for ξ, η, and ζ coordinates.
	#min	Direction lower bound.
	#max	Direction upper bound.
Examples	-mass.flow inlet imin j 1 39 k 1 91	

Table 10.149: Geometry mass flow summation option

Heat Flux		
Syntax	-heat.flux $name [$zone {imin | imax | jmin | jmax | kmin | kmax} [{i | j | k}=#min,#max]]...	
Description	Integrate the heat flux on the specified surface.	
Parameters	$name	Surface summation name.
	$zone	Zone name
	imin, imax	The surface edge.
	jmin, jmax	The surface edge.
	kmin, kmax	The surface edge.
	i, j, k	Keywords for ξ, η, and ζ coordinates.
	#min	Direction lower bound.
	#max	Direction upper bound.
Examples	-heat.flux inlet imin j 1 39 k 1 91	

Table 10.150: Geometry heat flux summation option

Momentum		
Syntax	-momentum $name [$zone {imin | imax | jmin | jmax | kmin | kmax} [{i | j | k}=#min,#max]]...	
Description	Integrate the translational/angular momentum flux on the specified surface.	
Parameters	$name	Surface summation name.
	$zone	Zone name
	imin, imax	The surface edge.
	jmin, jmax	The surface edge.
	kmin, kmax	The surface edge.
	i, j, k	Keywords for ξ, η, and ζ coordinates.
	#min	Direction lower bound.
	#max	Direction upper bound.
Examples	-momentum inlet imin j 1 39 k 1 91	

Table 10.151: Geometry momentum summation option

10.6 Solver Options

The solver section options start with “-solve”. The options are listed in the tables that are described in the following sections.

10.6.1 Equation Sets

Equationset		
Syntax	-solve equationset {inviscid | laminar | turbulent}	
Description	Defines the equation set that is solved (see Chapter 2, specifically Sections 2.2.1 and 3.2).	
Parameters	inviscid	Solve the Euler equations assuming inviscid flow (single-component, perfect-gas physical model).
	laminar	Solve the Navier-Stokes equations assuming laminar flow (single-component, perfect-gas physical model).
	turbulent	Solve the Navier-Stokes equations assuming turbulent flow (single-component, perfect-gas physical model).
Examples	-solve equationset inviscid	
	-solve equationset laminar	
	-solve equationset turbulent	

Table 10.152: Solver equationset option

Model.tu SA

Syntax	-solve model.tu {sa | sa.2nd | sa.mp4 | sa.mp5 | sa.weno5} [edwards | 2020 | obsolete] [mlc]
Description	Set the turbulence model to the Spalart-Allmaras model (see Sections 3.2 and 3.3). High order approximations (above 2nd) are not recommended, nor typically required. Unless aware of the implications, it is not recommended to use the mixing layer correction (mlc) options.
Parameters	sa — SA model, first order accurate. sa.2nd — SA model, second order accurate. sa.mp4 — SA model, MP 4^{th} order approximation. sa.mp5 — SA model, MP 5^{th} order approximation. sa.weno5 — SA model, WENO 5^{th} order approximation. edwards — SA model with Edwards correction (see Section 3.3.1). 2020 — The 2020 variant of the model (see Section 3.3.2). obsolete — Old version (kept here for backward compatibility). mlc — Use the SA model with the addition of the mixing layer correction (available for all variants).
Examples	-solve model.tu sa -solve model.tu sa.2nd -solve model.tu sa.weno5 edwards mlc

Table 10.153: Solver model.tu SA option

Model.tu TNT		
Syntax	-solve model.tu {tnt | tnt.2nd | tnt.mp4 | tnt.mp5 | tnt.weno5} [compressibility.correction | compressibility.correction.wall.distance | compressibility.correction.wilcox]	
Description	Set the turbulence model to the k-ω-TNT model (see Sections 3.2 and 3.4.1). High order approximations (above 2nd) are not recommended, nor typically required.	
Parameters	tnt	TNT model, first order accurate.
	tnt.2nd	TNT model, second order accurate.
	tnt.mp4	TNT model, MP 4^{th} order approximation.
	tnt.mp5	TNT model, MP 5^{th} order approximation.
	tnt.weno5	TNT model, WENO 5^{th} order approximation.
	com...tion	TNT model with compressibility correction.
	com...ance	TNT model with compressibility correction, blending function is wall distance based.
	com...cox	TNT model with compressibility correction based on Wilcox.
Examples	-solve model.tu tnt	
	-solve model.tu tnt.2nd	
	-solve model.tu tnt compressibility.correction.wall.distance	

Table 10.154: Solver model.tu TNT option

Model.tu CES

Syntax	-solve model.tu {tnt.ces | tnt.ces.2nd | tnt.ces.mp4 | tnt.ces.mp5 | tnt.ces.weno5} [compressibility.correction | compressibility.correction.wall.distance | compressibility.correction.wilcox]
Description	Set the turbulence model to the hybrid CES model. High order approximations (above 2nd) are not recommended, nor typically required.
Parameters	tnt.ces — CES model, first order accurate. tnt.ces.2nd — CES model, second order accurate. tnt.ces.mp4 — CES model, MP 4^{th} order approximation. tnt.ces.mp5 — CES model, MP 5^{th} order approximation. tnt.ces.weno5 — CES model, WENO 5^{th} order approximation. com...tion — CES model with compressibility correction. com...ance — CES model with compressibility correction, blending function is wall distance based. com...cox — CES model with compressibility correction based on Wilcox.
Examples	-solve model.tu tnt.ces -solve model.tu tnt.ces.2nd -solve model.tu tnt.ces compressibility.correction.wall.distance

Table 10.155: Solver model.tu CES option

Model.tu X-LES

Syntax	-solve model.tu {tnt.xles | tnt.xles.2nd | tnt.xles.mp4 | tnt.xles.mp5 | tnt.xles.weno5} [sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla]
Description	Set the turbulence model to the hybrid RANS/LES X-LES model with k-ω-TNT as the background model (see Sections 3.2, 3.5, and 3.5.2). High order approximations (above 2nd) are not recommended, nor typically required.
Parameters	tnt.xles — X-LES model, first order accurate. tnt.xles.2nd — X-LES model, second order accurate. tnt.xles.mp4 — X-LES model, MP 4^{th} order approximation. tnt.xles.mp5 — X-LES model, MP 5^{th} order approximation. tnt.xles.weno5 — X-LES model, WENO 5^{th} order approximation. sg.delta.volume — Use volume based subgrid scale (see Section 3.5.1.1). sg.delta.max — Use edge length based subgrid scale (see Section 3.5.1.2). sg.delta.lsq — Use the LSQ subgrid scale (see Section 3.5.1.3). sg.delta.omega — Use vorticity aligned subgrid scale (see Section 3.5.1.4). sg.delta.sla — Use shear layer adapted subgrid scale (see Section 3.5.1.5).
Examples	-solve model.tu tnt.xles -solve model.tu tnt.xles.2nd -solve model.tu tnt.xles sg.delta.max

Table 10.156: Solver model.tu X-LES option

Model.tu SST

Syntax	-solve model.tu {sst | sst.2nd | sst.mp4 | sst.mp5 | sst.weno5} [compressibility.correction | compressibility.correction.wall.distance | compressibility.correction.wilcox | axi.compressibility.correction | jet.correction | shock.unsteadiness.correction | sas]
Description	Set the turbulence model to the k-ω-SST model (see Sections 3.2 and 3.4.2). High order approximations (above 2nd) are not recommended, nor typically required.
Parameters	sst — SST model, first order accurate. sst.2nd — SST model, second order accurate. sst.mp4 — SST model, MP 4^{th} order approximation. sst.mp5 — SST model, MP 5^{th} order approximation. sst.weno5 — SST model, WENO 5^{th} order approximation. com...tion — SST model with compressibility correction. com...ance — SST model with compressibility correction, blending function is wall distance based. com...cox — SST model with compressibility correction based on Wilcox. axi...tion — SST model with axisymmetric compressibility correction. jet...tion — SST model with correction for jet flows. sh...tion — Shock unsteadiness correction. sas — SST-SAS model.
Examples	-solve model.tu sst -solve model.tu sst.2nd -solve model.tu sst axi.compressibility.correction

Table 10.157: Solver model.tu SST option

Model.tu DDES

Syntax	-solve model.tu {sst.ddes | sst.ddes.2nd | sst.ddes.mp4 | sst.ddes.mp5 | sst.ddes.weno5} {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla} [gl.delta.volume | gl.delta.max]	
Description	Set the turbulence model to the hybrid RANS/LES DDES model with k-ω-SST as the background model (see Sections 3.2, 3.5, 3.5.3, and 3.5.3.1). High order approximations (above 2nd) are not recommended, nor typically required.	
Parameters	sst.ddes	DDES model, first order accurate.
	sst.ddes.2nd	DDES model, second order accurate.
	sst.ddes.mp4	DDES model, MP 4^{th} order approximation.
	sst.ddes.mp5	DDES model, MP 5^{th} order approximation.
	sst.ddes.weno5	DDES model, WENO 5^{th} order approximation.
	sg.delta.volume	Use volume based subgrid scale (see Section 3.5.1.1).
	sg.delta.max	Use edge length based subgrid scale (see Section 3.5.1.2).
	sg.delta.lsq	Use the LSQ subgrid scale (see Section 3.5.1.3).
	sg.delta.omega	Use vorticity aligned subgrid scale (see Section 3.5.1.4).
	sg.delta.sla	Use shear layer adapted subgrid scale (see Section 3.5.1.5).
	gl.delta.volume	Grid length scale, by volume (default).
	gl.delta.max	Grid length scale by maximal face surface.
Examples	-solve model.tu sst.ddes sg.delta.volume	
	-solve model.tu sst.ddes.2nd sg.delta.sla gl.delta.max	

Table 10.158: Solver model.tu DDES option

Model.tu IDDES

Syntax	-solve model.tu {sst.iddes | sst.iddes.2nd | sst.iddes.mp4 | sst.iddes.mp5 | sst.iddes.weno5} {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla} [gl.delta.volume | gl.delta.max]
Description	Set the turbulence model to the hybrid RANS/LES IDDES model with k-ω-SST as the background model (see Sections 3.2, 3.5, 3.5.3, and 3.5.3.2). High order approximations (above 2nd) are not recommended, nor typically required.
Parameters	sst.iddes — IDDES model, first order accurate. sst.iddes.2nd — IDDES model, second order accurate. sst.iddes.mp4 — IDDES model, MP 4^{th} order approximation. sst.iddes.mp5 — IDDES model, MP 5^{th} order approximation. sst.iddes.weno5 — IDDES model, WENO 5^{th} order approximation. sg.delta.volume — Use volume based subgrid scale (see Section 3.5.1.1). sg.delta.max — Use edge length based subgrid scale (see Section 3.5.1.2). sg.delta.lsq — Use the LSQ subgrid scale (see Section 3.5.1.3). sg.delta.omega — Use vorticity aligned subgrid scale (see Section 3.5.1.4). sg.delta.sla — Use shear layer adapted subgrid scale (see Section 3.5.1.5). gl.delta.volume — Grid length scale, by volume (default). gl.delta.max — Grid length scale by maximal face surface.
Examples	-solve model.tu sst.iddes sg.delta.volume -solve model.tu sst.iddes.2nd sg.delta.sla gl.delta.max

Table 10.159: Solver model.tu IDDES option

Model.tu PANS

Syntax	-solve model.tu {sst.pans | sst.pans.2nd | sst.pans.mp4 | sst.pans.mp5 | sst.pans.weno5} {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla} {fk.const #fk | fk.gah | fk.fay} [gl.delta.volume | gl.delta.max]	
Description	Set the turbulence model to the hybrid RANS/LES PANS model with k-ω-SST as the background model. High order approximations (above 2nd) are not recommended, nor typically required.	
Parameters	sst.pans	PANS model, first order accurate.
	sst.pans.2nd	PANS model, second order accurate.
	sst.pans.mp4	PANS model, MP 4^{th} order approximation.
	sst.pans.mp5	PANS model, MP 5^{th} order approximation.
	sst.pans.weno5	PANS model, WENO 5^{th} order approximation.
	sg.delta.volume	Use volume based subgrid scale (see Section 3.5.1.1).
	sg.delta.max	Use edge length based subgrid scale (see Section 3.5.1.2).
	sg.delta.lsq	Use the LSQ subgrid scale (see Section 3.5.1.3).
	sg.delta.omega	Use vorticity aligned subgrid scale (see Section 3.5.1.4).
	sg.delta.sla	Use shear layer adapted subgrid scale (see Section 3.5.1.5).
	fk.const	Set the f_k function as constant.
	#fk	Constant f_k value.
	fk.gah	Set the f_k function to $f_{k_{gah}}$.
	fk.fay	Set the f_k function to $f_{k_{fay}}$.
	gl.delta.volume	Grid length scale, by volume (default).
	gl.delta.max	Grid length scale by maximal face surface.
Examples	-solve model.tu sst.pans sg.delta.sla fk.const 1	
	-solve model.tu sst.pans.2nd sg.delta.volume fk.gah gl.delta.max	

Table 10.160: Solver model.tu PANS option

Model.tu ADHRL

Syntax	-solve model.tu {sst.adhrl | sst.adhrl.2nd | sst.adhrl.mp4 | sst.adhrl.mp5 | sst.adhrl.weno5} {sg.delta.volume | sg.delta.max | sg.delta.lsq | sg.delta.omega | sg.delta.sla} {fk.const #fk | fk.gah | fk.fay} [gl.delta.volume | gl.delta.max]
Description	Set the turbulence model to the hybrid RANS/LES ADHRL model with k-ω-SST as the background model. High order approximations (above 2nd) are not recommended, nor typically required.
Parameters	sst.adhrl — ADHRL model, first order accurate. sst.adhrl.2nd — ADHRL model, second order accurate. sst.adhrl.mp4 — ADHRL model, MP 4^{th} order approximation. sst.adhrl.mp5 — ADHRL model, MP 5^{th} order approximation. sst.adhrl.weno5 — ADHRL model, WENO 5^{th} order approximation. sg.delta.volume — Use volume based subgrid scale (see Section 3.5.1.1). sg.delta.max — Use edge length based subgrid scale (see Section 3.5.1.2). sg.delta.lsq — Use the LSQ subgrid scale (see Section 3.5.1.3). sg.delta.omega — Use vorticity aligned subgrid scale (see Section 3.5.1.4). sg.delta.sla — Use shear layer adapted subgrid scale (see Section 3.5.1.5). fk.const — Set the f_k function as constant. #fk — Constant f_k value. fk.gah — Set the f_k function to $f_{k_{gah}}$. fk.fay — Set the f_k function to $f_{k_{fay}}$. gl.delta.volume — Grid length scale, by volume (default). gl.delta.max — Grid length scale by maximal face surface.
Examples	-solve model.tu sst.adhrl sg.delta.sla fk.const 1 -solve model.tu sst.adhrl.2nd sg.delta.volume fk.gah gl.delta.max

Table 10.161: Solver model.tu ADHRL option

Model.tu RSM

Syntax	-solve model.tu {rsm | rsm.2nd | rsm.mp4 | rsm.mp5 | rsm.weno5}
Description	Set the turbulence model to the SSG/LRR-ω model (see Sections 3.2 and 3.9). Selecting the SSG/LRR-ω model explicitly invokes the use of exact.wall.distance (see Table 10.212). High order approximations (above 2nd) are not recommended, nor typically required.
Parameters	rsm — RSM model, first order accurate. rsm.2nd — RSM model, second order accurate. rsm.mp4 — RSM model, MP 4^{th} order approximation. rsm.mp5 — RSM model, MP 5^{th} order approximation. rsm.weno5 — RSM model, WENO 5^{th} order approximation.
Examples	-solve model.tu rsm -solve model.tu rsm.2nd

Table 10.162: Solver model.tu RSM option

Model.tu SST.gammaRt		
Syntax	-solve model.tu {sst.gammaRt | sst.gammaRt.2nd | sst.gammaRt.mp4 | sst.gammaRt.mp5 | sst.gammaRt.weno5} [langtry | menter | frauholz]	
Description	Set the turbulence model to the k-ω-SST-γ-R_θ transition model (langtry is the default). High order approximations (above 2nd) are not recommended, nor typically required.	
Parameters	sst.gammaRt	k-ω-SST-γ-R_θ model, first order accurate.
	sst.gammaRt.2nd	k-ω-SST-γ-R_θ model, second order accurate.
	sst.gammaRt.mp4	k-ω-SST-γ-R_θ model, MP 4^{th} order approximation.
	sst.gammaRt.mp5	k-ω-SST-γ-R_θ model, MP 5^{th} order approximation.
	sst.gammaRt.weno5	k-ω-SST-γ-R_θ model, WENO 5^{th} order approximation.
	langry	Original model by Langtry (default).
	menter	Menter modification.
	frauholz	Frauholz modification, suitable for hypersonic flows.
Examples	-solve model.tu sst.gammaRt	
	-solve model.tu sst.gammaRt.2nd menter	

Table 10.163: Solver model.tu SST.gammaRt option

Model.tu RSM.gammaRt	
Syntax	-solve model.tu {rsm.gammaRt | rsm.gammaRt.2nd | rsm.gammaRt.mp4 | rsm.gammaRt.mp5 | rsm.gammaRt.weno5} [nie | frauholz]
Description	Set the turbulence model to the k-ω-RSM-γ-R_θ transition model (nie is the default). High order approximations (above 2nd) are not recommended, nor typically required.
Parameters	rsm.gammaRt — SSG/LRR-ω-γ-R_θ model, first order accurate.
	rsm.gammaRt.2nd — SSG/LRR-ω-γ-R_θ model, second order accurate.
	rsm.gammaRt.mp4 — SSG/LRR-ω-γ-R_θ model, MP 4^{th} order approximation.
	rsm.gammaRt.mp5 — SSG/LRR-ω-γ-R_θ model, MP 5^{th} order approximation.
	rsm.gammaRt.weno5 — SSG/LRR-ω-γ-R_θ model, WENO 5^{th} order approximation.
	nie — Model by Nie (default).
	frauholz — Frauholz modification, suitable for hypersonic flows.
Examples	-solve model.tu rsm.gammaRt
	-solve model.tu rsm.gammaRt.2nd frauholz

Table 10.164: Solver model.tu RSM.gammaRt option

PANS.reset.averages		
Syntax	-solve pans.reset.averages [#pos]	
Description	Allow reseting the velocity averages for the PANS model without #pos reset at the beginning of the current simulation.	
Parameters	#pos	Time step to reset the averages.
Examples	-solve pans.reset.averages	

Table 10.165: Solver pans.reset.averages option

Realize.rsm		
Syntax	-solve {realize.rsm.omin | realize.rsm.simple}	
Description	Sets a realizability condition for RSM.	
Parameters	realize.rsm.omin	Sets an ω-minimal condition for RSM realizability (default).
	realize.rsm.simple	Sets a simple RSM realizability condition (refrain from using, kept here for backward compatibility).
Examples	-solve realize.rsm.simple	

Table 10.166: Solver realize.rsm option

Realize.rsm.chimera.old	
Syntax	-solve realize.rsm.chimera.old
Description	Sets the old realizability condition for Chimera cells in RSM (refrain from using, kept here for backward compatibility).
Parameters	N/A
Examples	-solve realize.rsm.chimera.old

Table 10.167: Solver realize.rsm.chimera.old option

Trb.ambient.sustain	
Syntax	-solve trb.ambient.sustain
Description	Adds ambient turublence sustaining terms that counteract the natural free-stream decay of turbulence.
Parameters	N/A
Examples	-solve trb.ambient.sustain

Table 10.168: Solver trb.ambient.sustain option

Sigma.k/Sigma.o		
Syntax	-solve [sigma.k | sigma.o] #sigma	
Description	Overrides the default $k - \omega$ model parameters (see Section 3.5.2, use with care).	
Parameters	sigma.k	Set the coefficient σ_k (default is 1.5).
	sigma.o	Set the coefficient σ_ω (default is 2.0).
	#sigma	Value for σ_k or σ_ω.
Examples	-solve sigma.k 1.3	

Table 10.169: Solver sigma option

Ausm.kp/Ausm.ku		
Syntax	-solve [ausm.kp | ausm.ku] #k	
Description	Defines AUSM$^+$-up constants (see Section 4.2.2).	
Parameters	kp	Set the constant kp (default is 0.25). Use with care!
	ku	Set the constant ku (default is 0.75). Use with care!
Examples	-solve kp 0.5	

Table 10.170: Solver ausm.kp/ausm.ku option

10.6.2 Schemes and Order

Scheme	
Syntax	-solve scheme {exp | rk3 | rk4 | rk3.tvd | pgs | lgs | adi | ddadi | ddadii {#} | {rkimp.3 | rkimp.5}}
Description	Defines the time marching scheme that is used to advance the solution in (pseudo-) time (see Section 4.6). Setting the rkimp.3 or rkimp.5 options invokes a Runge-Kutta implicit procedure for any of the implicit time marching schemes.
Parameters	exp — Explicit Euler (see Section 4.6.2.1). rk3 — 3^{rd} order Runge-Kutta (see Section 4.6.2.2). rk4 — 4^{th} order Runge-Kutta (see Section 4.6.2.2).. rk3.tvd — Total variation diminishing 3^{rd} order Runge-Kutta. pgs — Point Gauss-Seidel (see Section 4.6.3.1). lgs — Line Gauss-Seidel (see Section 4.6.3.2). adi — Alternate direction implicit (see Section 4.6.3.3). ddadi — Diagonally dominant ADI (see Section 4.6.3.3). ddadii # — Diagonally dominant ADI with # sub-iterations. rkimp.3 — Third order Runge-Kutta implicit. rkimp.5 — Fifth order Runge-Kutta implicit.
Examples	-solve scheme lgs -solve scheme ddadii 3

Table 10.171: Solver scheme option

Scheme.tu	
Syntax	-solve scheme.tu {pgs | lgs | adi | ddadi | ddadii {#}}
Description	Defines the scheme to use for the turbulence equations time integration. Unless defined, the turbulence model equations are solved together with the mean-flow equations, using the scheme that is defined in Table 10.171. When defined, the turbulence scheme is separate from the mean-flow scheme. Unrecommended to use unless solving the mean-flow with an explicit scheme which is by itself not recommended
Parameters	pgs — Point Gauss-Seidel. lgs — Line Gauss-Seidel. adi — Alternate direction implicit. ddadi — Diagonally dominant ADI. ddadii # — Diagonally dominant ADI with # sub-iterations.
Examples	-solve scheme.tu lgs

Table 10.172: Solver scheme.tu option

B2	
Syntax	-solve b2
Description	Use the b2 scheme for implicit simulation (see Section 4.6.4).
Parameters	b2 — Use the b2 scheme (the default is not to use).
Examples	-solve b2

Table 10.173: Solver B2 option

Precondition		
Syntax	-solve precondition [local | #kvr]	
Description	Low-Mach preconditioning (See Section 4.8 for details).	
Parameters	local	Locally evaluate the reference velocity, V_{ref} (see Equation (4.144)).
	#kvr	Change the reference velocity coefficient (default is $\sqrt{3}$, see Equation (4.144)).
Examples	-solve precondition	

Table 10.174: Solver precondition option

Order		
Syntax	-solve order {1st | uw2 | uw3}	
Description	Defines order of spatial accuracy. The reader is referred to Section 4.2.7 for a description of high order schemes.	
Parameters	1st	1^{st} order (default).
	uw2	Upwind 2^{nd} order.
	uw3	Upwind 3^{rd} order biased.
Examples	-solve order uw3	

Table 10.175: Solver order option

Order (MP-4)		
Syntax	-solve order {mp4.cons | mp4.prim | mp4.char}	
Description	Use MP-4 high order approximation. The reader is referred to Section 4.2.8.1 for a description of high order approximations.	
Parameters	mp4.cons	Use MP 4^{th} order approximation for variables reconstruction; apply using the conservative variables.
	mp4.prim	Use MP 4^{th} order approximation for variables reconstruction; apply using the primitive variables (recommended).
	mp4.char	Use MP 4^{th} order approximation for variables reconstruction; apply using the characteristic variables.
Examples	-solve order mp4.cons	

Table 10.176: Solver order option (MP-4)

MP4.epsilon		
Syntax	-solve mp4.epsilon #eps	
Description	Set the dimensionless ϵ for MP-4	
Parameters	#eps	Dimensionless ϵ.
Examples	-solve mp4.epsilon 1e-5	

Table 10.177: Solver mp4.epsilon option

Order (MP-5)

Syntax	-solve order {mp5.cons | mp5.prim | mp5.char}	
Description	Use MP-5 high order approximation. The reader is referred to Section 4.2.8.1 for a description of high order approximations.	
Parameters	mp5.cons	Use MP 5^{th} order approximation for variables reconstruction; apply using the conservative variables.
	mp5.prim	Use MP 5^{th} order approximation for variables reconstruction; apply using the primitive variables (recommended).
	mp5.char	Use MP 5^{th} order approximation for variables reconstruction; apply using the characteristic variables.
Examples	-solve order mp5.cons	

Table 10.178: Solver order option (MP-5)

MP5.epsilon		
Syntax	-solve mp5.epsilon #eps	
Description	Set the dimensionless ϵ for MP-5	
Parameters	#eps	Dimensionless ϵ.
Examples	-solve mp5.epsilon 1e-5	

Table 10.179: Solver mp5.epsilon option

Order (WENO-5)		
Syntax	-solve order {weno5.cons | weno5.prim | weno5.char}	
Description	Use WENO-5 high order approximation. The reader is referred to Section 4.2.8.2 for a description of high order approximations.	
Parameters	weno5.cons	Use WENO 5^{th} order approximation for variables reconstruction; apply using the conservative variables.
	weno5.prim	Use WENO 5^{th} order approximation for variables reconstruction; apply using the primitive variables.
	weno5.char	Use WENO 5^{th} order approximation for variables reconstruction; apply using the characteristic variables.
Examples	-solve order weno5.cons	

Table 10.180: Solver order option (WENO-5)

WENO5.epsilon	
Syntax	-solve weno5.epsilon #eps
Description	Set the dimensionless ϵ for WENO-5 (see Section 4.2.8.2).
Parameters	#eps Dimensionless ϵ.
Examples	-solve weno5.epsilon 1e-5

Table 10.181: Solver weno5.epsilon option

Shock.sensor	
Syntax	-solve shock.sensor [#xi]
Description	Damp the high-order limiter in the vicinity of shock waves where #xi is a parameter in the range [0-1] determining the TVD blend of the shock-sensor. Recommended for stabilizing the simulation of high Mach number flows. Default value of #xi is 0.8
Parameters	#xi Shock sensor TVD blend.
Examples	-solve shock.sensor 0.7

Table 10.182: Solver shock.sensor option

No.dt.sweep		
Syntax	-solve no.dt.sweep {i | j | k}	
Description	Remove computational direction from virtual time step calculation. Usually used for axisymmetric runs (in 3D mode) to remove the radial influence on the time step.	
Parameters	i	Remove ξ direction from virtual time step calculation.
	j	Remove η direction from virtual time step calculation.
	k	Remove ζ direction from virtual time step calculation.
Examples	-solve no.dt.sweep k	

Table 10.183: Solver no.dt.sweep option

Limiter		
Syntax	-solve {limiter | limiter.all} {vanalbada | koren | minmod | mlp | vanalbada.p | vanalbada.p.all | none | 1st} [#e]	
Description	Define the limiter (Section 4.2.7). If limiter.all is specified then the selected limiter is also used for all model (*e.g.*, turbulence) equations. Defaults are vanalbada and $\#e = 1.0e-20$.	
Parameters	vanalbada	Limiter by van Albada (default).
	koren	Limiter by Koren.
	minmod	Minmod limiter.
	mlp	MLP limiter.
	vanalbada.p	Limiter by van Albada with a pressure switch (acting only on the pressure).
	vanalbada.p.all	Limiter by van Albada with a pressure switch.
	none	No limiter.
	1st	Use first order values.
	#e	ϵ (default is $\epsilon = 1.0e-20$).
Examples	-solve limiter koren 1e-10	
	-solve limiter mlp	
	-solve limiter minomd	

Table 10.184: Solver limiter option

Hybrid.central.upwind

Syntax	-solve hybrid.central.upwind {des.[momentum.]{2nd|4th} | ducros.[momentum.]{2nd|4th} | larsson.[momentum.]{2nd|4th} travin.[momentum.]{2nd|4th} | fsm.[momentum.]{2nd|4th}} #sigma_0 [#sigma_0_me]	
Description	Hybrid central differencing with upwind approximations. Use together with k-ω hybrid models only!	
Parameters	des	Use the DES based blending function (see Section 4.3.1.1). Use together with DDES or IDDES hybrid models only!
	ducros	Use the ducros blending function (see Section 4.3.1.2).
	larsson	Use the Larsson blending function (see Section 4.3.1.3).
	travin	Use the Travin blending function (see Section 4.3.1.4).
	fsm	Use the FSM blending function (see Section 4.3.1.5).
	2nd	2^{nd} order central differencing.
	4th	4^{th} order central differencing.
	momentum	Apply hybrid central upwind to the momentum equation only.
	#sigma_0	Lower limit of the blending function.
	#sigma_0_me	When added, lower limit of the continuity and energy equations blending function.
Examples	-solve hybrid.central.upwind des.momentum.4th 0.1	

Table 10.185: Solver hybrid.central.upwind option

Hybrid.shield		
Syntax	-solve hybrid.shield #wall-distance-length [hybrid.central.upwind | central.upwind | rans.les | subgrid.scale.filter]	
Description	Shield the upwind blending or the hybrid turbulence model (ot both). To apply shield to both use the #wall-distance-length option only.	
Parameters	#wall-distance-length	Shield wall distance (in physical coordinates, *i.e.,* actual distance).
	hybrid.central.upwind	Apply shield to the hybrid central upwind blending only.
	central.upwind	Same as hybrid.central.upwin.
	subgrid.scale.filter	Apply shield to the hybrid subgrid scale filter only.
	rans.les	Same as subgrid.scale.filter.
Examples	-solve hybrid.shield 0.01	

Table 10.186: Solver hybrid.shield option

Convrhs

Syntax	-solve convrhs {hllc.davis | hllc.roe | ausm | ausm.up | ausm.jcam13 | ausm.dv | ausm.pw | slau | slau2 | ldfss.01 | ldfss.jcam13 | aufsr | aufsr+ | aufsr+.jcam13 | steger | maps [delta = #delta] }	
Description	Set the convective flux (see Section 4.2).	
Parameters	hllc.davis	HLLC with signal velocities (S_L, S_R) by Davis (see Section 4.2.1). In general, not recommended for use due to increased dissipation.
	hllc.roe	HLLC with signal velocities (S_L, S_R) by Roe (see Section 4.2.1).
	ausm	AUSM$^+$-up (see Section 4.2.2, same as ausm.up, kept here for backward compatibility).
	ausm.up	AUSM$^+$-up (all-mach, low-diffusion scheme, suitable for a wide range of flows, see Section 4.2.2).
	ausm.jcam13	AUSM-JCAM13 (Only for unsteady low-Mach number flows, see Section 4.2.2.2).
	ausm.dv	AUSM-DV (recommended for supersonic flows, see Section 4.2.2.3).
	ausm.pw	AUSMPW+N (only for unsteady low-Mach number flows).
	slau/slau2	Simple Low dissipation AUSM (all speed schemes). SLAU2 uses a modified pressure flux.
	ldfss.01	LDFSS (2001 version, all-Mach, low-diffusion scheme, suitable for a wide range of flows, see Section 4.2.3).
	ldfss.jcam13	JCAM13 modified version of LDFSS (Only for unsteady low-Mach number flows, see Section 4.2.3.1).
	aufsr	AUFSR (low-diffusion scheme, see Section 4.2.4).
	aufsr+	AUFSR+ (all-mach, low-diffusion scheme, see Section 4.2.4.1).
	aufsr+.jcam13	JCAM13 modified version of AUFSR+.
	steger	Steger-Warming flux vector splitting (see Section 4.5.1). Highly dissipative, recommended only as a means to initialize complex simulations.
	maps	MAPS flux difference splitting.
	#delta	MAPS dissipation parameter (default is 0.0).
Examples	-solve convrhs ausm	

Table 10.187: Solver convrhs option

Dx.div.dt.coef		
Syntax	-solve dx.div.dt.coef #constant	
Description	Set the constant for the $\frac{\Delta x_c}{\Delta t} f_t$ element in the AUSM-JCAM13 scheme (see Section 4.2.2.2, Equation 4.25).	
Parameters	#constant	The constant C in Equation 4.25.
Examples	-solve dx.div.dt.coef 0.5	

Table 10.188: Solver dx.div.dt.coef option

JCAM13.mach.correction		
Syntax	-solve jcam13.mach.correction $function	
Description	Set a different JCAM13 Mach correction function in the AUSM-JCAM13 scheme (see Section 4.2.2.2, Equation 4.25). To be used by experienced users only.	
Parameters	$function	The Mach correction function in Equation 4.25. Any of the function supplied by EZAir can be used.
Examples	-solve jcam13.mach.correction tanh	

Table 10.189: Solver jcam13.mach.correction option

Convjacs

Syntax	-solve convjacs {hllc | vanleer | roe | rossow | rossow.lowmach | rossow.lowmach.old | maps [delta = #delta] | steger | analytic}	
Description	Sets the convective flux Jacobians (for implicit simulations, see Section 4.6.3).	
Parameters	hllc	Use HLLC Jacobians on the LHS. Uses improved UPC Jacobians for the turbulence model equation set (use only with the HLLC convective flux, suitable for low-Mach number flows if invoked with preconditioning).
	vanleer	Use van Leer Jacobians on the LHS (generally recommended for use with all convective fluxes, and all flows, except for low-Mach number flows).
	roe	Use Roe Jacobians on the LHS.
	rossow	Use Jacobians by Rossow on the LHS.
	rossow.lowmach	Use low-Mach number Jacobians by Rossow on the LHS (recommended choice for low-Mach number flows). The option "rossow.lowmach.old" is kept for backward compatibility.
	maps	Use MAPS Jacobians on the LHS (recommended choice for unsteady, low-Mach number flows in conjunction with the jcam13 convective fluxes).
	#delta	MAPS dissipation parameter (default is 0.0).
	steger	Use Steger-Warming Jacobians on the LHS (recommended for supersonic flows, see Section 4.5.1).
	analytic	Use analytic Jacobians.
Examples	-solve convjacs maps	

Table 10.190: Solver convjacs option

Convjacs.tu

Syntax	-solve convjacs.tu 2021
Description	Sets the turbulence model equations convective flux Jacobians to the old UPC Jacobians (Kept for backward compatibility).
Parameters	N/A
Examples	-solve convjacs.tu 2021

Table 10.191: Solver convjacs.tu option

Visgrad

Syntax	-solve visgrad {metrics | thin.layer | thin.metrics}	
Description	Sets the method for evaluating the gradient for viscous estimation, default is thin.metrics (see Section 4.4).	
Parameters	metrics	Use metrics when evaluating the viscous terms for both LHS and RHS
	thin.layer	Use thin layer approximation when evaluating the viscous terms for both the LHS and RHS.
	thin.metrics	Use thin layer approximation on the LHS and metrics on the RHS (default)
Examples	-solve visgrad thin.layer	

Table 10.192: Solver visgrad option

Visgrad.tu	
Syntax	-solve visgrad.tu thin.layer
Description	Force the turbulent face gradients to be evaluated using thin-layer.
Parameters	N/A
Examples	-solve visgrad.tu thin.layer

Table 10.193: Solver visgrad.tu option

10.6.3 Time Step

CFL		
Syntax	-solve cfl #cfl_initial [#cfl_iters [linear | exponential | tangential]] #cfl_final]...	
Description	Sets the Navier-Stokes/Mean Flow CFL number (see Section 4.6.1).	
Parameters	#cfl_initial	Initial CFL number for the mean flow.
	#cfl_iters	Number of iteration for which the CFL number grows from #cfl_initial to #cfl_final.
	linear	Linear growth of CFL number.
	exponential	Exponential growth of CFL number.
	tangential	Hyperbolic tangent growth of CFL number.
	#cfl_final	CFL number for the mean flow.
Examples	-solve cfl 3	
	-solve cfl 5 50 linear 10	
	-solve cfl 0.1 3 exponential 3	
	-solve cfl 5 50 exponential 10 100 linear 20	

Table 10.194: Solver CFL option

CFL.TU		
Syntax	-solve cfl.tu {infinite | finite | {#cfl_initial [#cfl_iters [linear | exponential | tangential]]... #cfl_final]}}	
Description	Sets the turbulent CFL number of the simulation (for dual-time or unsteady explicit schemes (the default is the mean flow CFL).	
Parameters	infinite	Infinite turbulent CFL number (default for steady flows).
	finite	Finite turbulent CFL number (use the mean flow CFL number).
	#cfl_initial	Initial CFL number for the turbulence model equations.
	#cfl_iters	Number of iterations for which the CFL number grows from #cfl_initial to #cfl_final.
	linear	Linear growth of CFL number.
	exponential	Exponential growth of CFL number.
	tangential	Hyperbolic tangent growth of CFL number.
	#cfl_final	Final CFL number for the turbulence model equations.
Examples	-solve cfl.tu 0.1 100 linear 100	

Table 10.195: Solver CFL.TU option

Dualtime.blend.alpha		
Syntax	-solve dualtime.blend.alpha #alpha	
Description	Blending activation threshold parameter for dualtime.blend scheme (see Table 10.15).	
Parameters	#alpha	Activation threshold parameter (default is 0.0).
Examples	-solve dualtime.blend.alpha 0.1	

Table 10.196: Solver dualtime.blend.alpha option

10.6.4 Reference Quantities

Reflong	
Syntax	-solve reflong #longitudinal-length
Description	Sets the longitudinal reference length. Used for evaluating the Reynolds number.
Parameters	#longitudinal-length — Reference length (default is 1.0).
Examples	-solve reflong 2.0

Table 10.197: Solver reflong option

Refarea	
Syntax	-solve refarea #area
Description	Sets the reference area. Used for making the forces and moments dimensionless.
Parameters	#area — Reference area (default is 1.0).
Examples	-solve refarea 330.0

Table 10.198: Solver refarea option

Reflen		
Syntax	-solve reflen #lenx #leny #lenz	
Description	Sets the reference length. Used for calculating the moment coefficients.	
Parameters	#lenx	Reference length for the x coordinate direction (default is 1.0).
	#leny	Reference length for the y coordinate direction (default is 1.0).
	#lenz	Reference length for the z coordinate direction (default is 1.0).
Examples	-solve reflen 0.5 0.5 0.5	

Table 10.199: Solver reflen option

Refpoint		
Syntax	-solve refpoint #x #y #z	
Description	Sets the reference point for moments evaluation.	
Parameters	#x	Reference point for the x coordinate direction (default is 0.0).
	#y	Reference point for the y coordinate direction (default is 0.0).
	#z	Reference point for the z coordinate direction (default is 0.0).
Examples	-solve refpoint 1.0 0.0 0.0	

Table 10.200: Solver refpoint option

Refvel	
Syntax	-solve refvel #uvel #vvel #wvel
Description	Sets the refernce velocity components that are used by preconditioning schemes. If set, supersedes the reference velocity for the mean flow equations. Free-stream velocity components are used by default.
Parameters	#uvel X component of reference velocity. #vvel Y component of reference velocity. #wvel Z component of reference velocity.
Examples	-solve refvel 300.0 0.0 20.0

Table 10.201: Solver refvel option

Refvel.trb	
Syntax	-solve refvel.trb #vel
Description	Sets the refernce turbulence velocity. If set, supersedes the reference velocity for the mean flow equations. Free-stream velocity components are used by default.
Parameters	#vel Reference velocity.
Examples	-solve refvel.trb 200.0

Table 10.202: Solver refvel.trb option

Refmach		
Syntax	-solve refmach #mx #my #mz	
Description	Set the reference Mach number components for the mean flow equations that are used by preconditioning schemes and turbulence intensity. Supersede the free-stream Mach number components that are used by default.	
Parameters	refmach	Set reference Mach number components for the mean flow equations.
	#mx	X component of reference Mach number.
	#my	Y component of reference Mach number.
	#mz	Z component of reference Mach number.
Examples	-solve refmach 0.6 0.0 0.05	

Table 10.203: Solver refmach option

10.6.5 Simulation Control

Iters	
Syntax	-solve iters #iterations
Description	Sets the number of solution iterations (sub-iterations in the case of dual-time simulations).
Parameters	#iterations — Number of iterations (default is 500).
Examples	-solve iters 1000

Table 10.204: Solver iters option

Wall.slowstart	
Syntax	-solve wall.slowstart #iters
Description	Wall slow start. Gradually impose the boundary conditions on all walls (blended with free stream conditions).
Parameters	#iters — Number of iterations for slow start process.
Examples	-solve wall.slowstart 50

Table 10.205: Solver wall.slowstart option

DQ.factor		
Syntax	-solve { dq.factor | dq.mss.factor | dq.mom.factor | dq.nrg.factor | dq.trb.factor} #f	
Description	Multiply DQ by a factor, typically used for stabilizing the simulation via under-relaxation (factor < 1).	
Parameters	dq.factor	Factor DQ by #f.
	dq.mss.factor	Factor mass equation DQ by #f.
	dq.mom.factor	Factor momentum equations DQ by #f.
	dq.nrg.factor	Factor energy equation DQ by #f.
	dq.trb.factor	Factor turbulence model equations DQ by #f.
	#f	Factor.
Examples	-solve dq.factor 0.7	

Table 10.206: Solver dq.factor option

SOR		
Syntax	-solve { sor.mf | sor.tu | sor.all} #relax	
Description	Successive under-relaxation (0 < #relax < 1) or over-relaxation (1 < #relax < 2) of linear solver for the mean flow and/or turbulence model equations.	
Parameters	sor.mf	Successive under/over-relaxation for the mean flow equations.
	sor.tu	Successive under/over-relaxation for the turbulence model equations.
	sor.all	Successive under/over-relaxation for all equation sets.
	#relax	Relaxation parameter.
Examples	-solve sor.all 1.5	

Table 10.207: Solver SOR option

Sweep		
Syntax	-solve sweep {i | j | k | .i | .j | .k | ij | ik | jk | .ij | .ik | .jk | .ji | ijk | ikj | jik | jki | kij | kji | .ijk | .ikj | .jik | .jki | .kij | .kji} [reduced]	
Description	Control the direction of the implicit solution sweeps within an iteration. When a "." appears before the index/indices the inversion direction is reversed. For better stability and convergence, the user should attempt to have the 'walls' direction as the last sweep.	
Parameters	i	Inversion in the ξ direction only.
	j	Inversion in the η direction only.
	k	Inversion in the ζ direction only.
	ij	Inversion in the ξ and η directions only.
	ik	Inversion in the ξ and ζ directions only.
	jk	Inversion in the η and ζ directions only.
	ji	Inversion in the η and ξ directions only.
	ijk	Regular sweep order (default).
	ikj	Specific sweep order.
	jik	Specific sweep order.
	jki	Specific sweep order.
	kij	Specific sweep order.
	kji	Specific sweep order (reversed sweep order).
	reduced	Conduct forward sweeps only.
Examples	-solve sweep kji	

Table 10.208: Solver sweep option

Convstop	
Syntax	-solve convstop {$logitem {< | >} { | #}}...
Description	Sets convergence criteria for the virtual time iterations. It is possible to request multiple convergence criterions. Writing multiple criterions in a single line will perform a logical ‘and’ between those criterions. Writing the criterions in seperate commands will perform a logical ‘or’ between them. When a convergence criteria has been fulfilled the simulation exits in a steady state simulation or moves on to the next time step in a time accurate simulation.
Parameters	$logitem — Every parameter that is available in the ‘Log Options’ table. See Table B.1 in Appendix B. < — Lower than. > — Greater than. # — Value.
Examples	-solve convstop log(ns) < -5 -solve convstop log(L) < -5 log(D) <-5 -solve convstop mt > 0.05

Table 10.209: Solver convstop option

Axisymmetric	
Syntax	-solve axisymmetric [uniform.dt] [angle.deg #deg]
Description	Enables the simulation of axisymmetric (3D) flow using a 2D grid. Assumes that the axis is at $Y = 0, Z = 0$, and that the supplied 2D grid represents a slice that lies entirely in the X-Y plane (ξ-η). Works only with linear turbulence models (*i.e.*, k-ω-TNT, k-ω-SST, X-LES, SA). Note that in the axisymmetric mode, the forces are calculated for the entire body of revolution, so that mass, moment of inertia, etc., should be input accordingly.
Parameters	uniform.dt — Do not account for circumferential cell size in calculating the local time-step (as in a standard 2D grid). angle.deg — Axisymmetric section size. #deg — Axisymmetric section (default is 1 degree).
Examples	-solve axisymmetric -solve axisymmetric uniform.dt -solve axisymmetric angle.deg 5

Table 10.210: Solver axisymmetric option

Mode		
Syntax	-solve mode {3D | 2D-I | 2D-J | 2D-K}	
Description	Force the simualtion to be 3D, or 2D. For 2DK (x-y) simulations, the computational coordinates must be $\xi - \eta$. For 2DJ (x-z) simulations, the computational coordinates must be $\xi - \zeta$. For 2DI (y-z) simulations, the computational coordinates must be $\eta - \zeta$. The default is automatically identified based on the mesh.	
Parameters	3D	Force 3-D simulation (for example, for axisymmetric simulations with a 2D mesh).
	2D-I	Force 2-D $(y - z)$ simulation.
	2D-J	Force 2-D $(x - z)$ simulation.
	2D-K	Force 2-D $(x - y)$ simulation.
Examples	-solve mode 2D-I	

Table 10.211: Solver mode option

Exact.wall.distance		
Syntax	-solve exact.wall.distance | hiorder.wall.distance	
Description	Calculate the wall distance without assuming that the computational mesh is orthogonal. This option should be used for non-orthogonal meshes (very highly recommended). The runtime of the exact wall distance calculation (*i.e.*, no mesh orthogonality assumed) is estimated at 2.5 times slower than the approximate (default) calculation.	
Parameters	exact.wall.distanc	Exact wall distance calculation.
	hiorder.wall.distar	High order, exact wall distance calculation (based on shape functions).
Examples	-solve exact.wall.distance	
	-solve hiorder.wall.distance	

Table 10.212: Solver exact wall distance option

Freeze.wall.distance	
Syntax	-solve freeze.wall.distance
Description	Freeze the wall distance after the initial calculation.
Parameters	N/A
Examples	-solve freeze.wall.distance

Table 10.213: Solver freeze wall distance option

Axes.impermeable	
Syntax	-solve axes.impermeable
Description	Treat axis boundary conditions as impermeable wall for axi-symmetric simulations. Required to accurately calculate diffusive fluxes of nearby cells.
Parameters	N/A
Examples	-solve axes.impermeable

Table 10.214: Solver axes.impermeable option

Force.freestream.pressure		
Syntax	-solve force.freestream.pressure #pinf	
Description	When integrating forces, subrtract from local pressure in [pa]. Required, for example, when bodies are not closed.	
Parameters	#pinf	Free stream pressure for subtraction (default is 0.0 [pa]).
Examples	-solve force.freestream.pressure 100000	

Table 10.215: Solver force.freestream.pressure option

10.6.6 Flow Conditions

Vel/Mach		
Syntax	-solve {vel | mach} #	
Description	Sets the far field velocity or Mach number.	
Parameters	vel	Set the free stream velocity.
	mach	Set the free stream Mach number.
	#	Free stream velocity or Mach number.
Examples	-solve mach 0.185	
	-solve vel 160	

Table 10.216: Solver vel/mach options

AOA		
Syntax	-solve alpha #aoa	
Description	Sets the far field angle of attack in [deg].	
Parameters	#aoa	Angle of attack in [deg] (default is 0.0 [deg]).
Examples	-solve alpha 2.56	

Table 10.217: Solver aoa option

Beta		
Syntax	-solve beta #beta	
Description	Sets the far field side slip angle in [deg].	
Parameters	#beta	Side slip in [deg] (default is 0.0 [deg]).
Examples	-solve beta -5	

Table 10.218: Solver beta option

u		
Syntax	-solve u #u	
Description	If entered supersedes the far field x direction velocity evaluated in [m/s]	
Parameters	#u	far field x direction velocity in [m/s]
Examples	-solve u 290.0	

Table 10.219: Solver u option

v		
Syntax	-solve v #v	
Description	If entered supersedes the far field y direction velocity evaluated in [m/s]	
Parameters	#v	far field y direction velocity in [m/s]
Examples	-solve v 0.0	

Table 10.220: Solver v option

w	
Syntax	-solve w #w
Description	If entered supersedes the far field z direction velocity evaluated in [m/s]
Parameters	#w far field z direction velocity in [m/s]
Examples	-solve w 10.0

Table 10.221: Solver w option

T	
Syntax	-solve T #te
Description	Sets the far field temperature in [Kelvin].
Parameters	#te Far field temperature in [Kelvin] (default is 288.0 [Kelvin]).
Examples	-solve T 300.0

Table 10.222: Solver T option

p	
Syntax	-solve p #p
Description	Sets the far field pressure in [pa].
Parameters	#p Far field pressure in [pa] (default is 101325 [pa]).
Examples	-solve p 10000.0

Table 10.223: Solver p option

ti		
Syntax	-solve ti #ti	
Description	Sets the far field turbulent intensity in [%].	
Parameters	#ti	Far field turbulent intensity in [%] (default is 0.1 [%]).
Examples	-solve ti 1	

Table 10.224: Solver ti option

mt		
Syntax	-solve mt #mt	
Description	Sets the far field turbulent viscosity in [%] of μ	
Parameters	#mt	Far field turbulent viscosity in [%] of μ (default is 1.0 [%μ]).
Examples	-solve mt 0.5	

Table 10.225: Solver mt option

10.6.7 Gas Constants

EZAir provides the means to select any calorically perfect gas by setting the appropriate constants (the default gas is air). Assuming a perfect gas, the equation of state takes the form (see Section 2.2.3):

$$p = \rho R T \tag{10.1}$$

or, for a calorically perfect gas (the specific heats C_p and C_v are constant):

$$p = \rho\left(\gamma - 1\right) e_I \tag{10.2}$$

Table 10.226 allows to set the gas constants γ and R (default is air). One can also set the Sutherland Law coefficients, C_1 and C_2 in Equation 10.3 using Table 10.227. Setting γ, R, C_1, and C_2 allows to select any calorically perfect gas for the simulation.

$$f = C_1 \frac{T^{\frac{3}{2}}}{T + C_2} \tag{10.3}$$

Gas constants		
Syntax	-solve {gas.gamma | gas.constant} #	
Description	Set the properties of the gas, the ratio of specific heats, γ, and the gas constant, R (default gas is air).	
Parameters	gas.gamma #	Set the ratio of specific heats, γ (default gas is air, $\gamma = 1.4$).
	gas.constant #	Set the gas constant, R (default gas is air, $R = 287.22$).
Examples	-solve gas.gamma 1.4	

Table 10.226: Solver gas constants option

Sutherland's Law		
Syntax	-solve {sutherland.mu | sutherland.ka} #c1 #c2	
Description	Set Sutherland's law parameters of the viscosity & thermal conductivity of the gas (see Section 2.2.4, Equation 2.13).	
Parameters	#c1	Set C_1 (default gas is air, $C_1 = 1.458 \times 10^{-6}$ for μ and $C_1 = 2.495 \times 10^{-3}$ for κ).
	#c2	Set C_2 (default gas is air, $C_1 = 110.3$ for μ and $C_1 = 194$ for κ).
Examples	-solve sutherland.mu c2 110.3	

Table 10.227: Solver Sutherland's constants option

10.7 Multi Grid

Multigrid

Syntax	-solve multigrid #levels #pre-iterations [#defect-correction-iterations] #coarsest-grid-iterations #post-iterations [[all.levels | level #] [do.not.coarse.in.i | do.not.coarse.in.j | do.not.coarse.in.k]...] [prolong.lin | prolong.log] [damp]	
Description	Declare a Multigrid V cycle (see Section 4.9 and Figure 4.2). It is strongly recommended to construct the mesh such that the number of cells in each dimension is dividable by ($2^{\#levels-1}$). The user can override specific zone settings using the directive described in Table 10.100. Note that AMR and dual-time simulations are not supported yet.	
Parameters	#levels	Number of multigrid levels.
	#pre-iterations	Number of iterations prior to coarsening.
	#coarsest-grid-iterations	Number of iterations on the coarsest grid.
	#post-iterations	Number of iterations following interpolation from coarse grids.
	#defect-corrections-iterations	Number of defect correction iterations. Employ reduced-order (1st) iterations on coarse grids, in order to enhance stability in cases involving sharp discontinuities.
	all.levels	Directional directives apply to all levels.
	level	Specific level.
	#	Level number.
	do.not.coarse.in.i	Directional coarsening directive.
	do.not.coarse.in.j	Directional coarsening directive.
	do.not.coarse.in.k	Directional coarsening directive.
	prolong.lin	(Tri)-linear prolongation (interpolation) of coarse-grid corrections (default).
	prolong.log	Positivity-preserving, log-based prolongation (interpolation) of coarse-grid corrections.
	damp	Damping of coarse-grid defect and time-step near discontinuities (shock/interface).
Examples	-solve multigrid 3 1 8 2 -solve multigrid 3 2 10 3	

Table 10.228: Solver multigrid option

FMG		
Syntax	-solve multigrid.init #levels #iterations	
Description	Invoke a preliminary, nested iteration (FMG) initialization process, where #iterations are first conducted on #levels of the coarse grids in the multigrid hierarchy to obtain a 'cheap', improved initial solution on the finest grid (see Section 4.9.2 and Figure 4.3).	
Parameters	#levels	Number of levels.
	#iteration	Number of iterations.
Examples	-solve multigrid.init 4 500	

Table 10.229: Solver mg multigrid.init option

MG Defect Factor

Syntax	-solve {mg.E.factor | mg.E.mss.factor | mg.E.mom.factor | mg.E.nrg.factor | mg.E.trb.factor} #f	
Description	Multiply the mean-flow equations defect (E) by a factor, typically smaller than one, to stabilize the non-linear multigrid process.	
Parameters	mg.E.factor	Multiply all the mean-flow equations defect by the factor.
	mg.E.mss.factor	Multiply the mass equation defect by the factor.
	mg.E.mom.factor	Multiply the momentum equations defects by the factor.
	mg.E.nrg.factor	Multiply the energy equation defect by the factor.
	mg.E.trb.factor	Multiply the turbulence model equation(s) defect(s) by the factor.
	#f	Factor.
Examples	-solve mg.E.factor 0.5	

Table 10.230: Solver mg defect factor option

MG DQF Factor

Syntax	-solve mg.dqf.factor | mg.dqf.mss.factor | mg.dqf.mom.factor | mg.dqf.nrg.factor | mg.dqf.trb.factor #f
Description	Multiplies the coarse-grid-correction (DQF) by a factor, typically smaller than one, to stabilize the non-linear multigrid process.
Parameters	mg.dqf.factor: Multiply all the mean-flow equations DQF by the factor. mg.dqf.mss.factor: Multiply the mass equation DQF by the factor. mg.dqf.mom.factor: Multiply the momentum equations DQF by the factor. mg.dqf.nrg.factor: Multiply the energy equation DQF by the factor. mg.dqf.trb.factor: Multiply the turbulence model equation(s) DQF(s) by the factor. #f: Factor.
Examples	-solve mg.dqf.factor 0.5

Table 10.231: Solver mg DQF factor option

10.8 Boundary Conditions Options

This section contains a description of boundary conditions options. The description is specific to each boundary condition type. However, there are options, or flags, that pertain to all boundary conditions. These flags are in addition to the options that are specific boundary condition type. See Chapter 5 for a description of the available boundary conditions.

10.8.1 Boundary Conditions General Flags

All boundaries have additional flags that are used to set specific conditions per boundary condition: [$u = \# \mid v = \# \mid w = \# \mid p = \# \mid mt = \# \mid T = \# \mid q = \# \mid rq = \#$]. Unless these properties are set, free-stream flow conditions are assumed wherever required. Specific rules apply to wall boundaries (impermeable/no-slip):

1. The default velocity on no-slip boundaries is zero (or grid velocity, in the case of moving grids)

2. The default wall-normal component of the velocity on impermeable boundaries is zero (or grid velocity, in the case of moving grids) .

3. The heat flux rate, q, can be prescribed for walls only, and not together with T.

4. Normal mass injection at a rate of rq [kg/m^2/s] can be set for noslip walls.

5. By default, the wall is assumed adiabatic (see Section 5.2.3). However, when T is set for a wall, that wall is assumed isothermal (see Section 5.2.4).

6. The radiative equilibrium boundary condition, rad.eq, can be prescribed for walls only, and not together with T or q. The required input parameter is the wall emissivity.

The flags: [restrict.by $phantom] can be set for all boundaries. These are used to restrict a boundary condition to a geometrical area on top of the computational coordinates. Only a single restriction is allowed.

Moreover, [i = #min, #max], [j = #min, #max], and [k = #min, #max], can be used to limit any boundary condition to a sub-surface of a grid edge. The option [extended.info] can be also used for printing additional information concerning a specific boundary condition in the log file. Another option is the {x | y | z} = # option. It can be added to every boundary condition as it enforces the specified edge to a specific coordinate set by #. The reader is referred to the example in Table 10.234.

Periodic		
Syntax	-bc {idim | jdim | kdim} {omesh | periodic | 3d.inf}	
Description	Defines omesh or periodic condition on one of the dimensions for the current zone.	
Parameters	idim	ξ direction.
	jdim	η direction.
	kdim	ζ direction.
	omesh	O type mesh (geometrically periodic) in the specified direction.
	periodic	Rotated periodic boundary conditions in the specified direction.
	3d.inf	Non-rotated periodic boundary conditions in the specified direction.
Examples	-bc idim omesh	

Table 10.232: BC periodic option

Axis		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} axis	
Description	Defines axis boundary condition over one of the zone edges	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
	axis	Singular axis geometry
Examples	-bc idim axis	

Table 10.233: BC axis option

Symmetry		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} symmetry	
Description	Defines a symmetry boundary condition over one of the zone edges.	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
	symmetry	Symmetric mesh in the specified direction
Examples	-bc jmin symmetry y = 0.0	
	-bc kmax symmetry	

Table 10.234: BC symmetry option

Emesh		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} {emesh.i | emesh.j | emesh.k} [{i | j | k}=#min,#max...]	
Description	Defines an emesh boundary condition over one of the zone edges. emesh.i Signifies that cell (i) and cell (maxi-i) have a common face. An emesh is a mesh where a cut condition is identified or imposed, *e.g.*, a C type mesh about an airfoil.	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
	emesh.i	Collapsed in the ξ direction.
	emesh.j	Collapsed in the η direction.
	emesh.k	Collapsed in the ζ direction.
	i, j, k	Additional limitations.
	#min	Additional limitations.
	#max	Additional limitations.
Examples	-bc jmin emesh.i	

Table 10.235: BC emesh option

Impermeable/noslip	
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} {impermeable | slip | noslip [omega.menter | omega.wilcox | automatic.wall.func.incompressible | nichols.wall.func]} [force | not.force] [entity | not.entity][distance | not.distance] [dof.by $formation] [{auto.length.tolerance | auto.angle.tolerance.deg | auto.angle.tolerance.rad} | #tolerance] [by.height | by.dsf] [auto.logical.interpolation.from.zone $zone] [auto.force.index | auto.force.distance | auto.force.surface | auto.force.priority]
Description	Defines a wall boundary condition over one of the zone edges. Note, the dof.by $formation option forces the use of the 6dof definitions of the formation $formation instead of it's own zone (formation) 6dof definitions. May be used for example if a blunt body that part of it is a bearing is modeled using a single grid, and it is desired that the no-slip condition over the bearing area will represent the bearing's angular velocity. The parameters auto.length.tolerance, auto.angle.tolerance.deg, and auto.angle.tolerance.rad are used to define tolerances for automatic force overlap and logical interpolation. The parameter auto.logical.interpolation.from.zone sets a zone precedence in searching for automatic logical interpolation. The parameters: auto.force.index, auto.force.distance, auto.force.surface, and priority provide the means to selects various modes for the automatic force overlap.

Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
	impermeable	Enforce an impermeable boundary condition at the associated edge (see Section 5.2.1).
	noslip	Enforce a no-slip boundary condition at the associated edge (see Section 5.2.2).
	slip	Enforce an isothermic slip boundary condition at the associated edge (for hypersonic flows with appropriate flow conditions only).
	force	Sum the forces over the wall (default).
	not.force	Do not sum the forces over the walls.
	entity	Wall that is used for hole cutting (default).
	not.entity	Wall that is not used for hole cutting.
	distance	Wall that is used for wall distance calculations (default for no slip).
	not.distance	Wall that is not used for wall distance calculations (default for impermeable).
	dof.by $formation	See description above.

Parameters (continued)	auto.length.tolerance	Defines tolerances for automatic force overlap and logical interpolation.
	auto.angle.tolerance.deg	Defines tolerances for automatic force overlap and logical interpolation.
	auto.angle.tolerance.rad	Defines tolerances for automatic force overlap and logical interpolation.
	#tolerance	Defines tolerances for automatic force overlap and logical interpolation.
	by.height	Set interpolation strategy based on height.
	by.dsf	Set interpolation strategy based on donor suitability function.
	auto.logical.interpolation...zone	Set a zone precedence in searching for automatic logical interpolation.
	$zone	Zone.
	auto.force.distance	Priority is given to the face closest to its' zone center.
	auto.force.surface	Priority is given to the face with the smaller area.
	auto.force.index	Priority is given to the face whose zone index is lower.
	auto.force.priority	Absolute priority over other faces.

Examples	-bc kmin noslip j 29 89 -bc imin noslip nichols.wall.func k = 1,31

Table 10.236: BC impermeable/noslip option

Inlet/outlet		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} {turkel | riemann | inout | turkel.inlet | riemann.inlet | inlet | pressure.inlet | turkel.outlet | riemann.outlet | outlet [force.outflow]}	
Description	Define an inlet/outlet boundary condition over one of the zone edges. By default, the direction (inlet/outlet) is automatically selected based on the direction of the dot product between the free-stream flow and the local boundary normal vector. The characteristic (riemann/turkel) boundary conditions may be forced to be inlet or outlet, via the '.inlet', '.outlet' additions. The force.outflow directive forces the flow to exit the domain at outlet boundaries by rotating it as required. Default is automatic detection based on the boundary condition velocity.	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
	turkel	Use Turkel type simplified characteristic relations.
	riemann	Use Riemann invariants.
	inout	Use inout condition.
	turkel.inlet	Explicit variant of turkel.
	riemann.inlet	Explicit variant of Riemann invariants.
	inlet	Explicit variant of inout condition.
	pressure.inlet	Use stagnation inlet condition (set pressure is the stagnation pressure).
	turkel.outlet	Explicit variant of turkel.
	riemann.outlet	Explicit variant of Riemann invariants.
	outlet	Explicit variant of inout condition.
	force.outflow	Force the flow to point outward.
Examples	-bc imax turkel -bc imin outlet force.outflow	

Table 10.237: BC inlet/outlet option

Massflux.in		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} massflux.in $massflow {massflux.mdot #mdot} {massflux.Tt #Ttotal}	
Description	Define an inlet mass-flux boundary condition over one of the zone edges.	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
	$massflow	Surface summation name (see Table 10.149).
	massflux.mdot	Mass flux directive.
	#mdot	Mass flux value.
	massflux.Tt	Total temperature directive.
	#Ttotal	Total temperature value [K].
Examples	-bc imin massflux.in s1 massflux.mdot 0.05 massflux.Tt 600	

Table 10.238: BC massflux.in option

Massflux.out		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} massflux.out $massflow {massflux.mdot #mdot}	
Description	Define an outlet mass-flux boundary condition over one of the zone edges.	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
	$massflow	Surface summation name (see Table 10.149).
	massflux.mdot	Mass flux directive.
	#mdot	Mass flux value.
	massflux.Tt	Total temperature directive.
Examples	-bc imin massflux.out s1 massflux.mdot 0.03	

Table 10.239: BC massflux.out option

Extrap		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} extrap	
Description	Define a zeroth order extrapolation boundary condition over one of the zone edges.	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
Examples	-bc imin extrap	

Table 10.240: BC extrap option

Fixed	
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} fixed [u=#u | v=#v | w=#w | p=#p | T=#T | rq=#rq...] [{p|T|v}.spatial.axis $v1 $v2 {const | linear | sin | cos | tanh | table}] [{p|T|u|v|w}.spatial.{x|y|z|xy|xz|yz} {const | linear | sin | cos | tanh | table | simple.quad | full.quad | nearest.neighbor | shepard{1|2}}]

Description	Defines a fixed boundary condition (prescribed values) over one of the zone edges. Note that free-stream values are used for the fixed-type BC unless specifically overridden by u=#u, v=#v, ... directives. The spatial.axis and spatial.{x|y|z} may be used to indicate a spatial distribution of the primitive flow variables on the edge. Spatial.axis introduces a spatial distribution based on radial distance from the axis defined by the vertices $v1, $v2 while spatial.{x|y|z} introduces a spatial distribution based on the absolute cartesian grid coordinates. Note that v.spatial.axis adds to the magnitude of the velocity in the direction of the vector defined by $v1, $v2, *i.e.*, u_i = mag(v) * (v2.x_i - v1.x_i) / |v2 - v1| The primitives can be additionaly spatially changed based on a single coordinate or two coordinates, where the options are: {p|u|v|w|T}.spatial.{x|y|z|xy|xz|yz}, There are five ways to supply the dual parameter spatial functions as follows: The first one is to define a local two dimensional planar mesh as follows: "simple.quad n m {x1 .. xn} {y1 .. ym} {{z11 .. zn1} .. {zn1 .. zm1}} end.quad" where $x?, y?$ are the mesh coordinates and $z??$ is the function value. The second is the full quad, namely, a two-dimensional PLOT3D function file like as follows "full.quad n m {{x11 .. xn1} .. {xn1 .. xm1}} {{y11 .. yn1} .. {yn1 .. ym1}} {{z11 .. zn1} .. {zn1 .. zm1}} end.full.quad." The third uses the nearest neighbor interpolation as follows: " nearest.neighbor n {x1 y1 z1} .. {xn yn zn} end.nearest.neighbor." The other two options are shepard1 and shepard2 as follows: "shepard{1|2} n {x1 y1 z1} .. {xn yn zn} end.shepard{1|2}"

Parameters	imin, imax, ...	Edge selection.
	u,v,w,p,T	Prescribed flow quantities.
	v1,v2	Vertices defining the axis used in the radial spatial distribution option.
	const	Constant value.
	linear #a #b	The distribution $a + b * s$.
	sin #a #b #c #d	The distribution $a + b * \sin(c * s + d)$.
	cos #a #b #c #d	The distribution $a + b * \cos(c * s + d)$.
	tanh #a #b #c #d	The distribution $a + b * \tanh(c * s + d)$.
	table {#s #y}... end.table	Table whose rows contain location-value pairs.
	simple.quad ...	See above (in the description).
	full.quad ...	See above (in the description).
	nearest.neighbor ...	See above (in the description).
	shepard{1|2} ...	See above (in the description).
	s	Distance from axis in radial distributions, or absolute cartesian coordinate in coordinate distributions.
Examples	-vertex v1 0 0 0 -vertex v2 1 0 0 -bc kmin fixed i = 106, 120 p = 0 u = 0 v = 0 w = 0 p.spatial.axis = v1 v2 linear 2e5 -1e7 v.spatial.axis = v1 v2 linear 2e5 -1e7	

Table 10.241: BC fixed option

Intake		
Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} intake}	
Description	Defines an intake boundary condition over one of the zone edges.	
Parameters	imin	Minimum i direction edge.
	imax	Maximum i direction edge.
	jmin	Minimum j direction edge.
	jmax	Maximum j direction edge.
	kmin	Minimum k direction edge.
	kmax	Maximum k direction edge.
Examples	-bc imin intake	

Table 10.242: BC intake option

Disk.model

Syntax	-bc {imin | jmin | kmin | imax | jmax | kmax} disk.model {axis $vertex1 $vertex2} [thrust # | dCT.dr #] [power # | dCP.dr #] [Fr # | dCFr.dr #] [rpm #] [inner.radius #] [outer.radius #]
Description	Defines a special boundary condition for multi-block-faces to simulate a rotor using the actuator disk model (see Section 5.9). The axis (mandatory input), should start from the center of the disk, and must point in the desired axial thrust direction (upstream). In case that dCT.dr, dCP.dr, dCFr.dr, or power, are used, the user must also set the rpm, and the inner & outer radii. The options thrust, power, and Fr accept a scalar value. The options dCT.dr, dCP.dr, dCFr.dr also accept functions of normalized radius. Note that the disk.model should be specified on two grids, one from each side of the disk.
Parameters	imin, jmin, kmin, imax, jmax, kmax — Location of rotor disk. $vertex1 — First vertex for axis. $vertex2 — Second vertex for axis. thrust # — Rotor disk total thrust. dCT.dr # — Thrust coefficient distribution along the dimensionless radius (local thrust coefficient). power # — Rotor disk power. dCP.dr # — Power coefficient distribution along the dimensionless radius (local power coefficient). Fr # — Radial force. dCFr.dr # — Radial force coefficient distribution along the dimensionless radius (local radial force coefficient). rpm # — RPM of the rotor disk. inner.radius # — Inner radius of the rotor disk. outer.radius # — Outer radius of the rotor disk.
Examples	-bc imax disk.model j=1,25 thrust=10000 outer.radius=0.5 inner.radius=0

Table 10.243: BC disk.model option

10.9 Movement Options

10.9.1 Motion Simulation

Reference.frame		
Syntax	-formation reference.frame $formation	
Description	Sets relative motion to different formation, otherwise motion is relative to observer (earth) frame recursive relative motion is not check and should be avoided.	
Parameters	$formation	Reference frame formation name.
Examples	-formation reference.frame payload-form	

Table 10.244: DOF reference.frame option

Time span options

Syntax	-dof [start | time | all | {span {start | #t0} {end | #t1}}] dof-option...]...
Description	Defines a 6DOF grid motion directive. See Table A.1 in Appendix A for a complete list of the options.
Parameters	start — Apply the requested options at the beginning of the simulation. time — Apply the requested options at a specific time in the simulation. all — Apply the requested options throughout the simulation. span — Apply the requested options over a span of time. The time span is defined using the following options: start — Simulation beginning. #t0 — Specific start time. end — Simulation end. #t1 — Specific end time.
Examples	-dof start origin.o 0.0 0.0 0.0 offset.p 0.8 0 0.16 attitude.p.deg 0 0 0 span 0.1 end force sf all force.o.explicit 0 0 -980 CG

Table 10.245: DOF time span options

Evaluation of any “dof” parameter may be evaluated with pre-defined functions: const, linear, sin, cos, tanh, or a table where:

- const is a constant
- linear #a #b signifies the linear equation $y = a + b * t$
- sin #a #b #c #d signifies the equation $y = a + b * \sin(c * t + d)$
- cos #a #b #c #d signifies the equation $y = a + b * \cos(c * t + d)$
- tanh #a #b #c #d signifies the equation $y = a + b * \tanh(c * t + d)$
- table {#t #y}... end.table signifies a table of inputs with a linear relation between discrete times (#t must increase monotonically).

The “dof” direction can be limited by space region using the directives “current.o” or “delta”. The directive “unlimit.pos” closes the limiting section. The syntax is given by:

{current.o | delta} {{x | y | z} {< | >} #}...

where:

- The directive “current.o” limits subsequent directions with the current (absolute) origin position.
- The directive “delta” limits subsequent directions with the delta (offset) position.
- The directive “unlimit.pos” removes the limits from subsequent directions.

The “dof” direction can also be limited by the limit/unlimit flag of any formation: [when.in.origin.limit | when.in.attitude.limit | when.origin.not.in.limit | when.attitude.not.in.limit] $formation when.in.origin.limit enables to following dof directive is the origin limit flag is raised when.origin.not.in.limit enables to following dof directive is the origin limit flag is not raised when.in.attitude.limit enables to following dof directive is the attitude limit flag is raised when.attitude.not.in.limit enables to following dof directive is the attitude limit flag is not raised unlimit unlimits sub-sequent directions.

The theory behind the motion simulation module is found in Chapter A. For a complete list of the "-dof" options, including the force options, the lock options, and the integration method options, the reader is referred to Table A.1 in Appendix A.

10.9.2 Aeroelasticity

As described in Chapter 8, using the computational aeroelasticity module requires a few steps. The first step is to map the structural modes from the structural model grid onto the aerodynamic model (*i.e.,* CFD mesh). This is conducted through the "-spline" directive (see Table 10.246).

The next step is to prescribe the frequency in which a static deformation is performed (for static aeroelasticity) or the integration method for the dynamic generalized equation of motion (Equation (8.6)). In the latter case, a deformation is conducted every time step and a time accurate, dual-time stepping integration is recommended (see Table 10.15).

As all aeroelastic simulations refer to "formations," a formation has to be declared (see Section 10.5.1). The following tables contain descriptions of all relevant directives and options.

<table>
<tr><th colspan="3">Spline</th></tr>
<tr><td>Syntax</td><td colspan="2">-spline $spline-name [p3d.structural] [p3d.deform #] [p3d.deform.structural.phi #...] [p3d.deform.aero.phi | modes] [rigid.tolerance #] [{xi | zeta} {#index #value}...] [surface-options]...</td></tr>
<tr><td>Description</td><td colspan="2">Sets up the spline for mode mapping.</td></tr>
<tr><td>Parameters</td><td>$spline-name</td><td>Spline name</td></tr>
<tr><td></td><td>p3d.structural</td><td>Output the structural grid.</td></tr>
<tr><td></td><td>p3d.deform</td><td>Output the aeroelastic modes followed by the ‘ξ’ value to be used for the output.</td></tr>
<tr><td></td><td>p3d.deform.structural.phi</td><td>Outputs the spline matrixes used.</td></tr>
<tr><td></td><td>p3d.deform.aero.phi</td><td>Outputs the spline matrixes used.</td></tr>
<tr><td></td><td>modes</td><td>Selects specific modes.</td></tr>
<tr><td></td><td>rigid.tolerance</td><td>Sets tolerance for rigid body mode eigenvalues.</td></tr>
<tr><td></td><td>xi | zeta</td><td>Sets a specific ξ / ζ value to be used for a specific mode.</td></tr>
<tr><td></td><td>#index</td><td>The index of ξ.</td></tr>
<tr><td></td><td>#value</td><td>The value of ξ.</td></tr>
<tr><td></td><td>surface-options</td><td>Sets the surface options, see Table 10.247.</td></tr>
<tr><td>Examples</td><td colspan="2">-spline phdp-spline</td></tr>
<tr><td></td><td colspan="2">p3d.structural</td></tr>
<tr><td></td><td colspan="2">p3d.deform 10</td></tr>
<tr><td></td><td colspan="2">wing kmin; i = 23,183; j = 1, 24</td></tr>
<tr><td></td><td colspan="2">grid grid_s_102.dat all</td></tr>
<tr><td></td><td colspan="2">mode mode_s_102.dat</td></tr>
<tr><td></td><td colspan="2">ips</td></tr>
</table>

Table 10.246: Aeroelasticity spline option

Surface Options

Syntax	{$zonename {imin | imax | jmin | jmax | kmin | kmax} [{i | j | k}=#min,#max]]... [force #i #j #k $name]... [force.o.explicit #i #j #k #fx #fy #fz]... {grid $nastran-grid-file {all | $points-file}} {mode $nastran-mode-file [#mode]...} [str-trans]... {ips | bs}}... [aero-trans]...	
Description	Transformation can be any combination of: offset, rotate.x.rad, rotate.y.rad, rotate.z.rad, rotate.x.deg, rotate.y.deg, rotate.z.deg They are defined similarly to the same option of formation, body, or zone transformations (see for example Tables 10.49 and Tables 10.50). When they appear before the first surface they refer to the structural grid. Otherwise they refer to the aerodynamic grid.	
Parameters	$zonename	The aerodynamic zone name.
	imin, imax	The surface edge.
	jmin, jmax	The surface edge.
	kmin, kmax	The surface edge.
	#min, #max	Index values.
	grid	Set the Nastran grid file ($nastran-grid-file).
	all	Use all grid points.
	point-file	File for selected grid points.
	mode	Set the Nastran mode file ($nastran-mode-file).
	#mode	Selected modes.
	str-trans	Set of structural grid transformations.
	ips / bs	Set an infinite plate spline or a beam spline.
	aero-trans	Set of aerodynamic grid transformations.
	force	Adds the aerodynamic force of an entity ($name) at an explicit surface grid-point (#i #j #k)
	force.o.explicit	Adds an explicit force (#fx #fy #fz) at an explicit surface grid-point (#i #j #k)
Examples	N/A	

Table 10.247: Aeroelasticity surface options

Static.deformation		
Syntax	-formation static.deformation {impermeable | noslip | slip} from # every # until #	
Description	Sets up static aeroelasticity. Defines the start, end, and frequency of deformation.	
Parameters	impermeable	Inviscid wall.
	noslip	No-slip wall.
	slip	Slip boundary conditions.
	from	Starting step.
	every	Deformation frequency.
	until	Ending step.
Examples	-formation static.deformation from 100 every 25 until 1000	

Table 10.248: Aeroelasticity static.deformation option

Dynamic.deformation

Syntax	-formation dynamic.deformation {impermeable | noslip | slip} [$integration-method]	
Description	Sets up the integration method for dynamic aeroelasticity.	
Parameters	impermeable	Inviscid wall.
	noslip	No-slip wall.
	slip	Slip boundary conditions.
	$integration-method	Integration method (see Table 10.250 for available integration methods).
Examples	-formation dynamic.deformation int.rk4	

Table 10.249: Aeroelasticity dynamic.deformation option

Dynamic Aeroelasticity Integration Methods

Directive	Integration method
int.rk3	Runge-Kutta 3rd order.
int.rk4	Runge-Kutta 4th order (default).
int.rk.merson	Runge-Kutta Merson (4th order).
int.ab.1	Adams-Bashforth 1st order (Explicit Euler).
int.ab.2	Adams-Bashforth 2nd order.
int.ab.3	Adams-Bashforth 3rd order.
int.ab.4	Adams-Bashforth 4th order.
int.ab.5	Adams-Bashforth 5th order.
int.am.1	Adams-Moulton 1st order.
int.am.2	Adams-Moulton 2nd order.
int.am.3	Adams-Moulton 3rd order.
int.am.4	Adams-Moulton 4th order.
int.am.5	Adams-Moulton 5th order.

Table 10.250: Dynamic aeroelasticity integration methods

10.10 Adaptive Mesh Refinement

The Adaptive Mesh Refinement (AMR) module automatically refines and coarsens certain segments of a mesh based on certain sensors. The refinement is conducted on a per zone basis. The refinement itself works as follows: The initial user grid is split into 8x8x8 cells called "patch"s. When a refinement criterion is met for at least one cell in a patch, the entire patch is refined to a higher level of 16x16x16 cells, and so forth. When a coarsening criterion is met by all the cells in a patch, the patch is coarsened to a lower level.

Adaptive mesh refinement (AMR) is controlled via the "-amr" directive as described in Table 10.251. The AMR refinement types may be chosen as trilinear or based on spline as described in Table 10.252. The sensors are described in Table 10.253 and the required reference values are described in Table 10.254. The output is controlled by adding options to the "-amr" directive as described in Table 10.255.

10.10.1 AMR Sensor Implementation

Let S be a sensor as described in Table 10.253 and let $ref.S$ be a reference value as described in Table 10.254. Therefore the sensor function takes the form:

$$F = \frac{S_{i+1} - 2 * S_i + S_{i-1}}{ref.S} \tag{10.4}$$

Consequently, the user controls the refinement using the selected sensor and the reference value.

AMR

Syntax

-amr {{levels | from | every | until} #} $refinement-type [allow.refinement.on.boundaries imin | imax | jmin | jmax | kmin | kmax] [cfl.automatic] [restrict.by $restrict-shape] $shape | refine # $sensor... [$reference #]...]...

Description

Parameters

levels	Number of levels the zone can be refined.
from, every, until #	Defines when to refine/coarsen the zone based on the iteration/step.
$refinement-type	Type of refinement. See Table 10.252 for refinement types.
allow.refinement..	Allow refinement on boundaries (applies only to zones whose dimensions are divisible by 8; default is no refinement on boundaries).
imin, imax	Refinement extents in the ξ direction.
jmin, jmax	Refinement extents in the η direction.
kmin, kmax	Refinement extents in the ζ direction.
cfl.automatic	Zonal CFL factor decreasing the CFL in a power of two for every refinement level.
$restrict-shape	Restrict all refinement and coarsening criteria to the designated $shape.
$sensor	Defines the sensor type. See Table 10.253 for sensor types.
$reference #	See Table 10.254.

Examples

```
-amr levels 1
        refine.bi.sectional
        allow.refinement.on.boundaries jmin
        from 20 every 5 until 2900
        refine level 1
               sensor.p ref.p 1000
        coarse level 1
               sensor.p ref.p 1000
```

Table 10.251: AMR option

AMR Refinement Types		
Syntax	$refinement-type [refine.bi.sectional | refine.spline | refine.spline.precalc | {refine.spline.fly #}]	
Description	Prescribe the AMR type.	
Parameters	refine.bi.sectional	Trilinear refinement.
	refine.spline	Calculates the spline on demand, slowest but economical in memory.
	refine.spline.precalc	Pre-calculates the spline, fastest but memory intensive.
	refine.spline.fly #	Calculates the spline with user defined buffers' sizes..
Examples	N/A	

Table 10.252: AMR refinement types

AMR Sensors		
Syntax	$sensor	
Description	Define the sensor for refinement.	
Parameters	sensor.p	Pressure sensor.
	sensor.te	Temperature sensor.
	sensor.r	Density sensor.
	sensor.u	x coordinate direction velocity component sensor.
	sensor.v	y coordinate direction velocity component sensor.
	sensor.w	z coordinate direction velocity component sensor.
	sensor.primitives	Sensor for all the primitives (p, r, u ,v, w).
	sensor.velocities	Sensor for all the velocities (u ,v, w).
Examples	N/A	

Table 10.253: AMR sensors

AMR Reference		
Syntax	$reference	
Description	Define the reference values.	
Parameters	ref.p	Pressure reference.
	ref.te	Temperature reference.
	ref.r	Density reference.
	ref.u	x coordinate direction velocity component reference.
	ref.v	y coordinate direction velocity component reference.
	ref.w	z coordinate direction velocity component reference.
Examples	N/A	

Table 10.254: AMR references

AMR p3d Options	
Option	Outcome
amr.none	No AMR output.
amr.orig	Output 'per refined 8x8x8 block' except original zones.
amr.only	Output the refined levels combined to a complete block.
amr.all	Output the refined levels as well as the original zones.

Table 10.255: AMR p3d options

10.11 Output Options

There are various types of output files that EZAir produces. A log file, a Plot3D file, and a Table. The number of the output files is unlimited and may be controlled through the input file. The following sections describe the file types and directives.

10.11.1 Log File

A log file may be generated by using the directive "-log". A description of the command is given in Table 10.256 in Appendix B. All the options for log file entries are described in Table B.1.

Log	
Syntax	-log $file {per.iter | per.step | per.run | per.cycle} options
Description	Defining the log file columns. Every option adds columns to the relevant log file. It has a scope: 'global', all.zones, all.bodies, all.formations, all.forces, $bodyname, $zonename, or $forcename. This is followed by '@' and the desired value. If no scope is explicitly stated, the current scope is used. The keyword 'default' sets the default columns. Any combination of the items listed in Table B.1 (may be found in Appendix B) can be used.
Parameters	$file — Log file name. per.iter — Print a log line every iteration. per.step — Print a log line every step (physical time step). per.run — Print a log line every iteration or step based on the simulation type. per.cycle — Print a log line every multigrid cycle.
Examples	-log global per.iter default max(mt) pos(max(mt)) -log forces per.run cl@airfoil cd@airfoil cm@airfoil

Table 10.256: Log options

10.11.2 Post-processing Options

Post-processing options include the output of Plot3D files and tabular files. A description of the command is given in Table 10.257 below. Each option in the input file results in a file (or files) having the content as described by the command (the option may span over multiple lines). All functions have the current default scope: global, body, or zone. The format of Plot3D files is the default format as described in Table 10.17. The format of a specific file can be overridden using the format parameter (see table below) and it can be any of the formats that can be set with the command "-sim format" (see Table 10.17). To explicitly add a different scope one should use the directive: function@scope. Vertical (horizontal) creates a tabular file for every 'do' iteration/step. In addition, horizontal appends the last function as the last row in the table file. A list of all available functions is given in Table C.1 (may be found in Appendix C).

Post Options	
Syntax	{-p3d | -table} $filename [cell \| node] [$zonename [{i | j | k} = #min [,#max]] [imin | imax | jmin | jmax | kmin | kmax]] [vertical | horizontal] [functions ...][format] [combined.fn | separated.fn] [in.solver | in.chimera] [{do | do.overwrite | do.solution | do.all} #start, [#end, [#step]]]
Description	Adds a Plot3D file or a table. See Table C.1 in Appendix C for a complete list of functions and a list of parameters and examples in the following pages. Read above for an additional description.

Parameters	-p3d / -table	Plot3D type output or a tabular output.
	cell/node	Cell center or cell vertex (default) output.
	$zonename	Zone name to output.
	i, j, k	Limit the range (default is entire grid).
	#min / #max	Zone limits.
	imin, imax	Table i direction limits.
	jmin, jmax	Table j direction limits.
	kmin, kmax	Table k direction limits.
	vertical	Vertical table.
	horizontal	Horizontal table (default).
	format	File format (see Table 10.17).
	combined.fn	Combine all the functions into a single file.
	separated.fn	Separate function files (default).
	in.solver	Use computational cells only to reconstruct node values.
	in.chimera	use computational cells + chimera cells (default)) to reconstruct node values.
	do	Write the grid file (once, time signature); Overwrite other output functions (steady state). Write the grid file with a time signature only in case it moves and write other functions with a time signature (time accurate).
	do.overwrite	Overwrite grid file and other output functions (no time signature).
	do.solution	Overwrite grid file and write other output functions with a time signature.
	do.all	Force printing a time signature (or step number in case of a steady state simulation).
	start	Initialize the output loop (may use a specific step number instead).
	step	Output loop interval.
	end	End the output loop (may use a specific step number instead).

Examples	-p3d nlf0414 node global xyzb q mt do 2000, end, 500
	-table cp node vertical
	airfoil jmin
	x xsi cp do start, end, 50
	-p3d nlf0414 node global xyzb q mt unf r8 do 2000, end, 500

Table 10.257: Post-processing options

10.12 Parallelization Options

Parallelization options are used to control the placement of a problem on the actual boards of the machine(s). See Chapter 9 for a detailed description.

Board		
Syntax	-parallel [#boards] [dualquad | quad | single | quadsixcore | dual-sixcore | #threads]	
Description	Adds shared memory boards of single processor, quad cores, dual-quad cores, and etc' The order in which they are added matters and should confirm to the h_opt_paraty environment used by the relevant OpenMP library.	
Parameters	#boards	Number of boards to be added. Not entering a values implies 1 board. Boards is the number of "machines" in case of MPI, or boards in case of shared memory.
	#threads	Number of threads to be added.
Examples	ezair -f ezfile_naca-23012 -parallel 2 8	

Table 10.258: Parallelization board option

10.13 Run-time Options

Create an ascii text file named 'ezair.ins' containing one of the following:

Run-time Options

Option	Action
stop	Stop and save a restart.
kill	Immediately stop the run (even in dual-time mode) and save a restart.
cfl #	Change to the specified constant CFL.
cfl.tu #	Change to the specified constant CFL for the turbulence solver.
dt #	Change to the specified constant time step.
iters #	Change to the specified number of iterations.
steps #	Change to the specified number of steps.
p3d	Dump all p3d files requested by the user at the next available opportunity.

Table 10.259: Run-time options

Chapter 11

User Defined "N" Degrees of Freedom Motion

The EZAir suite of flow solvers has a six degrees of freedom motion simulation module that is embedded within the solvers. The capability can be extended even further using input commands. Recently, a need for an extended capability has surfaced. This need may arise when there is a relationship between different rigid bodies, or 'formations' using EZAir terms. Or, when the movement is dictated by a control system, or when the movement is too complex to be described by EZAir input capabilities. Such unlimited capability can be obtained by introducing a comprehensive, programmatically controlled on top of the movement module (DOF), capability into EZAir. This report describes the newly introduced module, codename 'NDOF', that was developed and implemented for such cases. The module enables the user to dynamically link at run-time any code that describes any general motion to EZAir as long as it supports the detailed interface.

11.1 Library Interface

The following chapter discusses the requirements and interfaces that the library must uphold in order to correctly link to EZAir. For the sake of clarity, ***EZAir input file options and keywords*** appear in different style than the C source code keywords of the user NDOF library. The following example assumes that the user is writing a library called ‘MyLibrary’. Following, in Listing 11.1, there is an example from an EZAir input file showing how to load the NDOF library and hook EZAir to the library principle callbacks.

```
-ndof MyLibrary-lib ./path/libMyLibrary.so
    init MyLibrary_init MyLibrary.in
    integrate MyLibrary_integrate
    end MyLibrary_end
    save MyLibrary_save
    load MyLibrary_load
    sample MyLibrary_sample
```

Listing 11.1: EZAir Input File, DOF Interface

The user must supply a relative path to the users’ dynamically loaded NDOF library: ‘./path/libMyLibrary.so’. Three more callback functions must be provided:

1. Initialization callback (MyLibrary_init) where it takes also a string to pass to the callback (‘MyLibrary.in’). The Interface definition is shown in Listing 11.2.

```
char MyLibrary_init(const char* filename);
```

Listing 11.2: Requirement - Initialization Interface

2. Integration callback (MyLibrary_integrate), where the integration interval (tInitial, tFinal) is passed to the library by EZAir. The Interface definition is shown in Listing 11.3.

```
char MyLibrary_integrate(const double tInitial, const double tFinal);
```

Listing 11.3: Requirement - Integration Interface

3. End callback (MyLibrary_end), allowing the library to deallocate memory, close log files, and etc’. The Interface definition is shown in Listing 11.4

```
char MyLibrary_end();
```

Listing 11.4: Requirement - End Interface

Note, the return value of these functions should always be NDOF_SUCCESS, or NDOF_FAIL which are two constants defined by the NDOF module and not user scalars.

The user can add two more callbacks to enable 'restart' while using the library. The callbacks are for saving (MyLibrary_save) and loading (MyLibrary_load). These callbacks receive in their argument list the filename in which to save or load the 'restart' information from or to. The Interface definition is shown in Listing 11.5.

```
char MyLibrary_save(const char* filename);
char MyLibrary_load(const char* filename);
```

Listing 11.5: Requirement - Save/Load Interface

The most recent addition to the NDOF library is enabling the user to sample the CFD solution by an array of Cartesian coordinates. The user can supply the sample function (MyLibrary_sample). The Interface definition is shown in Listing 11.6.

```
char MyLibrary_sample(
  long* n, double* x[], double* y[], double* z[],
  long* stat[],
  double* u[], double* v[], double* w[],
  double* p[], double* r[], double* T[]);
```

Listing 11.6: Requirement - Vertex Interface

Where the user passes the number of vertexes, n, needed for sampling, and the x, y, and z arrays that are the Cartesian coordinates arrays. The primitive flow data, velocity components, pressure, density, and temperature are then stored in the u, v, w, p, r, and T Arrays. The user also provides an array to store status data per point where one may check if the sampling per that vertex has been found correctly. If the stat of a vertex equals LONG_MAX (defined in limits.h) it means that the vertex was not found, and in such a case the primitives shall contain the 'freestream' values. Note, this function is called by EZAir NDOF module prior to calling the integration function. The user can assume that the arrays holding the flow primitive data are set once the vertex function returned.

Passing aerodynamic forces to the NDOF library is done by indication per 'force' to report the values. Listing 11.7 shows the input file, and Listing 11.8 shows its library counterpart.

```
-force airfoil-f
    noslip airfoil   jmin
    refpoint cg
  MyLibrary-lib MyLibrary_from_ezair
```

Listing 11.7: EZAir Input File, DOF Interface

Note, the user can supply any reference point about which to calculate the moments, however, it would be easier for the user if that reference point is the center of motion of the rigid body.

```
char MyLibrary_from_ezair(
   const double WetSurface,
   const double Fx, const double Fy, const double Fz,
   const double Mx, const double My, const double Mz,
   const double x, const double y, const double z);
```

Listing 11.8: Aerodynamic Forces From EZAir to the Library

The argument 'WetSurface', appearing in Listing 11.8, is the sum of all the surfaces over which the force is integrated. (Fx, Fy, Fz) is the force vector, (Mx, My, Mz) is the moment vector, and (x, y, z) is the reference point for the moment calculation (cg in the Listing 11.7). The Forces and Moments are calculated in an inertial system (stationary observer frame).

Last hook is to pass the movement information back to EZAir. Within the EZAir input file, the user hooks both center of motion vertex and movement as shown in Listing 11.9

```
-vertex cg 0 0 0 after.trans
-dof   all MyLibrary-lib MyLibrary_to_ezair cg
```

Listing 11.9: EZAir Input File, DOF Interface

The 'after.trans' keyword in the vertex option indicates that the vertex will always contain the data that is passed back from the library with the '-dof' option without any EZAir transformations. The library hook is shown in Listing 11.10

```
char MyLibrary_to_ezair(
   double* xo, double* yo, double* zo,
   double * qow, double* qox, double* qoy, double* qoz,
   double* m,
   double* Ixx, double* Ixy, double* Ixz,
   double* Iyy, double* Iyz, double* Izz,
   double* ax, double* ay, double* az,
   double* u, double* v, double* w,
```

```
double* x, double* y, double* z,
double* pdot, double* qdot, double* rdot,
double* p, double* q, double* r,
double* qw, double* qx, double* qy, double* qz);
```

Listing 11.10: Movement FromLibrary to EZAir

Where (xo, yo, zo), (wow, qox, qoy, qoz) are the initial center of motion and attitude quaternion thereby enabling changing center of motion during the simulation. The inertial coefficients are mass (m) and body frame inertia tensor (Ixx, Ixy, Ixz, Iyy, Iyz, Izz). The translational parameters in a stationary observer frame are: (ax, ay, az) acceleration, (u, v, w) velocity, (x, y, z) position. The Angular parameters in body frame are: (pdot, qdot, rdot) angular acceleration, (p, q, r) angular velocity, and (qw, qx, qy, qz) quaternion.

11.2 Automatic Tool

In this chapter a step by step instruction on how to convert the automatic C code into an NDOF library is given based on the HingedAirfoils Sample.

11.2.1 Global Variables

1. Include 'ndoflib.c' After the initial global variable definition.
2. Move the variables: 'integStp', 'absError', 'relError', 'printIntScreenAsDouble', 'printIntFileAsDouble' from 'main' function scope to the global scope.
3. Move the variable 'hc' from 'IntegratorStep' scope to the global scope.
4. Add global variables to describe the initial center of motion and attitude.

The code before the 'main' function should look as in Listing 11.11

```
#include"ndoflib.c"
double integStp, absError, relError;
double printIntScreenAsDouble, printIntFileAsDouble;
```

```
double hc = 0.0;
origin_t Ao, Bo;
```

Listing 11.11: Library Global Variable, HingedAirfoils Sample

11.2.2 Split ‘main’ Function

Split ‘main’ function into three parts and rename them ‘init’, ‘integrate’, and ‘end’ and change the functions header to match the library interface as shown in Listings 11.12, 11.13, and 11.14.

In particular, the steps needed for the ‘init’ function are:

1. Change the value of the inputFileName to the parameter of the function.
2. Add NDOF_SUCCESS return value.
3. Add evaluation of the initial position.

```
char HingedAirfoils_init(const char* filename)
{
   const char* inputFileName = filename; /* "HingedAirfoils4.in"; */
   FILE* inputFilePtr = fopen( inputFileName, "r" );
   double tInitial, tFinal;
   int   iloop;

   /* Ensure input file can be opened and read comment lines */
   if( !inputFilePtr ) { printf("Error: unable to open file %s\n",
      inputFileName); return 1; }
   for( iloop = 0; iloop < 4; iloop++ ) ReadFormattedString( inputFilePtr,
      NULL );

   /* Read values of constants from input file */
   ReadFormattedNumbers( inputFilePtr, &cA, &g, &hinge_dx, &IzzA, &IzzB,
      &kA, &kB, &mA, &mB, &xcgA, &xcgB, NULL );

   /* Read the initial value of each variable from input file */
```

```
ReadFormattedNumbers( inputFilePtr, &thetaA, &thetaB, &thetaAp,
    &thetaBp, NULL );

/* Read integration parameters from input file */
ReadFormattedNumbers( inputFilePtr, &tInitial, &tFinal, &integStp,
    &printIntScreenAsDouble, &printIntFileAsDouble, &absError,
    &relError, NULL );

/* Close input file */
fclose( inputFilePtr );

/* Unit conversions */
Pi       = 3.141592653589793;
DEGtoRAD = Pi / 180.0;
RADtoDEG = 180.0 / Pi;

SetArrayFromVariables( varArrayToIntegrate );

{
    CalculateOutput(0, Output);
    Ao.x = xA;
    Ao.y = yA;
    Ao.z = 0;
    Bo.x = xB;
    Bo.y = yB;
    Bo.z = 0;
    convert_rotation_to_quaternion(&Output[46], &Ao.qw, &Ao.qx, &Ao.qy,
        &Ao.qz, invert);
    convert_rotation_to_quaternion(&Output[63], &Bo.qw, &Bo.qx, &Bo.qy,
        &Bo.qz, invert);
}

return NDOF_SUCCESS;
```

```
}
```

Listing 11.12: Library Init Function, HingedAirfoils Sample

For the 'integrate' function add both return values NDOF_FAIL and NDOF_-SUCCESS.

```
char HingedAirfoils_integrate(const double tInitial, const double tFinal)
{
   if( !IntegrateForwardOrBackward( _NumberOfODES_, varArrayToIntegrate,
      Output, tInitial, tFinal, integStp, absError, relError, (unsigned
      int)printIntScreenAsDouble, (unsigned int)printIntFileAsDouble ) )
      return NDOF_FAIL;
   return NDOF_SUCCESS;
}
```

Listing 11.13: Library Integrate Function, HingedAirfoils Sample

```
char HingedAirfoils_end()
{
   /* Close output files. Inform user of input and output filename(s). */
   OutputToScreenOrFile( NULL, 0, 0, (unsigned int)printIntScreenAsDouble,
      0 );
   return NDOF_SUCCESS;
}
```

Listing 11.14: Library End Function, HingedAirfoils Sample

11.2.3 Add Save and Load Functions

These functions should save and load all the global variables as shown in Listing 11.15 and Listing 11.16.

```
char HingedAirfoils_save(const char* filename)
{
   FILE* fp = fopen(filename, "wb");

   while(fp)
   {
```

```
    if(1 != fwrite(&cA, sizeof(double), 1, fp))
        break;

    if(1 != fwrite(&g, sizeof(double), 1, fp))
        break;

    /* ... ALL GLOBL VARIABLES SIMILARLY ... */

    if(_NumberOfODES_ != fwrite(varArrayToIntegrate, sizeof(double),
        _NumberOfODES_, fp))
        break;

    if(79 != fwrite(Output, sizeof(double), 79, fp))
        break;

    if(1 != fwrite(&numberOfCallsToEqns1, sizeof(unsigned long), 1, fp))
        break;

    /* added to library */
    if(1 != fwrite(&Ao, sizeof(origin_t), 1, fp))
        break;

    if(1 != fwrite(&Bo, sizeof(origin_t), 1, fp))
        break;

    /* from main */
    if(1 != fwrite(&integStp, sizeof(double), 1, fp))
        break;

    if(1 != fwrite(&absError, sizeof(double), 1, fp))
        break;

    if(1 != fwrite(&printIntScreenAsDouble, sizeof(double), 1, fp))
```

```
            break;

        if(1 != fwrite(&printIntFileAsDouble, sizeof(double), 1, fp))
            break;

        /* from IntegratorStep */
        if(1 != fwrite(&hc, sizeof(double), 1, fp))
            break;

        fclose(fp);
        fp = NULL;
        return NDOF_SUCCESS;
    }

    if(fp)
    {
        fclose(fp);
        fp = NULL;
    }

    return NDOF_FAIL;
}
```

Listing 11.15: Library Save, HingedAirfoils Sample

```
char HingedAirfoils_load(const char* filename)
{
    FILE* fp = fopen(filename, "rb");

    while(fp)
    {
        if(1 != fread(&cA, sizeof(double), 1, fp))
            break;
```

```
if(1 != fread(&g, sizeof(double), 1, fp))
    break;

/* ... ALL GLOBL VARIABLES SIMILARLY ... */

if(_NumberOfODES_ != fread(varArrayToIntegrate, sizeof(double),
    _NumberOfODES_, fp))
    break;

if(79 != fread(Output, sizeof(double), 79, fp))
    break;

if(1 != fread(&numberOfCallsToEqns1, sizeof(unsigned long), 1, fp))
    break;

/* added to library */
if(1 != fread(&Ao, sizeof(origin_t), 1, fp))
    break;

if(1 != fread(&Bo, sizeof(origin_t), 1, fp))
    break;

/* from main */
if(1 != fread(&integStp, sizeof(double), 1, fp))
    break;

if(1 != fread(&absError, sizeof(double), 1, fp))
    break;

if(1 != fread(&printIntScreenAsDouble, sizeof(double), 1, fp))
    break;
```

```
        if(1 != fread(&printIntFileAsDouble, sizeof(double), 1, fp))
            break;

        /* from IntegratorStep */
        if(1 != fread(&hc, sizeof(double), 1, fp))
            break;

        fclose(fp);
        fp = NULL;
        return NDOF_SUCCESS;
    }

    if(fp)
    {
        fclose(fp);
        fp = NULL;
    }

    return NDOF_FAIL;
}
```

Listing 11.16: Library Load

11.2.4 Add 'From' Hooks

Every aerodynamic force that reports back to the NDOF library should implement a specific function as follows in Listing 11.17

```
char HingedAirfoils_A_from_ezair(
    const double WetSurface,
    const double Fx, const double Fy, const double Fz,
    const double Mx, const double My, const double Mz,
    const double x, const double y, const double z)
{
```

```
    FyA = Fy;

    /* assuming (x,y,z) is moving with center mass, otherwise steiner rule
        for calculating Moments */
    MzA = Mz;

    return NDOF_SUCCESS;
}
```

Listing 11.17: EZAir Aerodynamic Forces To HingedAirfoils Library

Note, the sample case is two dimensional and just F_y and M_z are needed. The Moments are evaluated in a stationary observer frame (and **not** in the body frame) about (x, y, z) vertex. The vertex can be the center of motion if the user sets the reference point of the aerodynamic force to the vertex NDOF placeholder within the '-dof' option. The user can rotate the moments to the body frame in particular with pre-supplied functions.

11.2.5 Add 'To' Hooks

Every formation that is moved using the NDOF library passes the entire movement information back to EZAir using a similar function as shown in Listing 11.18.

```
char HingedAirfoils_A_to_ezair(
     /* center of motion in initial grid */
    double* _xo, double* _yo, double* _zo,
     /* attitude in initial grid */
    double * _qow, double* _qox, double* _qoy, double* _qoz,
    /* Mass */
    double* m,
    /* Inertia Tensor */
    double* _Ixx, double* Ixy, double* Ixz,
    double* _Iyy, double* Iyz, double* _Izz,
    /* translational Acceleration in Observer Frame */
```

```
    double* ax, double* ay, double* az,
    /* translational Velocity in Observer Frame */
    double* u, double* v, double* w,
    /* translational Offset in Observer Frame */
    double* x, double* y, double* z,
    /* angular Acceleration in Body Frame */
    double* pdot, double* qdot, double* rdot,
    /* angular Velocity in Body Frame */
    double* p, double* q, double* r,
    /* angular State Quaternion */
    double* qw, double* qx, double* qy, double* qz)
{
    /* initial state condition of center mass and attitude, in the original
       grid state */
    *_xo = xoA;
    *_yo = yoA;
    *_zo = 0;

    *_qow = qoAw;
    *_qox = qoAx;
    *_qoy = qoAy;
    *_qoz = qoAz;

    /* current state */
    *_m = mA;
    *_Ixx = 1;
    *_Iyy = 1;
    *_Izz = IzzA;
    Ixy = Ixz = Iyz = 0;

    *_x = xA;
    *_y = yA;
    *_z = 0;
```

```
   *u = VxA;
   *v = VyA;
   *w = 0;

   *ax = AxA;
   *ay = AyA;
   *az = 0;

   convert_rotation_to_quaternion(&Output[46], _qw, _qx, _qy, _qz, invert);

   *_p = 0;
   *_q = 0;
   *_r = thetaAp;

   *_pdot = 0;
   *_qdot = 0;
   *_rdot = thetaApp;

   return NDOF_SUCCESS;
}
```

Listing 11.18: HingedAirfoils Library Movement Parameter to EZAir

Note, some of the function parameters prefix is underscore. This is simply to distinguish between the local function arguments and the library global variables and not a special requirement. Also note that the initial position and initial attitude is always passed.

11.2.6 Sample Flow Primitives

The user may be interested in the flow primitives (u, v, w, p, ρ, T) in specific vertexes. To use that capability the user needs to allocate memory either on the heap, but preferably on the stack as shown in Listing 11.19 here for two vertexes:

```
#define _NumberOfVertexes 2
double vertex_x[_NumberOfVertexes];
double vertex_y[_NumberOfVertexes];
double vertex_z[_NumberOfVertexes];
double vertex_u[_NumberOfVertexes];
double vertex_v[_NumberOfVertexes];
double vertex_w[_NumberOfVertexes];
double vertex_p[_NumberOfVertexes];
double vertex_r[_NumberOfVertexes];
double vertex_T[_NumberOfVertexes];
long vertex_stat[_NumberOfVertexes];
```

Listing 11.19: HingedAirfoils Library Vertex Allocation

Note, the user also allocates memory for 'status' information. A single callback should be implemented as shown in Listing 11.20:

```
char HingedAirfoils_sample(
   long* n, double* x[], double* y[], double* z[],
   long* stat[],
   double* u[], double* v[], double* w[],
   double* p[], double* r[], double* T[])
{
   *n = _NumberOfVertexes;
   *x = vertex_x;
   *y = vertex_y;
   *z = vertex_z;
   *u = vertex_u;
   *v = vertex_v;
   *w = vertex_w;
   *p = vertex_p;
   *r = vertex_r;
   *T = vertex_T;
   *stat = vertex_stat;
```

```
   vertex_x[0] = -2; vertex_y[0] = 0; vertex_z[0] = 0;
   vertex_x[1] = -1; vertex_y[1] = 0; vertex_z[1] = 0;

   return NDOF_SUCCESS;
}
```

Listing 11.20: HingedAirfoils Library Vertex Callback

The user can assume that once the function returned the information in the allocated arrays is filled with the primitives. In particular, when the 'integrate' callback is called, the user can retrieve the sampled data from the arrays back into his own variables. The interpolation is carried from the grid whose cell volumes is the smallest. In case that an interpolation could not be found, the values of the primitives would contain 'free stream' values and the status array would contain LONG_MAX.

11.3 Compilation of "N" DOF Library

A single file compilation to a shared library with the Intel icc compiler is done similar to Listing 11.21.

```
icc MyLibrary.c -o libMyLibrary.so -shared -fPIC
```

Listing 11.21: Compiling The Library

Appendix A

DOF Options

DOF Options

Option	Description
origin.o {$vertex | #x #y #z}	Set body frame origin, explicit coordinates are moved similary to named vertex
offset.p #x #y #z	Set offset from origin, in parent frame
mass #m	Set mass, used for 6dof motion eq.
inertia.b #I11 #I12 #I13 #I22 #I23 #I33	Set inertia tensor, in body frame. used for 6dof motion eq.
uvw.p #u #v #w	Set velocity of body origin in parent coordinates
uvw.dot.p #udot #vdot #wdot	Set acceleration of body origin in parent coordinates
attitude.p.rad | attitude.p.deg #phi #tet #psi	Set attitude, in parent frame coordinates
pqr.b.rad | pqr.b.deg #p #q #r	Set angular velocity in body frame coordinates

pqr.dot.b.rad | pqr.dot.b.deg #pdot #qdot #rdot	Set angular acceleration in body frame coordinates
force $force	Adds force (and moment) to motion eq. from predefined force.
force.ex $force #cfx #cfy #cfz #cmx #cmy #cmx	Adds force and moment to the motion equations from the predefined $force. [cfx, cfy, cfz] are the added fractions of the forces in reference frame. [cmx, cmy, cmz] are the added fractions of the moment in body frame (*e.g.*, $force-fx = $force-fx + #cfx * $force-fx)
force.o.explicit #fx #fy #fz {$vertex | {#x #y #z}}	Adds force in observer coordinates to motion eq. explicitly
force.b.explicit #fx #fy #fz {$vertex | {#x #y #z}}	Adds force in body frame coordinates to motion eq. explicitly
moment $force	Adds moment to motion eq. from predefined force.
moment.o.explicit #mx #my #mz	Adds pure moment in observer coordinates to motion eq. explicitly
moment.b.explicit #m1 #m2 #m3	Adds pure moment in body frame coordinates to motion eq. explicitly
center.o {$vertex | #x #y #z}	Set the center of gravity for 'weight' directive in observer frame
harness.o $vertex	Harness the origin to a vertex
origin.limit.o $vertex1 $vertex2	Limit the origin to move between the box that the vertexes form, sets the limit/unlimit state

[concentric.angular.limit.b.rad | concentric.angular.limit.b.deg | concentric.angular.limit.no.flag.b.rad | concentric.angular.limit.no.flag.b.deg] {#phi #theta #psi}_ref #deg	Limits the angular state in reference to the parent body frame, sets the limit/unlimit state (the 'no.flag' options do not raise the atitude limit flag)
force.weight.o $formation #gx #gy #gz	Adds weight force of a formation, acting at it's center
force.aerodynamic $formation	Adds the aerodynamic forces of a formation
dof.force $force-name	Adds a dof force summation to the force-name
force.aerodynamic.ex $formation #cfx #cfy #cfz #cmx #cmy #cmx	Adds the aerodynamic forces of a formation, while manipulating the aerodynamic force: [cfx, cfy, cfz] multiplies the force in reference frame. [cmx, cmy, cmz] multiplies the moment in body frame.
dof.force.ex $force-name #cfx #cfy #cfz #cmx #cmy #cmx	Adds the dof forces of a force-name, while manipulating the aerodynamic force: [cfx, cfy, cfz] multiplies the force in reference frame. [cmx, cmy, cmz] multiplies the moment in body frame.
mass.formation $formation	Adds the mass of a formation, should succeed mass
inertia.formation | inertia.formation.cg | inertia.formation.cm $formation	Adds the inertia tensor of a formation inertia.formation (should succeed inertia.b); inertia.formation.cg assumes the inertia is given in the gravity center; inertia.formation.cm assumes the inertia is given in the motion center.

wire #force-size #length-min #length-max $vertex1 $vertex2	Adds wire force acting at vertex1 in the direction of vertex2
torsion.spring.b.rad | torsion.spring.b.deg {#m1 #m2 #m3} {#a1 #a2 #a3}	Adds pure momentum (torsion spring based) in body frame coordinates to motion equation explicitly
moment.ea.b.rad | moment.ea.b.deg | moment.pqr.b.rad | moment.pqr.b.deg | moment.pqr.dot.b.rad | moment.pqr.dot.b.deg {#m1 #m2 #m3} {#d1 #d2 #d3} {#phi-ref #tet-ref #psi-ref} $formation	Adds pure momentum in body frame coordinates to the motion equation, explicitly based on Euler angles, angular velocity, or angular acceleration of a specific formation M = m1 * (phi - phi-ref) + m2 * (tet - tet-ref) + m3 * (psi - psi-ref) M1 = M * d1 M2 = M * d2 M3 = M * d3
force.spring.b {$vertex} {$vertex_reference} #a #b #c #d #e #f #g #h #i #j	Adds force in body frame coordinates to motion eq. explicitly. Where x, y, x are the coordinates of the reference vertex in body frame. Where u, v, w are the velocity of the reference vertex in body frame. The first vertex is the location where the force is acting. F = a + bx + cy + dz + eu + fv + gw F1 = F * h F2 = F * i F3 = F * j

force.spring.o {$vertex} {$vertex_reference} #a #b #c #d #e #f #g #h #i #j	Adds force in observer frame coordinates to motion eq. explicitly. Where x, y, x are the coordinates of the reference vertex in observer frame. Where u, v, w are the velocity of the reference vertex in observer frame. The first vertex is the location where the force is acting. $F = a + bx + cy + dz + eu + fv + gw$ $F1 = F * h$ $F2 = F * i$ $F3 = F * j$
lock.uvw.dot.p | lock.u.dot.p | lock.v.dot.p | lock.w.dot.p	Lock origin acceleration, relative to parent frame coordinates
[concentric.torsion.b.deg | concentric.torsion.b.rad] {#phi #tet #psi}_ref #k	Adds pure momentum based on the angle difference between the current frame attitude and the Euler angle reference acting towards the reference angles; k is the ‘spring‘ coefficient
unlock.uvw.dot.p | unlock.u.dot.p | unlock.v.dot.p | unlock.w.dot.p	Unlock origin acceleration
lock.uvw.p | lock.u.p | lock.v.p | lock.w.p	Lock origin velocity, implies zero acceleration
unlock.uvw.p | unlock.u.p | unlock.v.p | unlock.w.p	Unlock origin velocity
lock.pqr.dot.b | lock.p.dot.b | lock.q.dot.b | lock.r.dot.b	Lock angular acceleration, in body frame coordinates
unlock.pqr.dot.b | unlock.p.dot.b | unlock.q.dot.b | unlock.r.dot.b	Unlock angular acceleration

lock.pqr.b | lock.p.b | lock.q.b | lock.r.b	Lock angular velocity, implies zero angular acceleration, relative to body frame coordinates
unlock.pqr.b | unlock.p.b | unlock.q.b | unlock.r.b	Unlock angular velocity
org.a.am.1st | org.a.am.2nd | org.a.am.3rd | org.a.am.4th | org.a.am.5th | org.a.ab.1st | org.a.ab.2nd | org.a.ab.3rd | org.a.ab.4th | org.a.ab.5th	Integration method for translational acceleration into translational velocity.
org.v.am.1st | org.v.am.2nd | org.v.am.3rd | org.v.am.4th | org.v.am.5th | org.v.ab.1st | org.v.ab.2nd | org.v.ab.3rd | org.v.ab.4th | org.v.ab.5th	Integration method for translational velocity into translational displacement.
atd.a.am.1st | atd.a.am.2nd | atd.a.am.3rd | atd.a.am.4th | atd.a.am.5th | atd.a.ab.1st | atd.a.ab.2nd | atd.a.ab.3rd | atd.a.ab.4th | atd.a.ab.5th	Integration method for angular acceleration into angular velocity.
atd.v.am.1st | atd.v.am.2nd | atd.v.am.3rd | atd.v.am.4th | atd.v.am.5th | atd.v.ab.1st | atd.v.ab.2nd | atd.v.ab.3rd | atd.v.ab.4th | atd.v.ab.5th	Integration method for angular velocity into angular displacement.
int.am.1st | int.am.2nd | int.am.3rd | int.am.4th | int.am.5th | int.ab.1st | int.ab.2nd | int.ab.3rd | int.ab.4th | int.ab.5th	Integration method for all integrations.

evl.atd.a.2nd | evl.atd.a.3rd |		
evl.atd.a.4th		

Table A.1: DOF options

Appendix B

Log Options

Log Options

Option	Description
ANG.MOM	Angular momentum for cylindrical axes; available for: post;
B.ATD.A.DEG	DOF body frame angular acceleration in body frame [deg/s^2]; available for: formation; post;
B.ATD.A.RAD	DOF body frame angular acceleration in body frame [rad/s^2]; available for: formation; post;
B.ATD.V.DEG	DOF body frame angular velocity in body frame [deg/s]; available for: formation; post;
B.ATD.V.RAD	DOF body frame angular velocity in body frame [rad/s]; available for: formation; post;
B.CF	DOF Force sum coefficient, in body frame; available for: formation;
B.CM	DOF Moment sum coefficient, in body frame; available for: formation;

B.EA.DEG	DOF Euler angles in body frame [deg], calculated as inverse of O.EA.DEG; available for: formation; post;
B.EA.RAD	DOF Euler angles in body frame [rad], calculated as inverse of O.EA.RAD; available for: formation; post;
B.F	DOF Force Sum, in body frame [N]; available for: formation;
B.IXX	Inertia Tensor Ixx component in body frame; available for: formation; post;
B.IXY	Inertia Tensor Ixy component in body frame; available for: formation; post;
B.IXZ	Inertia Tensor Ixz component in body frame; available for: formation; post;
B.IYY	Inertia Tensor Iyy component in body frame; available for: formation; post;
B.IYZ	Inertia Tensor Iyz component in body frame; available for: formation; post;
B.IZZ	Inertia Tensor Izz component in body frame; available for: formation; post;
B.M	DOF Moment sum, in body frame [Nxm]; available for: formation;
B.ORG.A	DOF body frame translation acceleration, in body frame [m/s^2]; available for: formation; post;
B.ORG.V	DOF body frame translation velocity, in body frame [m/s]; available for: formation; post;
B.QUATERNION	DOF quaternion in body frame; available for: formation; post;
CD	Drag coefficient; available for: global; formation; body; zone; force; post;

CF	Force coefficients; available for: global; formation; body; zone; force; post;
CFF	Forces that are induced by friction coefficients; available for: global; formation; body; zone; force; post;
CFL	Courant-Friedrichs-Lewy number for the MF/NS Eq.; available for: global; formation; body; zone; force; post;
CFLT	Courant-Friedrichs-Lewy number for the turbulence model Eq.; available for: global; formation; body; zone; force; post;
CFP	Forces that are induced by pressure coefficients; available for: global; formation; body; zone; force; post;
CHK(CD)	Surface normalized cd pressure integration test; available for: global; formation; body; zone; force; post;
CHK(CFX)	Surface normalized cfx pressure integration test; available for: global; formation; body; zone; force; post;
CHK(CFY)	Surface normalized cfy pressure integration test; available for: global; formation; body; zone; force; post;
CHK(CFZ)	Surface normalized cfz pressure integration test; available for: global; formation; body; zone; force; post;
CHK(CL)	Surface normalized cl pressure integration test; available for: global; formation; body; zone; force; post;
CHK(CMX)	Surface normalized cmx pressure integration test; available for: global; formation; body; zone; force; post;
CHK(CMY)	Surface normalized cmy pressure integration test; available for: global; formation; body; zone; force; post;
CHK(CMZ)	Surface normalized cmz pressure integration test; available for: global; formation; body; zone; force; post;

CHK(CS)	Surface normalized cs pressure integration test; available for: global; formation; body; zone; force; post;
CHK(D)	Surface normalized drag pressure integration test; available for: global; formation; body; zone; force; post;
CHK(L)	Surface normalized lift pressure integration test; available for: global; formation; body; zone; force; post;
CHK(S)	Surface normalized side-force pressure integration test; available for: global; formation; body; zone; force; post;
CL	Lift coefficient; available for: global; formation; body; zone; force; post;
CM	Moment coefficient; available for: global; formation; body; zone; force; post;
CMF	Moments that are induced by friction coefficient; available for: global; formation; body; zone; force; post;
CMP	Moments that are induced by pressurr coefficient; available for: global; formation; body; zone; force; post;
CS	Side Force coefficient; available for: global; formation; body; zone; force; post;
CUT(DET)	Number of cutoffs from determinant; available for: global; formation; body; zone;
CUT(DNU)	Number of cutoffs for the dnu (spalart) turbulence variable; available for: global; formation; body; zone;
CUT(GM)	Number of cutoffs to gamma intermittency factor; available for: global; formation; body; zone;
CUT(K)	Number of cutoffs for the k turbulence variable; available for: global; formation; body; zone;

CUT(O)	Number of cutoffs for the omega turbulence variable; available for: global; formation; body; zone;
CUT(RT)	Number of cutoffs to Re_theta; available for: global; formation; body; zone;
CUT(R11)	Number of cutoffs to r11; available for: global; formation; body; zone;
CUT(R12)	Number of cutoffs to r12; available for: global; formation; body; zone;
CUT(R13)	Number of cutoffs to r13; available for: global; formation; body; zone;
CUT(R22)	Number of cutoffs to r22; available for: global; formation; body; zone;
CUT(R23)	Number of cutoffs to r23; available for: global; formation; body; zone;
CUT(R33)	Number of cutoffs to r33; available for: global; formation; body; zone;
CUT(TU)	Number of cutoffs to the turbulence, once per cell; available for: global; formation; body; zone;
CYCLE	Cycle; available for: global; formation; body; zone; force; post;
D	Drag [N]; available for: global; formation; body; zone; force; post;
DEL(D)	Delta-D; available for: global; formation; body; zone; force; post;
DEL(FX)	Delta-Fx; available for: global; formation; body; zone; force; post;

DEL(FY)	Delta-Fy; available for: global; formation; body; zone; force; post;
DEL(FZ)	Delta-Fz; available for: global; formation; body; zone; force; post;
DEL(HEAT.FLUX)	Delta heat flux; available for: post;
DEL(L)	Delta-L; available for: global; formation; body; zone; force; post;
DEL(MASS.FLOW)	Delta mass flow; available for: post;
DEL(MX)	Delta-Mz; available for: global; formation; body; zone; force; post;
DEL(MY)	Delta-My; available for: global; formation; body; zone; force; post;
DEL(MZ)	Delta-Mz; available for: global; formation; body; zone; force; post;
DEL(S)	Delta-S; available for: global; formation; body; zone; force; post;
DT	Physical time delta; available for: global; formation; body; zone;
ETIME	Execution time; available for: global; formation; body; zone; force; post;
F	Forces [N]; available for: global; formation; body; zone; force; post;
FF	Forces that are induced by friction [N]; available for: global; formation; body; zone; force; post;
FORM(B.CF)	Formation referenced force coefficient in body frame; available for: formation; body; zone; force; post;

FORM(B.CFF)	Formation referenced force coefficient from friction in body frame; available for: formation; body; zone; force; post;
FORM(B.CFP)	Formation referenced force coefficient from pressure in body frame; available for: formation; body; zone; force; post;
FORM(B.CM)	Formation referenced Momentum coefficient in body frame; available for: formation; body; zone; force; post;
FORM(B.CMF)	Formation referenced Momentum coefficient from friction in body frame; available for: formation; body; zone; force; post;
FORM(B.CMP)	Formation referenced Momentum coefficient from pressure in body frame; available for: formation; body; zone; force; post;
FORM(B.M)	Formation referenced Momentum in body frame; available for: formation; body; zone; force; post;
FORM(B.MF)	Formation referenced Momentum from friction in body frame; available for: formation; body; zone; force; post;
FORM(B.MP)	Formation referenced Momentum from pressure in body frame; available for: formation; body; zone; force; post;
FORM(O.CD)	Formation referenced drag coefficient in observer frame; available for: formation; body; zone; force; post;
FORM(O.CF)	Formation referenced force coefficient in observer frame; available for: formation; body; zone; force; post;
FORM(O.CFF)	Formation referenced force coefficient from friction in observer frame; available for: formation; body; zone; force; post;
FORM(O.CFP)	Formation referenced force coefficient from pressure in observer frame; available for: formation; body; zone; force; post;
FORM(O.CL)	Formation referenced lift coefficient in observer frame; available for: formation; body; zone; force; post;

FORM(O.CM)	Formation referenced Momentum coefficient in observer frame; available for: formation; body; zone; force; post;
FORM(O.CMF)	Formation referenced Momentum coefficient from friction in observer frame; available for: formation; body; zone; force; post;
FORM(O.CMP)	Formation referenced Momentum coefficient from pressure in observer frame; available for: formation; body; zone; force; post;
FORM(O.CS)	Formation referenced side coefficient in observer frame; available for: formation; body; zone; force; post;
FORM(O.M)	Formation referenced Momentum in observer frame; available for: formation; body; zone; force; post;
FORM(O.MF)	Formation referenced Momentum from friction in observer frame; available for: formation; body; zone; force; post;
FORM(O.MP)	Formation referenced Momentum from pressure in observer frame; available for: formation; body; zone; force; post;
FORM(REFPOINT)	Formation Reference Point Over Which Moments Are Summed FORM(O.M); available for: formation; body; zone; post;
FP	Forces that are induced by pressure [N]; available for: global; formation; body; zone; force; post;
HEAT.FLUX	Summed heatflux through surfaces; available for: post;
IRDV	Integral of Rho DV; available for: global; formation; body; zone; post;
ITER	Iteration; available for: global; formation; body; zone; force; post;
ITIME	Iteration time; available for: global; formation; body; zone; force; post;

L	Lift [N]; available for: global; formation; body; zone; force; post;
LEVL	Multigrid level; available for: global; formation; body; zone; force; post;
LOG(D)	Log(Delta-D); available for: global; formation; body; zone; force; post;
LOG(FX)	Log(Delta-Fx); available for: global; formation; body; zone; force; post;
LOG(FY)	Log(Delta-Fy); available for: global; formation; body; zone; force; post;
LOG(FZ)	Log(Delta-Fz); available for: global; formation; body; zone; force; post;
LOG(L)	Log(Delta-L); available for: global; formation; body; zone; force; post;
LOG(MASS)	Log(mass eq. residual); available for: global; formation; body; zone;
LOG(MOMX)	Log(momentum-x eq. residual); available for: global; formation; body; zone;
LOG(MOMY)	Log(momentum-y eq. residual); available for: global; formation; body; zone;
LOG(MOMZ)	Log(momentum-z eq. residual); available for: global; formation; body; zone;
LOG(MX)	Log(Delta-Mx); available for: global; formation; body; zone; force; post;
LOG(MY)	Log(Delta-My); available for: global; formation; body; zone; force; post;

LOG(MZ)	Log(Delta-Mz); available for: global; formation; body; zone; force; post;
LOG(NRGY)	Log(energy eq. residual); available for: global; formation; body; zone;
LOG(NS)	Log10(Res/Res0) (Navier-Stokes L2Norm); available for: global; formation; body; zone;
LOG(RGMA)	Log10(Res/Res0) of the relevant parameter; available for: global; formation; body; zone;
LOG(RHOK)	Log10(K eq. residual); available for: global; formation; body; zone;
LOG(RHOO)	Log10(Omega eq. residual); available for: global; formation; body; zone;
LOG(RRTH)	Log10(Res/Res0) of the relevant parameter; available for: global; formation; body; zone;
LOG(RXX)	Log10(Res/Res0) (Rxx eq. residual); available for: global; formation; body; zone;
LOG(RXY)	Log10(Res/Res0) (Rxy eq. residual); available for: global; formation; body; zone;
LOG(RXZ)	Log10(Res/Res0) (Rxz eq. residual); available for: global; formation; body; zone;
LOG(RYY)	Log10(Res/Res0) (Ryy eq. residual); available for: global; formation; body; zone;
LOG(RYZ)	Log10(Res/Res0) (Ryz eq. residual); available for: global; formation; body; zone;
LOG(RZZ)	Log10(Res/Res0) (Rzz eq. residual); available for: global; formation; body; zone;

LOG(S)	Log(Delta-S); available for: global; formation; body; zone; force; post;
LOG(TU)	Log10(Res/Res0) (Turbulent L2Norm); available for: global; formation; body; zone;
M	Moments [Nxm]; available for: global; formation; body; zone; force; post;
MASS	DOF Mass; available for: formation; post;
MASS.FLOW	Summed numeric mass flow through surfaces; available for: post;
MASS.FLOW.G	Summed mass flow through surfaces, based on the ghost cells; available for: post;
MASS.FLOW.R	Summed mass flow through surfaces, based on the real cells; available for: post;
MAX(CFL)	Maximal CFL (unsteady); available for: global; formation; body; zone;
MAX(CFLH)	Maximal XLES CFL; available for: global; formation; body; zone;
MAX(MT)	Maximal Mu_t; available for: global; formation; body; zone;
MAX(NS)	Maximal cell Navier-Stokes residual; available for: global; formation; body; zone;
MAX(P)	Maximum Pressure; available for: global; formation; body; zone;
MAX(T)	Maximum Temperature; available for: global; formation; body; zone;
MAX(TU)	Maximal cell Turbulence residual; available for: global; formation; body; zone;
MAX(YPLUS)	Maximal y+; available for: global; formation; body; zone;

MEM(ALL)	Memory allocate on all nodes; available for: global; post;
MEM(OWN)	Memory dedicated to a single(master) node; available for: global; post;
MF	Moments that are induced by friction [Nxm]; available for: global; formation; body; zone; force; post;
MG-LOG(MASS)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(MOMX)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(MOMY)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(MOMZ)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(NRGY)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(NS)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RGMA)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RHOD)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RHOK)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RHOO)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RRTH)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;

MG-LOG(RXX)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RXY)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RXZ)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RYY)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RYZ)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(RZZ)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-LOG(TU)	Multigrid Log10(Res/REs0) of the relevant parameter; available for: global; formation; body; zone;
MG-NCELLS	Number of Multigrid cells; available for: global; formation; body; zone; post;
MG-RES(MASS)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(MOMX)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(MOMY)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(MOMZ)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(NRGY)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;

MG-RES(NS)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RGMA)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RHOD)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RHOK)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RHOO)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RRTH)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RXX)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RXY)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RXZ)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RYY)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RYZ)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(RZZ)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;
MG-RES(TU)	Multigrid L2Norm of the relevant parameter; available for: global; formation; body; zone;

MIN(DTAU)	Minimal pseudo-time step; available for: global; formation; body; zone;
MIN(K)	Minimal K; available for: global; formation; body; zone;
MIN(O)	Minimal omega; available for: global; formation; body; zone;
MIN(P)	Minimal pressure; available for: global; formation; body; zone;
MIN(T)	Minimal temperature; available for: global; formation; body; zone;
MLOG(MASS)	Log10(Res/Max(Res)) (mass eq. maximal residual); available for: global; formation; body; zone;
MLOG(MOMX)	Log10(Res/Max(Res)) (momentum-x eq. maximal residual); available for: global; formation; body; zone;
MLOG(MOMY)	Log10(Res/Max(Res)) (momentum-y eq. maximal residual); available for: global; formation; body; zone;
MLOG(MOMZ)	Log10(Res/Max(Res)) (momentum-z eq. maximal residual); available for: global; formation; body; zone;
MLOG(NRGY)	Log10(Res/Max(Res)) (energy eq. maximal residual); available for: global; formation; body; zone;
MLOG(NS)	Log10(Res/Max(Res)) (Navier-Stokes maximal L2Norm); available for: global; formation; body; zone;
MLOG(RGMA)	Log10(Res/Max(Res)) of the relevant parameter; available for: global; formation; body; zone;
MLOG(RHOD)	Log10(Res/Max(Res)) of the relevant parameter; available for: global; formation; body; zone;
MLOG(RHOK)	Log10(Res/Max(Res)) of the relevant parameter; available for: global; formation; body; zone;

MLOG(RHOO)	Log10(Res/Max(Res)) of the relevant parameter; available for: global; formation; body; zone;
MLOG(RRTH)	Log10(Res/Max(Res)) of the relevant parameter; available for: global; formation; body; zone;
MLOG(RXX)	Log10(Res/Max(Res)) (Rxx eq. maximal residual); available for: global; formation; body; zone;
MLOG(RXY)	Log10(Res/Max(Res)) (Rxy eq. maximal residual); available for: global; formation; body; zone;
MLOG(RXZ)	Log10(Res/Max(Res)) (Rxz eq. maximal residual); available for: global; formation; body; zone;
MLOG(RYY)	Log10(Res/Max(Res)) (Ryy eq. maximal residual); available for: global; formation; body; zone;
MLOG(RYZ)	Log10(Res/Max(Res)) (Ryz eq. maximal residual); available for: global; formation; body; zone;
MLOG(RZZ)	Log10(Res/Max(Res)) (Rzz eq. maximal residual); available for: global; formation; body; zone;
MLOG(RK)	Log10(Res/Max(Res)) (K eq. maximal residual); available for: global; formation; body; zone;
MLOG(TU)	Log10(Res/Max(Res)) (Turbulent maximal L2Norm); available for: global; formation; body; zone;
MODAL.ETA	MODAL.ETA's (Xi Dot); available for: post; spline;
MODAL.GF	DEFORM Generalized Forces; available for: post; spline;
MODAL.XI	DEFORM Xi's; available for: post; spline;
MODAL.ZR	DEFORM Damping of relaxation; available for: post; spline;
MP	Moments that are induced by pressure [Nxm]; available for: global; formation; body; zone; force; post;

NCELLS	Number of cells; available for: global; formation; body; zone; post;
NUMB	Multigrid cycle inner iteration number; available for: global; formation; body; zone; force; post;
O.ATD.A.DEG	DOF body frame angular acceleration in observer frame $[deg/s^2]$, calculated as inverse of B.ATD.A.DEG; available for: formation; post;
O.ATD.A.RAD	DOF body frame angular acceleration in observer frame $[rad/s^2]$, calculated as inverse of B.ATD.A.RAD; available for: formation; post;
O.ATD.V.DEG	DOF body frame angular velocity in observer frame [deg/s], calculated as inverse of B.ATD.V.DEG; available for: formation; post;
O.ATD.V.RAD	DOF body frame angular velocity in observer frame [rad/s], calculated as inverse of B.ATD.V.RAD; available for: formation; post;
O.CENTER	Center of Mass; available for: formation; post;
O.CF	DOF Force sum coefficient, in observer frame; available for: formation;
O.CM	DOF Moment sum coefficient, in observer frame; available for: formation;
O.EA.DEG	DOF Euler angles in observer frame [deg]; available for: formation; post;
O.EA.RAD	DOF Euler angles in observer frame [rad]; available for: formation; post;
O.F	DOF Force Sum, in observer frame [N]; available for: formation;

O.M	DOF Moment sum, in observer frame [Nxm]; available for: formation;
O.ORG	DOF body frame origin, in observer frame [m]; available for: formation; post;
O.ORG.A	DOF body frame translation acceleration, in observer frame [m/s^2]; available for: formation; post;
O.ORG.V	DOF body frame translation velocity, in observer frame [m/s]; available for: formation; post;
O.QUATERNION	DOF quaternion in observer frame; available for: formation; post;
POS(MAX(CFL))	Position of Maximal CFL (unsteady); available for: global; formation; body; zone;
POS(MAX(CFLH))	Position of Maximal Xles CFL; available for: global; formation; body; zone;
POS(MAX(MT))	Position of Maximal Mu_t; available for: global; formation; body; zone;
POS(MAX(NS))	Position of Maximal cell Navier-Stokes residual; available for: global; formation; body; zone;
POS(MAX(P))	Position of Maximum Pressure; available for: global; formation; body; zone;
POS(MAX(T))	Position of Maximum Temperature; available for: global; formation; body; zone;
POS(MAX(TU))	Position of Maximal cell Turbulence residual; available for: global; formation; body; zone;
POS(MAX(YPLUS))	Position of maximal y+; available for: global; formation; body; zone;

POS(MIN(DTAU))	Position of minimal pseudo-time step; available for: global; formation; body; zone;
POS(MIN(P))	Position of Minimal pressure; available for: global; formation; body; zone;
POS(MIN(T))	Position of Minimal temperature; available for: global; formation; body; zone;
REFPOINT	Reference Point Over Which Moments Are Summed; available for: global; formation; body; zone; force; post;
REINITIALIZATION	reinitialization step; available for: global; post;
RES(MASS)	Mass eq. residual; available for: global; formation; body; zone;
RES(MOMX)	Momentum-x eq. residual; available for: global; formation; body; zone;
RES(MOMY)	Momentum-y eq. residual; available for: global; formation; body; zone;
RES(MOMZ)	Momentum-z eq. residual; available for: global; formation; body; zone;
RES(NRGY)	Energy eq. residual; available for: global; formation; body; zone;
RES(NS)	Navier-Stokes L2Norm; available for: global; formation; body; zone;
RES(RGMA)	L2Norm of the relevant parameter; available for: global; formation; body; zone;
RES(RHOD)	L2Norm of the relevant parameter; available for: global; formation; body; zone;
RES(RHOK)	K eq. residual; available for: global; formation; body; zone;

RES(RHOO)	Omega eq. residual; available for: global; formation; body; zone;
RES(RRTH)	L2Norm of the relevant parameter; available for: global; formation; body; zone;
RES(RXX)	Rxx eq. residual; available for: global; formation; body; zone;
RES(RXY)	Rxy eq. residual; available for: global; formation; body; zone;
RES(RXZ)	Rxz eq. residual; available for: global; formation; body; zone;
RES(RYY)	Ryy eq. residual; available for: global; formation; body; zone;
RES(RYZ)	Ryz eq. residual; available for: global; formation; body; zone;
RES(RZZ)	Rzz eq. residual; available for: global; formation; body; zone;
RES(TU)	Turbulent L2Norm; available for: global; formation; body; zone;
RLZ(O)	Number of omega realizability criteria reached; available for: global; formation; body; zone;
STEP	Dual time step; available for: global; formation; body; zone; force; post;
STIME	Step time; available for: global; formation; body; zone; force; post;
S	Side [N]; available for: global; formation; body; zone; force; post;
TIME	Dual time simulation-physical-time; available for: global; formation; body; zone; force; post;
TRN.MOM	translational momentum for cylindrical axes; available for: post;
TREND	trend; available for: global; formation; body; zone; force; post;

U	Vertex X velocity in observer frame; available for: post; vertex;
UVW	Vertex velocity in observer frame; available for: post; vertex;
U.DOT	Vertex X acceleration in observer frame; available for: post; vertex;
UVW.DOT	Vertex acceleration in observer frame; available for: post; vertex;
V	Vertex Y velocity in observer frame; available for: post; vertex;
V.DOT	Vertex Y acceleration in observer frame; available for: post; vertex;
W	Vertex Z velocity in observer frame; available for: post; vertex;
W.DOT	Vertex Z acceleration in observer frame; available for: post; vertex;
WETS	Wet Surface Summed in Forces; available for: global; formation; body; zone; force; post;
X	Vertex X coordinate position; available for: formation; post; vertex;
XYZ	Vertex full position; available for: formation; post; vertex;
Y	Vertex Y coordinate position; available for: formation; post; vertex;
Z	Vertex Z coordinate position; available for: formation; post; vertex;

Table B.1: Log options

Appendix C

Function File Types

Function File Types

File type	File content
xyz	Grid file.
xyz.center	Grid centered file.
xyzb	Grid/iblank file.
xyzb.center	Grid/iblank centered file.
xyzc	grid/iblank file where chimera are also blanked.
xyzc.center	grid/iblank centered file where chimera are also blanked.
q	Solution file.
<RK>/<R>	Average of rk relative to r.
<RU>/<R>	Average of ru relative to r.
<RUU>/<R>	Average of ruu relative to r.
<RUV>/<R>	Average of ruv relative to r.
<RUW>/<R>	Average of ruw relative to r.

<RV>/<R>	Average of rv relative to r.
<RVV>/<R>	Average of rvv relative to r.
<RVVEC>/<R>	conservatives mometum average relative to density.
<RVW>/<R>	Average of rvw relative to r.
<RVV>/<R>	Average of rvv relative to r.
<RW>/<R>	Average of rw relative to r.
<RWW>/<R>	Average of rww relative to r.
AC	Speed of sound.
AMR.COARSE	AMR coarsening criteria.
AMR.REFINE	AMR refinement criteria.
AVG(K)	Average of turbulent kinetic energy.
AVG(R)	Average of density.
AVG(RXX)	Average of XX component in the averaged resolved stress tensor.
AVG(RXY)	Average of XY component in the averaged resolved stress tensor.
AVG(RXZ)	Average of XZ component in the averaged resolved stress tensor.
AVG(RYY)	Average of YY component in the averaged resolved stress tensor.
AVG(RYZ)	Average of YZ component in the averaged resolved stress tensor.
AVG(RZZ)	Average of ZZ component in the averaged resolved stress tensor.

AVG(TTXX)	Average of XX component in the turbulent stress tensor.
AVG(TTXY)	Average of XY component in the turbulent stress tensor.
AVG(TTXZ)	Average of XZ component in the turbulent stress tensor.
AVG(TTYY)	Average of YY component in the turbulent stress tensor.
AVG(TTYZ)	Average of YZ component in the turbulent stress tensor.
AVG(TTZZ)	Average of ZZ component in the turbulent stress tensor.
AVG(U)	Average of u velocity.
AVG(UU)	Average of u * u velocity.
AVG(UV)	Average of u * v velocity.
AVG(UW)	Average of u * w velocity.
AVG(V)	Average of v velocity.
AVG(VV)	Average of v * v velocity.
AVG(VVEC)	velocity average vector (when applicable).
AVG(VW)	Average of v * w velocity.
AVG(V^2)	Average of u * u + v * v + w * w velocity.
AVG(W)	Average of w velocity.
AVG(WW)	Average of w * w velocity.
CELL.DT	Cell time step.
CELLS.PER.EDDY	number of cells that resolve an average eddy in hybrid simulation (should be above 5 in LES region).

CES	L+, K_res, epsilon_res.
CF	Cf; for tables or cells in turbulent simulation, and for entity edges only.
CFLH	Cell CFLH (for hybrid turbulence models).
CFVEC	Cf Vector; for tables in turbulent simulation, and on entity edges only.
CP	Pressure coefficient.
CPTOTAL	Total pressure coefficient.
DNU	Spalart-Allmaras, damped turbulent viscosity.
DONNER	Main donor zone (negative for multiple zones).
DSF	Donor suitability function.
ELEC.COND	Electrical conductivity.
EPPIC.ERR	EPPIC error analysis function.
ET	Total energy.
ETA	J (η) coordinate.
FORCE.FLAG	Force Flag.
FORMATION.DONNER	Main formation donor (negative for multiple formations).
GAMMART	Returns the gammaRT bit field.
GAVEC	Grid acceleration vector.
GAX	Grid acceleration in x direction.
GAY	Grid acceleration in y direction.
GAZ	Grid acceleration in z direction.
GRAD(U)	Gradient of u velocity.
GRAD(V)	Gradient of v velocity.

GRAD(W)	Gradient of w velocity.
GRID.DEFORMATION	Grid deformation in body frame.
GRID.LENGTH	Length measurements of a grid cell.
GVVEC	Grid velocity vector.
GVX	Grid velocity in x direction.
GVY	Grid velocity in y direction.
GVZ	Grid velocity in z direction.
HT	Total enthalpy.
HYBRID.RANS.LES	Hybrid des flag: rans(1)/les(0) (xles included).
HYBRID.SHIELD	The hybrid shield (0 - shielded, 1 - normal).
HYBRID.SUBGRID.SCALE.FI	Hybrid subgrid scale filter.
HYBRID.UPWIND.CENTRAL	Hybrid Central Upwind blend: upwind(1)/central(0).
HYBRID.UPWIND.CENTRAL	Hybrid Central Upwind blend: upwind(1)/central(0) for the mass/energy equations.
IBLANK	Iblank flag.
INTERPOLATION.ERR	No interpolation (6000) or logical/trilinear/inner/freestream/unhandled warnings (2/4/8/16/128) wrn/err layers (32/64).
INTERPOLATION.TYPE	Geometric/logical/trilinear/MB/AMR types (1/2/4/8/16).
IZOMBIE	iZombie Flag.
K	Turbulent kinetic energy.
LAMBDA2	Lambda2 criterion.
LIM.MIN.VANALBADA	Minimal cell limiter of all mean-flow primitives assuminng vanalbada limiter.

LMAX	cell lmax.
MA	Mach number.
MG.DAMP	Multigrid damping function.
MT	Turbulent viscosity.
MU	Viscosity.
MTTOTAL	Turbulent total viscosity.
NB	Boundary Conditions (see Table D.2 for legend).
NC	Chimera.
NERR	Holes information.
NF	Forces.
O	Specific turbulent dissipation rate.
OMIN	Omega turbulent realizability condition.
OPT.DONNER	Donor information, for developers use.
OPT.DSF	Optimal dsf found for the current cell.
OPT.ZONE	Optimal zone found for the current cell.
P	Pressure.
PANS	K, K_resolve, Fk.
PTOTAL	Total pressure.
QC	Q criterion.
QFLUX.1ST.ORDER	Heat flux; 1st oder approximation; for tables or cells in viscous simulation, and on entity edges only.
QFLUX	Heat flux; for tables or cells in turbulent simulation, and on entity edges only.
R	Density.

RES(MASS)	Mass Equation Residual.
RES(MOMX)	Momentum-x equation Residual.
RES(MOMY)	Momentum-y equation Residual.
RES(MOMZ)	Momentum-z equation Residual.
RES(NRGY)	Energy equation Residual.
RES(NS)	Mean flow residual.
RES(TU)	Turbulence equations residual.
RTHETA	Transition model Re-theta.
RXX	Rxx normal stress.
RXY	Rxy shear stress.
RXZ	Rxz shear stress.
RYY	Ryy normal stress.
RYZ	Ryz shear stress.
RZZ	Rzz normal stress.
SC	Schlieren numerical image.
SH.SENS	Shock sensor.
STAT.AVG(P) - statistical average of pressure.	
STAT.AVG(R) - statistical average of density.	
STAT.AVG(U) - statistical average of u-velocity.	
STAT.AVG(V) - statistical average of v-velocity.	
STAT.AVG(W) - statistical average of w-velocity.	

STENCIL.ETA	Stencil size in the η direction.
STENCIL.XSI	Stencil size in the ξ direction.
STENCIL.ZETA	Stencil size in the ζ direction.
STRUCTURE.ERR	Negative jacobian / unwise cells (1/2).
SUBGRID.SCALE.FILTER	The subgrid scale filter of a grid cell.
T	Temperature.
TRB.RESTRICT	trb.restrict flag.
TRBGMA	Transition model intermittency factor.
TW	Wall normal shear stress component (τ_{wall}); for tables or cells in turbulent simulation, and for entity edges only.
TWVEC	Wall shear vector ($\overline{\tau}_{wall}$); for tables in turbulent simulation, and on entity edges only.
TWX	X direction component of the wall normal shear stress component (τ_{wall}); for tables or cells in turbulent simulation, and for entity edges only.
TWY	Y direction component of the wall normal shear stress component (τ_{wall}); for tables or cells in turbulent simulation, and for entity edges only.
TWZ	Z direction component of the wall normal shear stress component (τ_{wall}); for tables or cells in turbulent simulation, and for entity edges only.
U	U velocity.
USER.SOURCE.MASS	User source addition to mass equation.
USER.SOURCE.MOMX	User source addition to x-momentum equation.
USER.SOURCE.MOMY	User source addition to y-momentum equation.

USER.SOURCE.MOMZ	User source addition to z-momentum equation.
USER.SOURCE.NRGY	User source addition to nrgy equation.
V	V velocity.
VMAG	Velocity magnitude.
VOL	Cell volume.
VVEC	Velocity vector.
W	W velocity.
WALL.DISTANCE	Distance to the nearest wall.
X	X coordinate.
XLES	XLES flag (obsolete).
XSI	I (ξ) coordinate.
Y	Y coordinate.
YPLUS	y^+; for tables or cells in turbulent simulation, and on entity edges only.
Z	Z coordinate.
ZETA	K (ζ) coordinate.
ZONE.ID	Zone ID.

Table C.1: Function file types

Appendix D

NERR and NB Legends

NERR Legend

Value	Description
-1001	Hole due to body.
-2001	Hole due to phantom.
-3001	Hole due to zone.
-4001	Hole due to envelope.
-5001	Hole due to kill.
-6001	Hole due to minimization.

Table D.1: ERR file legend

NB Legend

Value	Description
0	No boundary condition
1	2-D boundary condition
2	No slip boundary condition
3	Impermeable boundary condition
4	Slip boundary condition
5	mass flow inlet boundary condition
6	mass flow outlet boundary condition
7	Turkel boundary condition
8	Turkel inlet boundary condition
9	Turkel outlet boundary condition
10	Riemann boundary condition
11	Riemann inlet boundary condition
12	Riemann outlet boundary condition
13	Symmetry boundary condition
14	Omesh boundary condition
15	Emesh.i boundary condition
16	Emesh.j boundary condition
17	Emesh.k boundary condition
18	Axis boundary condition
19	Axes boundary condition
20	Extrapolation boundary condition
24	Intake boundary condition
25	Injet boundary condition
26	Periodic rotating boundary condition
27	Periodic parallel boundary condition
28	Fixed boundary condition
29	In/out boundary condition
30	Inlet boundary condition
31	Outlet boundary condition
1001	Multi-block boundary cell

Table D.2: NB file legend

Bibliography

[1] Benek J. A.and Buning P. G. and Steger J. L. A 3-d chimera grid embedding technique. In *7th Computational Physics Conference*, Cincinnati, OH, July 1985. AIAA Paper 85-1523.

[2] Baldwin B. S. and Lomax H. Thin layer approximation and algebraic model for separated turbulent flows. In *AIAA 16th Aerospace Sciences Meeting*, Reno, Nevada, January 1978. AIAA Paper 78-257.

[3] Cebeci T., Smith A. M. O., and Mosinkis G. Calculation of compressible adiabatic turbulent boundary layers. *AIAA Journal*, 8:1974–1982, 1970.

[4] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. 1992. AIAA paper 92 - 0439.

[5] Jack R. Edwards and Suresh Chandra. Comparison of eddy viscosity-transport turbulence models for three-dimensional, shock-separted flow fields. *AIAA Journal*, 34(4):756–763, 1996.

[6] Steven R Allmaras and Forrester T Johnson. Modifications and clarifications for the implementation of the spalart-allmaras turbulence model. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, pages 1–11, 2012.

[7] CL Rumsey, J-R Carlson, TH Pulliam, and PR Spalart. Improvements to the quadratic constitutive relation based on nasa juncture flow data. *AIAA Journal*, 58(10):4374–4384, 2020.

[8] Philippe R Spalart and Christopher L Rumsey. Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA journal*, 45(10):2544–2553, 2007.

[9] F. R. Menter. Zonal two equation $k-\omega$ turbulence models for aerodynamic flows. In *24rd AIAA Fluid Dynamics Conference*, Orlando, FL, July 1993. AIAA paper 93 - 2906.

[10] Yu Egorov and F Menter. Development and application of sst-sas turbulence model in the desider project. In *Advances in Hybrid RANS-LES Modelling: Papers contributed to the 2007 Symposium of Hybrid RANS-LES Methods, Corfu, Greece, 17-18 June 2007*, pages 261–270. Springer, 2008.

[11] FX Trias, Andrei Gorobets, MH Silvis, RWCP Verstappen, and A Oliva. A new subgrid characteristic length for turbulence simulations on anisotropic grids. *Physics of Fluids*, 29(11):115109, 2017.

[12] C. Kok, Johan, H. S. Doi, B. Oskam, and H. van der Van. Extra-large eddy simulation of massively separated flows. In *42th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 2004. AIAA paper 2004 - 264.

[13] Mikhail S Gritskevich, Andrey V Garbaruk, Jochen Schütze, and Florian R Menter. Development of ddes and iddes formulations for the k-ω shear stress transport model. *Flow, turbulence and combustion*, 88(3):431–449, 2012.

[14] René-Daniel Cécora, Bernhard Eisfeld, Axel Probst, Simone Crippa, and Rolf Radespiel. Differential reynolds stress modeling for aeronautics. In *50th AIAA Aerospace Sciences Meeting*, volume 465, 2012.

[15] H. Luo, J. D. Baum, and R. Lohner. Extension of Harten-Lax-van Leer scheme for flows at all speeds. *AIAA Journal*, 43(6):1160–1166, 2005.

[16] Ami Harten, Peter D. Lax, and Bram Van Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25(1):35–61, 1983.

[17] E. F. Toro, M. Spruce, and W. Spears. Restoration of the contact surface in the hll riemann solver. *Shock Waves*, 4(4):25–34, 1994.

[18] P. Batten, M. A. Leschziner, and U. C. Goldberg. Average-state Jacobians and implicit methods for compressible viscous and turbulent flows. *Journal of Computational Physics*, 137(1):38–78, 1997.

[19] B. Einfeldt, C. D. Munz, P. L. Roe, and B. Sjogreen. On Godunov-type methods near low densities. *Journal of Computational Physics*, 92(2):273–295, 1991.

[20] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[21] SF Davis. Simplified second-order godunov-type methods. *SIAM Journal on Scientific and Statistical Computing*, 9(3):445–473, 1988.

[22] Meng-Sing Liou and Christopher J. Steffen Jr. A new flux splitting scheme. *Journal of Computational Physics*, 107(1):23–39, 1993.

[23] Meng-Sing Liou. A sequel to AUSM: $AUSM^+ - up$. *Journal of Computational Physics*, 129:364–382, 1996.

[24] Meng-Sing Liou. A sequel to AUSM, part II: $AUSM^+ - up$ for all speeds. *Journal of Computational Physics*, 214:137–170, 2006.

[25] Yasuhiro Wada and Meng-Sing Liou. A Flux Splitting Scheme with High-resolution and Robustness for Discontinuities. In *32th AIAA Aerospace Science Meeting and Exhibit*, page 23, Reno, NV, 1994.

[26] Michael Dean Neaves and Jack R Edwards. All-speed time-accurate underwater projectile calculations using a preconditioning algorithm. *Journal of fluids engineering*, 128(2):284–296, 2006.

[27] Meng-Sing Liou. Progress towards an improved cfd method-ausm+. In *AIAA 12th Computational Fluid Dynamics Conference*, 1995. AIAA Paper 1995-1701.

[28] Daniel A Cassidy, Jack R Edwards, and Ming Tian. An investigation of interface-sharpening schemes for multi-phase mixture flows. *Journal of Computational Physics*, 228(16):5628–5649, 2009.

[29] PL Roe and J Pike. Efficient construction and utilisation of approximate riemann solutions. In *Proc. of the sixth int'l. symposium on Computing methods in applied sciences and engineering, VI*, pages 499–518. North-Holland Publishing Co., 1985.

[30] Hervé Guillard and Cécile Viozat. On the behaviour of upwind schemes in the low mach number limit. *Computers & fluids*, 28(1):63–86, 1999.

[31] Felix Rieper. A low-mach number fix for roe's approximate riemann solver. *Journal of Computational Physics*, 230(13):5263–5287, 2011.

[32] Y Mor-Yossef. Aufsr+: Low mach number enhancement of the aufsr scheme. *Computers & Fluids*, 136:301–311, 2016.

[33] Eiji Shima and Keiichi Kitamura. Parameter-free simple low-dissipation ausm-family scheme for all speeds. *AIAA journal*, 49(8):1693–1709, 2011.

[34] Kai Oßwald, Alexander Siegmund, Philipp Birken, Volker Hannemann, and Andreas Meister. L2roe: a low dissipation version of roe's approximate riemann solver for low mach numbers. *International Journal for Numerical Methods in Fluids*, 2015.

[35] Bram van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101–136, 1979.

[36] B van Leer. Upwind-difference methods for aerodynamic problems governed by the Euler equations. In *Large-Scale Computations in Fluid Mechanics; Proceedings of the Fifteenth Summer Seminar on Applied Mathematics*, pages 327–336, 1985.

[37] G D van Albada, B van Leer, and W W Roberts. A comparative study of computational methods in cosmic gas dynamics. *Astronomy and Astrophysics*, 108(1):76–84, 1982.

[38] A Suresh and HT Huynh. Accurate monotonicity-preserving schemes with runge–kutta time stepping. *Journal of Computational Physics*, 136(1):83–99, 1997.

[39] G S Jiang and C W Shu. Efficient implementation of weighted ENO schemes. 1995.

[40] Xinrong Su, Daisuke Sasaki, and Kazuhiro Nakahashi. On the efficient application of weighted essentially nonoscillatory scheme. *International Journal for Numerical Methods in Fluids*, 71(2):185–207, February 2012.

[41] Yiqing Shen, Gecheng Zha, and Baoyuan Wang. Improvement of Stability and Accuracy for Weighted Essentially Nonoscillatory Scheme. *AIAA journal*, 47(2):331–344, February 2009.

[42] Zhixiang Xiao, Jian Liu, Jingbo Huang, and Song Fu. Numerical dissipation effects on massive separation around tandem cylinders. *AIAA journal*, 50(5):1119–1136, 2012.

[43] Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[44] Joseph L Steger and R F Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. *Journal of Computational Physics*, 40(2):263–293, 1981.

[45] C. C. Rossow. A flux-spltting scheme for compressible and incompressible flows. *Journal of Computational Physics*, 164(1):104–122, 2000.

[46] Culbert B Laney. *Computational Gasdynamics*. Cambridge University Press, May 1998.

[47] Bernard Parent. Positivity-preserving high-resolution schemes for systems of conservation laws. *Journal of Computational Physics*, 231(1):173–189, 2012.

[48] J. M. Weiss and W. A. Smith. Preconditioning applied to variable and constant density time-accurate flows on unstructured meshes. *AIAA Journal*, 33(11):2050–2057, 1995.

[49] A Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computations*, 31(138):333–390, 1977.

[50] AP Verweij. An investigation in actuator disc cfd solution applicabillity for aeroacoustic analysis of propellers and rotors. 2010.