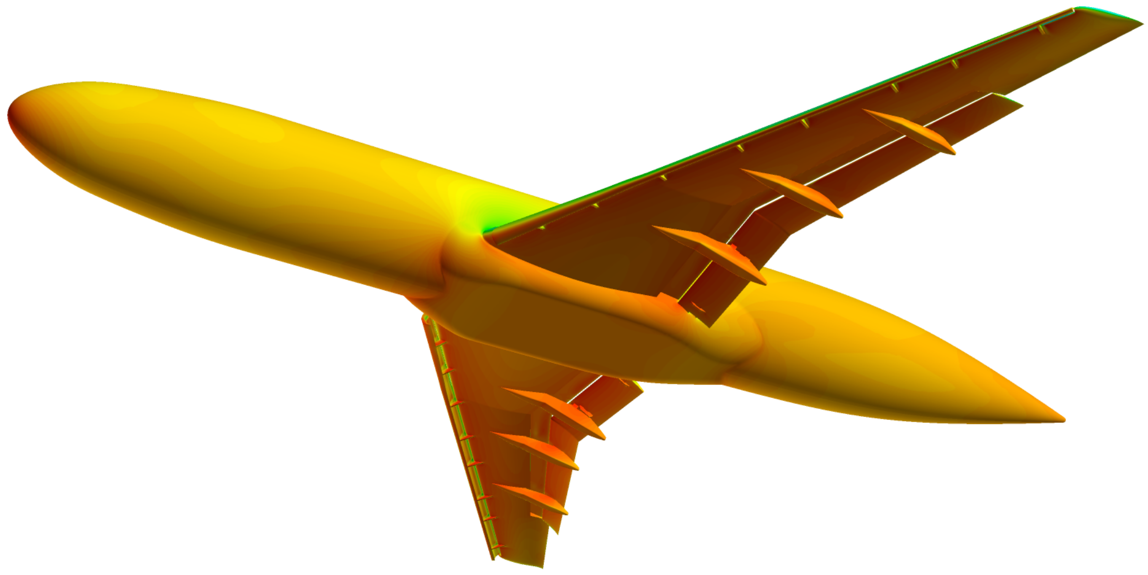
THE ARION UNSTRUCTURED GRIDS CFD SOLVER: THEORETICAL AND USER'S MANUAL

Version 2.60 (Build 2025-2-g0834bb1a)

ISCFDC Report 2025-09



Prepared by

Ya'eer Kidron, Sahar Shpitz, Noga Obstbaum, and Yuval Levy

ISCFDC LTD

Copyright and Trademark Information

Third-Party Software

The PETSc Open Source Library

This software makes use of the PETSc open-source library.
The following is legal information pertaining to the use and redistribution of PETSc:

The CGNS Library

This software makes use of the CGNS (CFD General Notation System) library.

The HDF Library

The METIS Library

Contents

Abstract **xxi**

1 Introduction **1**

1.1 Current Release 1

1.2 Users's Manual Arrangement 2

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 Equation of State 6

2.2.4 Transport Properties 7

2.3 Multi-component Gas (MCG) 8

2.3.1 Basic Assumptions 9

2.3.2 Governing Equations 9

2.3.3 Integral Form For Moving Grids 12

2.3.4 Thermodynamic Properties 14

2.3.4.1 Thermal Equilibrium 14

2.3.5 Transport Properties 14

2.3.5.1 Wilke's Model 15

2.3.6 Chemical Reaction Modeling 16

2.3.6.1 Equilibrium Gas Properties Tabulation 17

2.3.6.2 Finite-rate Chemical Kinetics 17

2.3.6.3 Reaction Rate Coefficients 17

2.4 Gas Selection 18

2.4.1 Single Component Perfect Gas 18

2.4.2 Multi Component Gas 19

3 Turbulence Models **20**

3.1 RANS Turbulence Model Equations 21

3.2 The Spalart Allmaras Model 23

3.2.1 SA-2020 Model 25

3.2.2 Boundary Conditions 26

3.3 The k-ω-TNT Turbulence Model 27

3.4 The k-ω-SST Model 27

3.4.1 k-ω-SST-2003 Model 28

3.4.2 k-ω-SST-SAS Model 30

3.5 Boundary Conditions for the k-ω-TNT and k-ω-SST Turbulence Models 31

3.6 Evaluation of the Reynolds Stress Tensor 32

3.7 Turbulence-Mean-flow Coupling 32

4 Computational Methods **33**

4.1 Spatial Discretization 33

4.2 Flux Approximation Schemes 34

4.2.1 HLLC 34

4.2.2 AUSM 37

4.2.2.1 AUSM^{+}-up 38

4.2.3 Passive Scalar Approach 40

4.2.4 High Order Flux Approximations 41

4.3 Diffusive Flux Vector Discretization 42

4.4 Time Marching Schemes 42

4.4.1 Explicit schemes 42

4.4.1.1 Explicit Euler Scheme 42

4.4.1.2 Runge-Kutta Schemes 43

4.4.1.3 Time-Accurate Third Order Runge-Kutta TVD . . . 43

4.4.2 Implicit Time Marching Formulation . . . 44
4.4.2.1 Point Gauss-Seidel . . . 44
4.4.2.2 Runge-Kutta Schemes . . . 45
4.5 Convergence Acceleration Methods . . . 45
4.5.1 Krylov Methods . . . 45
4.5.1.1 Preconditioning . . . 46
4.5.1.2 Portable, Extensible Toolkit for Scientific Computation (PETSc) . . . 46
4.5.2 Line Search . . . 47
4.5.2.1 Back-Traced Line Search . . . 47

5 Boundary Conditions **49**
5.1 Introduction . . . 49
5.2 Wall Boundary Conditions . . . 49
5.2.1 Impermeable Wall Conditions . . . 49
5.2.2 No-Slip Condition . . . 50
5.2.3 Adiabatic Wall . . . 50
5.3 Far Field Conditions . . . 50
5.3.1 Turkel Type Conditions . . . 50
5.3.1.1 Turkel Inlet . . . 51
5.3.1.2 Turkel Outlet . . . 51
5.3.2 Riemann Type Conditions . . . 51
5.3.2.1 Riemann Inlet . . . 51
5.3.2.2 Riemann Outlet . . . 52
5.3.3 Fixed (Supersonic Inlet) . . . 52
5.3.4 Extrapolation (Supersonic Outlet) . . . 52
5.3.5 Inlet . . . 52
5.3.6 Outlet . . . 53
5.3.6.1 Prescribed Mass Flow Rate . . . 53
5.3.7 Inout . . . 54
5.4 Stagnation Inlet . . . 54

5.5 Symmetry Boundary Conditions . . . 55

6 Computational Mesh **56**

6.1 Star-CD Export . . . 56

6.2 CGNS Export . . . 56

6.3 Automated Mesh Adaptation . . . 57

7 Parallelization **58**

8 Input File, Run Preparation, and Execution **59**

8.1 Preface . . . 59

8.1.1 Scripting Language . . . 60

8.1.2 Basic Input File: Examples . . . 61

8.1.2.1 Single Component Gas Example . . . 61

8.1.2.2 Multi Component Gas Example . . . 63

8.2 Grid Files . . . 64

8.3 Macros . . . 65

8.3.1 Def . . . 65

8.4 Conversion . . . 66

8.4.1 Conversion Between Different Orderings . . . 69

8.5 Log Control Options . . . 70

8.6 Plot Control Options . . . 73

8.7 Table Control Options . . . 76

8.8 Parallel Options . . . 80

8.9 Equation of State Options . . . 82

8.10 Molecular Options . . . 85

8.11 Tabulate Options . . . 89

8.12 Multi Component Gas Options . . . 92

8.13 BC Options . . . 101

8.13.1 BC Log Options . . . 106

8.13.2 Wall Distance Calculations . . . 108

8.13.3 Free-Stream Conditions Override . . . 109

8.14 System Options 114
8.15 Solve Options 129
8.15.1 Line Search 149
8.16 PETSc Toolkit 151
8.17 Domain Options 157
8.18 Init Options 159
8.19 UNS Options 163
8.19.1 Motion: DOF Options 172
8.20 Point Cloud Data 180
8.21 Run-time Options 185
8.21.1 Run-time Options Examples 185

Bibliography **191**

List of Figures

4.1 Wave structure of the HLLC approximate Riemann solver 35

List of Tables

2.1 Default coefficients for Sutherland formulae (type 1) 8
2.2 Default coefficients for Sutherland formulae (type 2) 8

3.1 SST coefficients 29

8.1 Def macro 65
8.2 Conversion 66
8.3 Metis options 67
8.3 Metis options (continued) 68
8.4 Conversion between different ordering (cnv2cnv) option 69
8.5 Log name option 70
8.6 Log prefix option 71
8.7 Log per option 71
8.8 Log stdout option 72
8.9 Log stderr option 72
8.10 Plot name option 73
8.11 Plot prefix option 73
8.12 Plot interval option 74
8.13 Plot sequential/overwrite option 74
8.14 Plot fvuns/eppic/cgns option 75
8.15 Table name option 76
8.16 Table prefix option 76
8.17 Table interval option 77
8.18 Table sequential/overwrite option 77
8.19 Table horizontal/vertical option 77

8.20 Table ray option 78
8.21 Table plane.bc option 78
8.22 Table bc option 79
8.23 Parallel cache option 80
8.24 Parallel rank option 81
8.25 equation.of.state name option 82
8.26 equation.of.state type option 83
8.27 EOS types 83
8.28 equation.of.state R option 83
8.29 equation.of.state gamma option 84
8.30 equation.of.state ref option 84
8.31 Molecular name option 85
8.32 molecular type option 85
8.33 Molecular Mu1 option 86
8.34 Molecular Mu2 option 86
8.35 Molecular Ka1 option 87
8.36 Molecular Ka2 option 87
8.37 Molecular Eta0 option 87
8.38 Molecular C0 option 88
8.39 Molecular T0 option 88
8.40 Molecular Prandtl option 88
8.41 Tabulate name option 89
8.42 Tabulate type option 90
8.43 Tabulate distribution option 90
8.44 Tabulate key option 91
8.45 Tabulate pT.chemistry option 91
8.46 Multi component gas name option 92
8.47 Multi component gas composition option 92
8.48 Multi component gas composition molar option 93
8.49 Multi component gas cy.eq option 93
8.50 Multi component gas cy.cutoff option 94

8.51 Multi component gas thrm.db option 94
8.52 Multi component gas trns.db option 94
8.53 Multi component gas react.db option 95
8.54 Multi component gas cantera.db option 95
8.55 Multi component gas thrm option 96
8.56 Multi component gas trns option 97
8.57 Multi component gas kinetics option 98
8.58 Multi component gas eqconst option 99
8.59 Multi component gas Sc option 99
8.60 Multi component gas Sc.trb option 100
8.61 BC name option 101
8.62 BC type option 101
8.63 Boundary condition types 102
8.64 BC mass.flow option 103
8.65 BC relax option 103
8.66 BC reference.pressure option 104
8.67 BC gas composition option 104
8.68 BC gas composition molar option 105
8.69 BC gas cy.eq option 105
8.70 Boundary conditions log option 106
8.71 BC log options 107
8.72 BC wall.distance option 108
8.73 BC no.wall.distance option 108
8.74 BC Mach option 109
8.75 BC velocity.magnitude option 109
8.76 BC alpha option 110
8.77 BC beta option 110
8.78 BC u option 110
8.79 BC v option 111
8.80 BC w option 111
8.81 BC T option 112

8.82 BC p option . . . 112
8.83 BC trb.intensity option . . . 113
8.84 BC trb.dnu option . . . 113
8.85 System type option . . . 114
8.86 System types . . . 115
8.87 System cell.gradient option . . . 116
8.88 Cell gradient types . . . 116
8.89 System face.gradient option . . . 117
8.90 Face gradient types . . . 117
8.91 Limiter option . . . 118
8.92 Limiter types . . . 118
8.93 System venka.k option . . . 119
8.94 System CFL option . . . 119
8.95 System implicit.jacobi option . . . 120
8.96 System convection.noslip.diagonal option . . . 121
8.97 System convection.impermeable.diagonal option . . . 121
8.98 System convection.symmetry.diagonal option . . . 122
8.99 System realizability.trb.w option . . . 122
8.100 System source.mpk option . . . 123
8.101 System damp option . . . 123
8.102 System relaxation option . . . 124
8.103 System iteration.convergence option . . . 124
8.104 System 2d.tolerance option . . . 124
8.105 System log option . . . 125
8.106 System log options . . . 126
8.107 System plot option . . . 127
8.108 Plot functions . . . 128
8.109 Solve convection.flux option . . . 129
8.110 Convection flux types . . . 129
8.111 Solve convection.jacobian option . . . 130
8.112 Convection Jacobian types . . . 130

8.113 Solve spatial.order option 131
8.114 Solve sweeps option 131
8.115 Solve time.march option 132
8.116 Time marching methods 133
8.117 Solve time.accurate.explicit option 134
8.118 Solve iterations option 134
8.119 Solve time.step.reduction option 135
8.120 Solve dual.time option 135
8.121 Solve global.minimal.timestep option 135
8.122 Solve save option 136
8.123 Solve save.path option 136
8.124 Solve save.sequential option 136
8.125 Solve load.path option 137
8.126 Solve load.sequence option 137
8.127 Solve eos option 137
8.128 Solve molecular option 138
8.129 Solve p option 138
8.130 Solve T option 138
8.131 Solve Mach option 139
8.132 Solve velocity magnitude option 139
8.133 Solve u option 139
8.134 Solve v option 140
8.135 Solve w option 140
8.136 Solve alpha option 140
8.137 Solve beta option 141
8.138 Solve trb.intensity option 141
8.139 Solve trb.mt option 141
8.140 Solve reference.velocity option 142
8.141 Solve reference.mach option 142
8.142 Solve reference.pressure option 142
8.143 Solve reference.velocity.trb option 143

8.144 Solve reference.mach.trb option 143
8.145 Solve reference.length option 144
8.146 Solve ref option 144
8.147 Solve plot option 145
8.148 Solve plot options 145
8.149 Solve log option 145
8.150 Solve log options 146
8.151 Solve wall distance option 147
8.152 Solve source.mom option 147
8.153 Solve axisymmetric option 148
8.154 Solve line search iterations option 149
8.155 Solve line search minimum dq option 149
8.156 Solve line search polynomial order option 150
8.157 Solve line search beta option 150
8.158 Petsc ksp_view option 151
8.159 Petsc ksp_monitor_true_residual option 151
8.160 Petsc ksp_norm_type option 152
8.161 Petsc ksp_max_it option 152
8.162 Petsc ksp_rtol option 153
8.163 Petsc ksp_atol option 153
8.164 Petsc ksp_type option 153
8.165 Petsc KSP Types 154
8.166 Petsc ksp_gmres_restart option 154
8.167 Petsc pc_type option 154
8.168 Petsc sub_pc_type option 155
8.169 Petsc PC Types 155
8.170 Petsc sub_pc_factor_levels option 156
8.171 Petsc sub_pc_factor_fill option 156
8.172 Domain name option 157
8.173 Domain init option 157
8.174 Domain type option 158

8.175 Init name option 159
8.176 Init T option 159
8.177 Init p option 160
8.178 Init u option 160
8.179 Init v option 160
8.180 Init w option 161
8.181 Init composition option 161
8.182 Init composition molar option 162
8.183 UNS name option 163
8.184 UNS prefix option 164
8.185 UNS block option 164
8.186 UNS base option 165
8.187 UNS zone option 165
8.188 UNS crc.sections option 165
8.189 UNS crc.version option 166
8.190 UNS key option 166
8.191 UNS offset option 167
8.192 UNS scale option 167
8.193 UNS rotate option [rad] 168
8.194 UNS rotate option [deg] 168
8.195 UNS log option 169
8.196 UNS log options 171
8.197 UNS DOF angular rate motion option 172
8.198 UNS DOF velocity motion option 173
8.199 UNS DOF acceleration motion option 174
8.200 UNS DOF angular rate init option 175
8.201 UNS DOF velocity init option 176
8.202 UNS DOF acceleration init option 176
8.203 UNS DOF motion origin option 177
8.204 DOF log options 179
8.205 PCD name option 180

8.206 PCD prefix option 181
8.207 PCD mach option 181
8.208 PCD pressure option 181
8.209 PCD density option 182
8.210 PCD velocity.magnitude option 182
8.211 PCD power option 183
8.212 PCD K option 183
8.213 PCD percentage option 183
8.214 PCD spacing option 184
8.215 PCD decay option 184
8.216 Run-time options 185
8.217 Run-time options examples 186

Listings

8.1 Example of a simple **Arion** input file 61
8.2 Example of a pcd section 180

Abstract

This manual describes the algorithms, methods, and input and output files of the **Arion** code. In addition, the manual serves as the user's manual of the code. The current version of the code has gone through a major revision. As of the current version, the code provides the capability to simulate inviscid, laminar, or turbulent steady flows of single-component or multi-component gas. Currently, the flow solver contains the HLLC approximate Riemann solver, the AUSM^{+}-up scheme, or the AUSM-DV schemes for the approximation of the convective fluxes. Turbulence may be modeled using either the k$-\omega$-TNT, the k$-\omega$-SST, or the Spalart-Allmaras turbulence models. The code provides the capability to simulate the flow of any perfect gas under a chemical equilibrium assumption. Near future versions of the code will support simulations under a chemical non-equilibrium assumption. The code is fully parallel using a distributed architecture. Domain decomposition is carried out using the Metis open source library. Convergence acceleration is obtained using a Krylov solver and a line search algorithm.

Chapter 1

Introduction

The **Arion** code is the Israeli CFD Center flow solver for unstructured grids. Along recent years, computational capabilities have been ported from the EZAir Suite of codes into the **Arion** code. This process shall continue in coming months and years at an accelerated pace. In addition, new capabilities are being introduced into the **Arion** code. The goal is to obtain a versatile flow solver that would possess the entire capabilities that the EZAir Suite of codes possesses. This manual describes the algorithms, methods, and input/output files of the **Arion** code. In addition, the manual serves as the user's manual of the code.

1.1 Current Release

The current version of the code has gone through a major revision. As of the current version, the code provides the capability to simulate inviscid, laminar, or turbulent steady flows of single-component or multi-component gas. Currently, the flow solver contains the HLLC approximate Riemann solver, the AUSM$^+$-up scheme, or the AUSM-DV schemes for the approximation of the convective fluxes. Turbulence may be modeled using either the k$-\omega$-TNT, the k$-\omega$-SST, or the Spalart-Allmaras turbulence models. The code provides the capability to simulate the flow of any perfect gas under a chemical equilibrium assumption. Near future versions of the code will support simulations under a chemical non-equilibrium assumption. The code is fully parallel using a distributed architecture. Domain decomposition is carried out using the Metis open source library. Convergence acceleration is obtained using a Krylov solver and a line search algorithm.

1.2 Users's Manual Arrangement

The report is arranged in the following manner: Chapter 2 contains a description of the available physical models while 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 type and format of unstructured meshes that the code supports. Chapter 7 describes the parallelization of the code. Finally, Chapter 8 contains a detailed description of the input file syntax, and actually serves as the code's 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 algorithms, and the quality of the computational mesh. At its current developmental stage, the **Arion** code provides the capability to simulate single-component perfect gas flows, or multi-component perfect gas flows under chemical equilibrium. This chapter contains a description of the physical models that are available in the solver.

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 inviscid flux vectors, E_c, F_c, and G_c, are

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

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

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

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:

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

where

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

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:

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

where u_g, v_g, and w_g are the Cartesian components of the grid velocity vector $\overline{V}_g$. Note that currently the grid velocity vector is set to $\overline{V}_g \equiv 0$.

2.2.3 Equation of State

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 ($R = 287.0$ for air). By assuming further that the gas 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.

2.2.4 Transport Properties

In addition to the equation of state, it is also necessary to establish constitutive relations, namely, relations for the coefficient of viscosity, μ, and the coefficient of thermal conductivity, κ. In the single-component perfect gas (SPG) model, the Sutherland formulae are exclusively used to evaluate these coefficients. The **Arion** code provides two different ways to evaluate μ and κ.

The first (called "molecular.sutherland.air.1", see Table 8.32) is based on the following relations:

$$\begin{aligned} \mu &= C_{\mu_1}\frac{T^{\frac{3}{2}}}{T + C_{\mu_2}} \\ \kappa &= C_{\kappa_1}\frac{T^{\frac{3}{2}}}{T + C_{\kappa_2}} \end{aligned} \tag{2.12}$$

The default of the coefficients C_{μ_1}, C_{μ_2}, C_{κ_1}, and C_{κ_2} correspond to air and are given in Table 2.1. For the purpose of simulating a gas whose transport properties are different than air, the coefficients can be set by the user using the directives as described in Tables 8.33, 8.34, 8.35, 8.36.

Fluid type	C_{μ_1}	C_{μ_2}	C_{κ_1}	C_{κ_2}
Air	1.458×10^{-6}	110.4	2.495×10^{-3}	194

Table 2.1: Default coefficients for Sutherland formulae (type 1)

The second way to evaluate μ and κ (called "molecular.sutherland.air.2", see Table 8.32) is based on the following relations:

$$\begin{aligned} \mu &= \eta_0 \frac{T_0 - C_0}{T - C_0} \left(\frac{T}{T_0} \right)^{\frac{3}{2}} \\ \kappa &= \frac{c_p \mu}{Pr} \end{aligned} \tag{2.13}$$

where P_r is the Prandtl number. The default of the coefficients η_0, T_0, and C_0 correspond to air and are given in Table 2.2. The coefficients can be set by the user using the directives as described in Tables 8.37, 8.38, 8.39, and 8.40.

Fluid type	η_0	C_0	T_0	P_r
Air	1.827×10^{-5}	120	291.15	0.72

Table 2.2: Default coefficients for Sutherland formulae (type 2)

2.3 Multi-component Gas (MCG)

High-speed flows contain physical phenomena that generally may not be modeled by the perfect gas form of the Navier-Stokes equations presented in Section 2.2. Such phenomena include chemical and thermal non-equilibrium, vibrational and electronic excitation, and ionization. These phenomena require the use of a "real gas" model. This section presents the extended form of the Navier-Stokes equations governing the flow of a general gas in chemical and thermal non-equilibrium, and the thermodynamic and transport models that are used to describe the gas. Focus is given to the numerical algorithms that are used in the solution of the extended Navier-Stokes equations and the finite-rate chemical kinetics and energy relaxation models that are employed to model non-equilibrium flow.

2.3.1 Basic Assumptions

The flow is modeled assuming that the continuum approximation is valid. Thermal equilibrium (*i.e.*, all energy modes are described by a single temperature) is assumed for general gases. For simulations involving air, the *two-temperature model* of Park [1] is employed to account for vibrational and electronic non-equilibrium. According to the two-temperature model, the translational and rotational energy modes are described by the translational temperature, T, while the vibrational and electronic energy modes of all species and the electron translational energy mode are described by a second temperature, denoted by T_v. The latter assumption considerably simplifies the system of equations by eliminating additional translational and vibrational energy equations for each polyatomic species and an energy equation for the free electrons. This model was shown to provide accurate results for aerodynamic coefficients and convective heat transfer rates in external flows of air at Mach numbers exceeding 20 [2]. It is therefore assumed adequate for the problems considered for simulation with the present code.

2.3.2 Governing Equations

Under the above assumptions, the extended three-dimensional Navier-Stokes equation set may be expressed in Cartesian coordinates as follows [3]:

$$\frac{\partial Q}{\partial t} + \frac{\partial \left(E_c - E_d\right)}{\partial x} + \frac{\partial \left(F_c - F_d\right)}{\partial y} + \frac{\partial \left(G_c - G_d\right)}{\partial z} = S \tag{2.14}$$

where Q is the vector of conserved variables,

$$Q = \begin{bmatrix} \rho u \\ \rho v \\ \rho w \\ E \\ \rho_1 \\ \vdots \\ \rho_N \\ E_v \end{bmatrix} \tag{2.15}$$

where u,v and w, denote the the Cartesian velocity vector components, E denotes the total (internal and kinetic) energy of the mixture, and E_v denotes the vibrational-electronic energy of the mixture. The mixture density, ρ, is given by a simple sum of the individual chemical species densities, ρ_s, namely,

$$\rho = \sum_{s=1}^{N} \rho_s = \sum_{s=1}^{N} \rho Y_s \tag{2.16}$$

where $Y_{s=1,\dots,N}$ denote the individual chemical species mass fractions. Note that in the case of thermal equilibrium, the mixture vibrational-electronic energy is identically zero ($E_v = 0$) and the last equation in the set is omitted.

The convective (inviscid) flux vectors, E_c, F_c, and G_c, are

$$E_c = \begin{bmatrix} \rho u^2 + p \\ \rho uv \\ \rho uw \\ u\left(E+p\right) \\ \rho_1 u \\ \vdots \\ \rho_N u \\ uE_v \end{bmatrix}, \quad F_c = \begin{bmatrix} \rho uv \\ \rho v^2 + p \\ \rho vw \\ v\left(E+p\right) \\ \rho_1 v \\ \vdots \\ \rho_N v \\ vE_v \end{bmatrix}, \quad G_c = \begin{bmatrix} \rho uw \\ \rho vw \\ \rho w^2 + p \\ w\left(E+p\right) \\ \rho_1 w \\ \vdots \\ \rho_N w \\ wE_v \end{bmatrix} \tag{2.17}$$

and the diffusive (viscous) flux vectors, E_d, F_d, and G_d, are

$$E_d = \begin{bmatrix} \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ \beta_x \\ -J_{x,1} \\ \vdots \\ -J_{x,N} \\ \beta_{v,x} \end{bmatrix}, \quad F_d = \begin{bmatrix} \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ \beta_y \\ -J_{y,1} \\ \vdots \\ -J_{y,N} \\ \beta_{v,y} \end{bmatrix}, \quad G_d = \begin{bmatrix} \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ \beta_z \\ -J_{z,1} \\ \vdots \\ -J_{z,N} \\ \beta_{v,z} \end{bmatrix} \tag{2.18}$$

where

$$\begin{aligned}
\tau_{xx} &= \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial u}{\partial x} \\
\tau_{yy} &= \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial v}{\partial y} \\
\tau_{zz} &= \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial w}{\partial z} \\
\tau_{xy} &= \tau_{yx} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\
\tau_{xz} &= \tau_{zx} = \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\
\tau_{yz} &= \tau_{zy} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\
\beta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - (q_x + q_{v,x}) - \sum_s (J_{x,s} h_s) \\
\beta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - (q_y + q_{v,y}) - \sum_s (J_{y,s} h_s) \\
\beta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - (q_z + q_{v,z}) - \sum_s (J_{z,s} h_s) \\
\beta_{v,x} &= -q_{v,x} - \sum_s (J_{x,s} e_{v,s}) \\
\beta_{v,y} &= -q_{v,y} - \sum_s (J_{y,s} e_{v,s}) \\
\beta_{v,z} &= -q_{v,z} - \sum_s (J_{z,s} e_{v,s})
\end{aligned} \tag{2.19}$$

the pressure is denoted by p, and h_s and $e_{v,s}$ are the specific enthalpy and vibrational energy of species s, respectively. The translational-rotational heat flux is given by $q = -\kappa \nabla T$ while $q_v = -\kappa_v \nabla T_v$ is the vibrational-electronic heat flux, where T, T_v are the translational and vibrational temperatures, respectively. The terms κ, κ_v are the thermal conductivities associated with the various energy modes (*i.e.*, translational and vibrational-electronic).

The species mass diffusion fluxes, $J_{x,s}$, $J_{y,s}$, $J_{z,s}$ are modeled using Fick's law,

$$J_s = -\rho D_s \nabla Y_s \tag{2.20}$$

where D_s is the diffusion coefficient of species s with respect to the mixture (see Section 2.3.5). Stokes hypothesis, $\lambda = -\frac{2}{3}\mu$, is employed to further simplify Equation (2.19).

2.3.3 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:

$$\frac{\partial}{\partial t}\int\limits_{\Omega} Q d\Omega + \oint\limits_{S} d\overline{S} \cdot \overline{H} = 0 \tag{2.21}$$

where

$$\overline{H} = (E_c - E_d)\,\hat{i} + (F_c - F_d)\,\hat{j} + (G_c - G_d)\,\hat{k} \tag{2.22}$$

In Equation (2.21) the vector of dependent variables, Q, remains as in Equation (2.15). Similarly, the viscous flux vectors, E_d, F_d, and G_d, appearing in Equation (2.22), remain as in Equations (2.18) and (2.19) (see Section 2.3.2). In contrast, the inviscid flux vectors take a form that reflects the motion of the grid as follows:

$$\begin{aligned}
E_c &= \begin{bmatrix} \rho u\left(u-u_g\right)+p \\ \rho v\left(u-u_g\right) \\ \rho w\left(u-u_g\right) \\ \left(E+p\right)\left(u-u_g\right)+u_g p \\ \rho_1\left(u-u_g\right) \\ \vdots \\ \rho_N\left(u-u_g\right) \\ \left(u-u_g\right)E_v \end{bmatrix} \\
F_c &= \begin{bmatrix} \rho u\left(v-v_g\right) \\ \rho v\left(v-v_g\right)+p \\ \rho w\left(v-v_g\right) \\ \left(E+p\right)\left(v-v_g\right)+v_g p \\ \rho_1\left(v-v_g\right) \\ \vdots \\ \rho_N\left(v-v_g\right) \\ \left(v-v_g\right)E_v \end{bmatrix} \\
G_c &= \begin{bmatrix} \rho u\left(w-w_g\right) \\ \rho v\left(w-w_g\right) \\ \rho w\left(w-w_g\right)+p \\ \left(E+p\right)\left(w-w_g\right)+w_g p \\ \rho_1\left(w-w_g\right) \\ \vdots \\ \rho_N\left(w-w_g\right) \\ \left(w-w_g\right)E_v \end{bmatrix}
\end{aligned} \tag{2.23}$$

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

2.3.4 Thermodynamic Properties

The pressure, p, is obtained from Dalton's law of partial pressures for a mixture of thermally perfect gases,

$$p = \rho \sum_s R_s Y_s T + \rho R_e Y_e T_v \tag{2.24}$$

where $R_s = \frac{R_0}{W_s}$ is the specific gas constant of species s, with R_0 denoting the universal gas constant and W_s denoting the molecular weight of that species. The subscript e denotes the free electron species.

2.3.4.1 Thermal Equilibrium

If the gas is in thermal equilibrium (*i.e.*, $T = T_v$), then nine-term polynomials of temperature are used for calculating the specific enthalpy, h_s, and the heat capacity at constant pressure, $C_{p,s}$, of individual gaseous species [4]. The total energy is then given by:

$$E = \sum_s \rho Y_s h_s - p + \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right) \tag{2.25}$$

2.3.5 Transport Properties

The transport properties of the mixture may be calculated with three different models. The first, most simple model, employs the Sutherland formulae (see Section 2.2.4) to evaluate the transport properties of standard air at low temperatures (up to 2,000 K). Therefore, it is recommended to use this model only for equilibrium air flows at relatively low speeds. A more elaborate model that accounts for the various components of the gas is based on Wilke's mixing rule [5] together with Blottner's [6] or Mcbride's [7] curve fits for approximating the viscosity, and Eucken's modified formula [8] for calculating the thermal conductivities of individual species. This model is suitable for air at temperatures of up to 10,000 K, and may not be used in simulation of ionized flows. For air at higher temperatures, arising typically in re-entry simulations, a second, more complex model is available owing to Gupta [9].

2.3.5.1 Wilke's Model

In this model, mixture transport properties are obtained using Wilke's mixing rule [5]:

$$\mu = \sum_s \frac{\mu_s X_s}{\phi_s} \quad , \quad \kappa = \sum_s \frac{\kappa_s X_s}{\phi_s} \tag{2.26}$$

where

$$\phi_s = \sum_r X_r \left[1 + \sqrt{\frac{\mu_s}{\mu_r}} \left(\frac{M_r}{M_s} \right)^{\frac{1}{4}} \right]^2 \left[\sqrt{8 \left(1 + \frac{M_s}{M_r} \right)} \right]^{-1} , \tag{2.27}$$

the term X_s denotes the molar fraction of species s in the mixture, μ_s is the species coefficient of viscosity and κ_s is the species thermal conductivity for each energy mode (*i.e.*, translational and vibrational).

The species viscosities are calculated using Blottner's [6] curve-fits of temperature for components of air, or McBride [7] curve-fits for general gaseous species. The species thermal conductivities, κ_s^{tr} and κ_s^{vib}, are determined from a generalized Eucken's relation, according to Hirschfelder [8]:

$$\kappa_s^{tr} = \mu_s \left(\frac{5}{2} C_{v,s}^t + \frac{C_{v,s}^r}{Sc} \right) \tag{2.28}$$

$$\kappa_s^{vib} = \mu_s \left(C_{v,s}^{vib} \right) \tag{2.29}$$

The terms $C_{v,s}^t, C_{v,s}^r, C_{v,s}^{vib}$ denote the translational, rotational and vibrational specific heats at constant volume of species s, respectively, and Sc is a constant Schmidt number, assumed equal for all species.

Finally, the mass diffusion constant of species s with respect to the mixture, D_s, is replaced by a single binary coefficient, D, that is evaluated from a constant, user-supplied Schmidt number, as follows:

$$D = \frac{\mu}{\rho Sc} \tag{2.30}$$

2.3.6 Chemical Reaction Modeling

The local chemical composition of the mixture may be determined in three different methods, ordered here by increasing computational complexity:

1. *Frozen gas*: Under this assumption, the chemical composition remains fixed throughout the flow-field, and is set to the composition of the free-stream. This mode is suitable only for air at low temperatures (up to 2000 K), and for general gases that are known to be chemically non-reacting at the simulated conditions.

2. *Equilibrium gas*: Under this assumption, the local chemical composition is determined uniquely as a function of two thermodynamic variables (e.g., temperature and pressure) via minimization of Gibbs or Helmholtz free energy [10]. For enhanced efficiency, the equilibrium properties of the mixture are tabulated upon start-up, so that the cumbersome process of non-linear minimization is avoided throughout the normal operation of the program. The chemical equilibrium mode is applicable to flows at moderate speeds, where chemical reactions occur at a much faster rate than the traversal rate of the molecules through the domain of interest. The **Arion** code utilizes tabulation for the thermodynamic and molecular properties of the gas (see Section 8.11 for tabulation directives and options). Note, it is required to utilize the eos.table type for the equilibrium gas assumption (see Tables 8.26 and 8.27). In addition, it is required to addd the −−multi.component.gas directive and options (see Section 8.12).

3. *Non-Equilibrium flow*: Under this assumption, finite-rate chemical kinetics models (detailed ahead) are employed to calculate the local composition of the mixture from the solution of additional mass conservation equations for the individual components. This is the most accurate (and expensive) method for calculating the local composition of the gas. It should be used in high-speed flows of air, and in other cases where finite-rate chemistry effects are of interest (*e.g.*, combustion).

2.3.6.1 Equilibrium Gas Properties Tabulation

The tabulation of the equilibrium gas properties is conducted as described in Section 8.11, Tables 8.41-8.60. Note that viscosity, conductivity and heat capacity are also tabulated, based on the selected thermo/transport models.

2.3.6.2 Finite-rate Chemical Kinetics

A reaction mechanism composed of N_r reversible reactions may be generally described by the following chemical formulae:

$$\sum_{s=1}^{N} \nu'_{s,q} M_s \underset{c_{r,q}}{\overset{c_{f,q}}{\rightleftharpoons}} \sum_{s=1}^{N} \nu''_{s,q} M_s \quad \forall \;\; q = 1 \dots N_r \tag{2.31}$$

where $\nu'_{s,q}$ and $\nu''_{s,q}$ are the stoichiometric coefficients of species s (represented by the chemical symbol M_s), acting as a reactant or product in reaction q, respectively. The code currently assumes laminar chemistry, according to which the Arrhenius approach is used to describe the mean reaction rates (*i.e.*, based only on mean values of temperature and species mass fractions). The formula for the mean reaction rate for species s, S_s, appearing as a source-term in the respective species mass conservation equation, is then given by:

$$S_s = W_s \sum_{q=1}^{N_r} \left(\nu''_{s,q} - \nu'_{s,q}\right) \left[c_{f,q} \prod_{p=1}^{N} \left(\frac{\rho Y_p}{W_p}\right)^{\nu'_{p,q}} - c_{r,q} \prod_{p=1}^{N} \left(\frac{\rho Y_p}{W_p}\right)^{\nu''_{p,q}} \right] \tag{2.32}$$

where $c_{f,q}$ and $c_{r,q}$ are the forward and reverse rate coefficients of reaction q, respectively.

2.3.6.3 Reaction Rate Coefficients

The forward rate coefficients are expressed in Arrhenius form as:

$$c_q(T_c) = A_q T_c^{\gamma_q} e^{-(E_q/R_0 T_c)} \tag{2.33}$$

where γ_q and A_q are constants, E_q is the activation energy, and T_c is a controlling temperature to be defined later. Note that the default units of the Arrhenius coefficient and the activation energy are $[A] = (m^3/kmol)^{n-1}/sec$ and $[E] = (J/kmol/K)$, respectively. Other units may be specified via a "UNITS" directive at the beginning of the reaction rates file. The reverse rate coefficients may either be given explicitly in Arrhenius form, for instance, as in Dunn and Kang's 5-species, 11-reactions model [11], or they may be calculated via the equilibrium constant of the reaction, as follows:

$$c_{r,q}(T_{bc}) = \frac{c_{f,q}(T_{bc})}{K_{eq,q}(T_{bc})} \tag{2.34}$$

Where $K_{eq,q}$ denotes the equilibrium constant of reaction q, and T_{bc} is the reverse rate controlling temperature that is not necessarily equal to the forward controlling temperature, T_c. The equilibrium constant, K_{eq}, itself may be calculated in two methods: First, for air only, the constants may be approximated with Park's curve-fits [12] of the local number density. For a general gas, the equilibrium constant of reaction q may only be calculated from Gibbs' free energy,

$$K_{eq}(T) = \left(\frac{10^5}{R_0 T}\right)^{\nu_q} \exp\left[-\sum_s \left(\nu''_{s,q} - \nu'_{s,q}\right)\left(\frac{h_s}{R_s T} - \frac{s_s}{R_s}\right)\right] \tag{2.35}$$

where $\nu_q = \sum_s \left(\nu''_{s,q} - \nu'_{s,q}\right)$, and the dimensionless enthalpy and entropy are obtained from curve-fits, as described in section 2.3.4.

2.4 Gas Selection

2.4.1 Single Component Perfect Gas

As mentioned above, the **Arion** code provides the means to simulate the flow of any perfect gas. This is facilitated via a series of directives that allow to set the specific gas constant, (R, see Table 8.28), the the heat capacity ratio, (γ, see Table 8.29, and the Sutherland formulae coefficients as described in Section 2.2.4.

2.4.2 Multi Component Gas

In the case of a multi-component gas, the actual composition of the gas has to be defined using the appropriate directives. The reader is referred to Section 8.12 and the tables therein.

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 **Arion** code currently contains only one turbulence model, a two-equation model.

3.1 RANS Turbulence Model Equations

The **Arion** code 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):

$$\frac{\partial Q}{\partial t} + \frac{\partial\left(E_c - E_d\right)}{\partial x} + \frac{\partial\left(F_c - F_d\right)}{\partial y} + \frac{\partial\left(G_c - G_d\right)}{\partial z} = S \tag{3.1}$$

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 (3.1) 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 $(\ ^{\prime\prime})$ 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:

$$Q = \begin{bmatrix} \bar{\rho} \\ \bar{\rho}\tilde{u} \\ \bar{\rho}\tilde{v} \\ \bar{\rho}\tilde{w} \\ \tilde{E} \\ \bar{\rho}q_1 \\ \dots \\ \bar{\rho}q_m \end{bmatrix}, \qquad S = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ s1 \\ \dots \\ s_m \end{bmatrix} \tag{3.2}$$

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:

$$E_c = \begin{bmatrix} \bar{\rho}\tilde{u} \\ \bar{\rho}\tilde{u}^2 + \bar{p} \\ \bar{\rho}\tilde{u}\tilde{v} \\ \bar{\rho}\tilde{u}\tilde{w} \\ \tilde{u}\left(\tilde{E} + \bar{p}\right) \\ \bar{\rho}\tilde{u}q_1 \\ \dots \\ \bar{\rho}\tilde{u}q_m \end{bmatrix}, \quad F_c = \begin{bmatrix} \bar{\rho}\tilde{v} \\ \bar{\rho}\tilde{u}\tilde{v} \\ \bar{\rho}\tilde{v}^2 + \bar{p} \\ \bar{\rho}\tilde{v}\tilde{w} \\ \tilde{v}\left(\tilde{E} + \bar{p}\right) \\ \bar{\rho}\tilde{v}q_1 \\ \dots \\ \bar{\rho}\tilde{v}q_m \end{bmatrix}, \quad G_c = \begin{bmatrix} \bar{\rho}\tilde{w} \\ \bar{\rho}\tilde{u}\tilde{w} \\ \bar{\rho}\tilde{v}\tilde{w} \\ \bar{\rho}\tilde{w}^2 + \bar{p} \\ \tilde{w}\left(\tilde{E} + \bar{p}\right) \\ \bar{\rho}\tilde{w}q_1 \\ \dots \\ \bar{\rho}\tilde{w}q_m \end{bmatrix} \tag{3.3}$$

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

$$E_d = \begin{bmatrix} 0 \\ \bar{\tau}_{xx} - \overline{\rho u'' u''} \\ \bar{\tau}_{yx} - \overline{\rho v'' u''} \\ \bar{\tau}_{zx} - \overline{\rho w'' u''} \\ \bar{\beta}_x \\ e_{d_1} \\ \dots \\ e_{d_m} \end{bmatrix}, \quad F_d = \begin{bmatrix} 0 \\ \bar{\tau}_{xy} - \overline{\rho u'' v''} \\ \bar{\tau}_{yy} - \overline{\rho v'' v''} \\ \bar{\tau}_{zy} - \overline{\rho w'' v''} \\ \bar{\beta}_y \\ f_{d_1} \\ \dots \\ f_{d_m} \end{bmatrix}, \quad G_d = \begin{bmatrix} 0 \\ \bar{\tau}_{xz} - \overline{\rho u'' w''} \\ \bar{\tau}_{yz} - \overline{\rho v'' w''} \\ \bar{\tau}_{zz} - \overline{\rho w'' w''} \\ \bar{\beta}_z \\ g_{d_1} \\ \dots \\ g_{d_m} \end{bmatrix} \tag{3.4}$$

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

$$\begin{aligned}
\bar{\beta}_x &= \tilde{u}\left(\bar{\tau}_{xx} - \overline{\rho u'' u''}\right) + \tilde{v}\left(\bar{\tau}_{xy} - \overline{\rho u'' v''}\right) + \tilde{w}\left(\bar{\tau}_{xz} - \overline{\rho u'' w''}\right) + \left(\bar{\kappa} + \bar{\kappa}_t\right)\tfrac{\partial \bar{T}}{\partial x} \\
\bar{\beta}_y &= \tilde{u}\left(\bar{\tau}_{yx} - \overline{\rho v'' u''}\right) + \tilde{v}\left(\bar{\tau}_{yy} - \overline{\rho v'' v''}\right) + \tilde{w}\left(\bar{\tau}_{yz} - \overline{\rho v'' w''}\right) + \left(\bar{\kappa} + \bar{\kappa}_t\right)\tfrac{\partial \bar{T}}{\partial y} \\
\bar{\beta}_z &= \tilde{u}\left(\bar{\tau}_{zx} - \overline{\rho w'' u''}\right) + \tilde{v}\left(\bar{\tau}_{zy} - \overline{\rho w'' v''}\right) + \tilde{w}\left(\bar{\tau}_{zz} - \overline{\rho w'' w''}\right) + \left(\bar{\kappa} + \bar{\kappa}_t\right)\tfrac{\partial \bar{T}}{\partial z}
\end{aligned} \tag{3.5}$$

The terms $\bar{\kappa}$ and $\bar{\kappa}_t$ are the averaged molecular and turbulent heat conductivities, respectively. The molecular heat conductivity is calculated using Sutherland's law (see Equation (2.12)) while the turbulent heat conductivity is calculated using

$$\bar{\kappa}_t = \frac{c_p \bar{\mu}_t}{Pr_t} \tag{3.6}$$

where $\bar{\mu}_t$ 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.

3.2 The Spalart Allmaras Model

The one-equation SA model was originally published in 1992 [13]. 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 [14] 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* [15] with a slight modification as proposed in the work by Rumsey *et al* [16].

The second variant is denoted as the *SA-2020* model. Both models were desgined to overcome the above mentioned anomalies. The Spalart-Allmaras model that is implemented in **Arion** is the *SA-2020* variant of the model.

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

$$\frac{\partial \rho\tilde{\nu}}{\partial t} + \frac{\partial \rho\tilde{\nu}u_k}{\partial x_k} = \frac{\partial}{\partial x_k}\left[\frac{1}{\sigma}\left(\mu + \rho\tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial x_k}\right] + P_{SA} - D_{SA} + SD_{SA} + C_{SA} \tag{3.7}$$

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:

$$P_{SA} = c_{b_1}\tilde{S}\rho\tilde{\nu} \tag{3.8}$$

and

$$D_{SA} = c_{w_1} f_w \rho \left(\frac{\tilde{\nu}}{d}\right)^2 \tag{3.9}$$

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

$$f_w = g\left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right)^{1/6} \tag{3.10}$$

where

$$g = r + c_{w2}\left(r^6 - r\right) \tag{3.11}$$

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

$$SD_{SA} = \rho\frac{c_{b_2}}{\sigma}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k} \tag{3.12}$$

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

The turbulent viscosity, μ_t, is defined as:

$$\mu_t = \rho \tilde{\nu} f_{v1} \tag{3.13}$$

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

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{3.14}$$

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

$$\chi \equiv \frac{\tilde{\nu}}{\nu} \tag{3.15}$$

The model constants are given as follows:

$$\begin{aligned} c_{b1} = 0.1355, \quad c_{b2} = 0.622, \quad \kappa = 0.41, \quad \sigma = 2/3 \\ c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma}, \quad c_{w2} = 0.3, \quad c_{w3} = 2, \quad c_{v1} = 7.1 \end{aligned} \tag{3.16}$$

3.2.1 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

$$\Omega_s = \sqrt{0.5\left(2 s_{ij} s_{ij} + 2\omega_{ij}\omega_{ij}\right)} \tag{3.17}$$

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

$$\bar{S} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \tag{3.18}$$

where the function f_{v2} is defined as:

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{3.19}$$

The parameter r is defined as:

$$r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\right) \tag{3.20}$$

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

$$C_{SA} = -\frac{1}{\sigma}\left(\nu + \tilde{\nu}\right)\frac{\partial \rho}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k} \tag{3.21}$$

3.2.2 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:

$$\tilde{\nu}_{wall} = 0 \tag{3.22}$$

The recommended far field inlet boundary condition of $\tilde{\nu}$, denoted by $\tilde{\nu}_\infty$, for external flows is given by: [17]:

$$\tilde{\nu}_\infty = 3\nu_\infty \; to \; 5\nu_\infty \tag{3.23}$$

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.

3.3 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. The source term of the model is given by:

$$S = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ P_k - \beta_k \bar{\rho} k \omega \\ \alpha_\omega \frac{\omega}{k} P_k - \beta_\omega \bar{\rho} \omega^2 + max(\mathcal{E}, 0) \end{array} \right\} \tag{3.24}$$

The production term is denoted by P_k and it is based on the Boussinesq approximation while $\mathcal{E}$ is the cross diffusion term. The turbulent viscosity is defined as

$$\mu_t = \frac{\overline{\rho} k}{\omega} \tag{3.25}$$

The remaining model constants are $\sigma_k = 1.5$, $\sigma_\omega = 2.0$, $\sigma_d = 0.5$, $\beta_\omega = 0.075$, $\beta_k = 0.09$, $\alpha_\omega = \frac{\beta}{\beta^*} - \frac{\sigma_\omega \kappa^2}{\sqrt{\beta^*}}$, with $\kappa = 0.41$.
A description of the boundary conditions appears in Section 3.5.

3.4 The k-ω-SST Model

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. 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. On the other hand, 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 [18] 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. Again, 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.

3.4.1 k-ω-SST-2003 Model

The transport form of the compressible k-ω-SST-2003 turbulence model contains two transport equations. A transport equation for the turbulence kinetic energy, k, and a second transport equation for the turbulent specific dissipation rate, ω. These transport equations take the following form:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_j k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_k \mu_t\right)\frac{\partial k}{\partial x_j}\right] + P_k - \beta^* \rho \omega k \tag{3.26}$$

$$\begin{aligned}\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_\omega \mu_t\right)\frac{\partial \omega}{\partial x_j}\right] + \frac{\alpha_\omega}{\mu_t}\rho P_k - \beta \rho \omega^2 \\ &+ 2\left(1 - F_1\right)\frac{\rho \sigma_{\omega 2}}{\omega}\max\left(\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 0\right)\end{aligned} \tag{3.27}$$

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, denoted by P_k, is defined as

$$P_k = \mathfrak{R}_{ij} \frac{\partial u_i}{\partial x_j} \tag{3.28}$$

where $\mathfrak{R}_{ij} = \overline{\rho u_i'' u_j''}$ are the Reynolds stress tensor components.

The coefficient β^* is a constant equal to $\beta^* = 0.09$. The rest of the model coefficients, ϕ_c=(σ_k, σ_ω, α_ω, β) are blended according to

$$\phi_c = F_1 \phi_1 + (1 - F_1)\, \phi_2 \tag{3.29}$$

where the coefficients ϕ_1 and ϕ_2 are given in Table 3.1.

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

Table 3.1: SST coefficients

The F_1 function, proposed by Menter [18] himself, is given as:

$$F_1 = tanh\left(\mathfrak{z}_1{}^4\right) \tag{3.30}$$

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

$$\mathfrak{z}_1 = min\left[max\left(\frac{\sqrt{k}}{\beta^* \omega d}, \frac{500\mu}{\rho \omega d^2} \right), \frac{4\sigma_{\omega 2} \rho k}{max\left(2\sigma_{\omega 2} \frac{\rho}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 10^{-10} \right) d^2} \right] \tag{3.31}$$

The turbulent viscosity is defined as

$$\mu_t = \frac{a_1 \rho k}{max\left(a_1 \omega, S F_2\right)} \tag{3.32}$$

where S is:

$$S = \sqrt{2 S_{ij} S_{ij}} \tag{3.33}$$

with

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.34}$$

and the function F_2 is defined as follow:

$$F_2 = tanh\left(\mathfrak{z}_2{}^2\right) \tag{3.35}$$

with

$$\mathfrak{z}_2 = max\left(2\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{\omega d^2}\right) \tag{3.36}$$

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

$$P_k = min\left(P_k, 10\beta^*\rho\omega k\right) \tag{3.37}$$

A description of the boundary conditions appears in Section 3.5.

3.4.2 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 [19]. 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, namely it is added to Equation 3.27, resulting in:

$$\begin{aligned}\frac{\partial\rho\omega}{\partial t} + \frac{\partial\rho u_j\omega}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_\omega\mu_t\right)\frac{\partial\omega}{\partial x_j}\right] + \frac{\alpha_\omega}{\mu_t}\rho P_k - \beta\rho\omega^2 \\ &+ 2\left(1 - F_1\right)\frac{\rho\sigma_{\omega 2}}{\omega}\max\left(\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j}, 0\right) + Q_{SAS}\end{aligned} \tag{3.38}$$

where Q_{SAS} is defined as:

$$Q_{SAS} = \max\left[\rho\xi\kappa S^2\left(\frac{L_l}{L_{vk}}\right)^2 - C\frac{2\rho k}{\sigma_\phi}\max\left(\frac{|\nabla\omega|^2}{\omega^2}, \frac{|\nabla k|^2}{k^2}\right), 0\right] \tag{3.39}$$

The modeled turbulence length scale, L_l, is given by

$$L_l = \sqrt{k}/\left(C_\mu^{1/4}\omega\right) \tag{3.40}$$

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

$$L_{vk} = \frac{\kappa S}{\left|\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right)\right|} \tag{3.41}$$

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

3.5 Boundary Conditions for the k-ω-TNT and k-ω-SST 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:

$$k_{wall} = 0 \tag{3.42}$$

$$\omega_{wall} = 10\frac{6\nu}{\beta_1\left(\Delta d_1\right)^2} \tag{3.43}$$

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

$$k_\infty = \frac{3}{2}\left(T_u \cdot U_\infty\right)^2 \tag{3.44}$$

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:

$$\omega_\infty = \frac{\bar{\rho}_\infty k_\infty}{\left(\mu_t\right)_\infty} \tag{3.45}$$

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

$$0.01 < \frac{(\mu_t)_\infty}{(\mu)_\infty} < 1.0 \tag{3.46}$$

3.6 Evaluation of the Reynolds Stress Tensor

The TNT model is a linear eddy viscosity model (LEVM), and therefore the Reynolds stress tensor that is added to the mean flow equations is defined based on the Boussinesq assumption, namely:

$$\mathfrak{R}_{ij} = \mathfrak{R}_{ij}^{LEVM} \tag{3.47}$$

where

$$\mathfrak{R}_{ij}^{LEVM} = 2\mu_t \left(S_{ij} - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \right) - \frac{2}{3}\bar{\rho}k\delta_{ij} \tag{3.48}$$

and

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{3.49}$$

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

3.7 Turbulence-Mean-flow Coupling

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

$$\begin{aligned} \mu &= \bar{\mu}_l + \bar{\mu}_t \\ \kappa &= \bar{\kappa}_l + \frac{C_p\bar{\mu}_t}{Pr_t} \end{aligned} \tag{3.50}$$

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

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 unstructured hybrid 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 number of cell neighbors, $N(i)$, depends on the type of the cell element. For example, a tetrahedron has 4 neighbors and therefore $N(i) = 4$ whereas for a hexahedron it is $N(i) = 6$. The term R_i signifies the residual of the equations. In what follows, the subscript "i" is dropped for compactness of the representation.

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 currently employed HLLC scheme is described in detail.

4.2.1 HLLC

The concept of average-state approximations was introduced by Harten, Lax and van-Leer [20] 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* [21] discussed this limitation, and proposed a modified

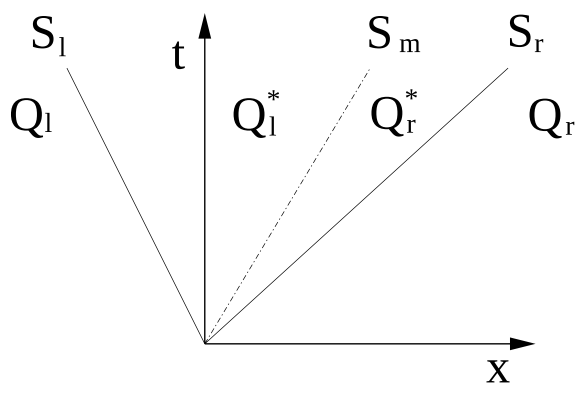


Figure 4.1: Wave structure of the HLLC approximate Riemann solver

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 [22].

The HLLC approximate Riemann solver that is implemented in the **Arion** code is as proposed by Batten *et al* [22]. 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* [23] as follows:

$$S_l = \min[\lambda_{min}, \lambda_{min}^{Roe}] \tag{4.4}$$

$$S_r = \max[\lambda_{max}, \lambda_{max}^{Roe}] \tag{4.5}$$

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 [24], respectively. In the second, the wave speeds S_l and S_r are computed according to Davis [25] as follows:

$$S_l = \min[\lambda_{min}(L), \lambda_{min}(R)] \tag{4.6}$$

$$S_r = \max[\lambda_{max}(L), \lambda_{max}(R)] \tag{4.7}$$

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

$$q = (u - u_g)\, n_x + (v - v_g)\, n_y + (w - w_g)\, n_z \tag{4.8}$$

The contact discontinuity speed S_m is evaluated according to Batten *et al* [22] by

$$S_m = \frac{\rho_r q_r (S_r - q_r) - \rho_l q_l (S_l - q_l) + p_l - p_r}{\rho_r (S_r - q_r) - \rho_l (S_l - q_l)} \tag{4.9}$$

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

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

Introducing the intermediate left state vector

$$\mathbf{Q}_l^* = \begin{Bmatrix} \rho_l^* \\ (\rho u)_l^* \\ (\rho v)_l^* \\ (\rho w)_l^* \\ E_l^* \end{Bmatrix} = \Omega_l \begin{Bmatrix} \rho_l (S_l - q_l) \\ (S_l - q_l)(\rho u)_l + (p^* - p_l)\, n_x \\ (S_l - q_l)(\rho v)_l + (p^* - p_l)\, n_y \\ (S_l - q_l)(\rho w)_l + (p^* - p_l)\, n_z \\ (S_l - q_l)\, E_l - p_l q_l + p^* S_m \end{Bmatrix} \tag{4.11}$$

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

$$\mathbf{H}^*_{c_l} \equiv \mathbf{H}_c\left(\mathbf{Q}^*_l\right) = \mathbf{Q}^*_l S_m + \left\{ \begin{matrix} 0 \\ p^* n_x \\ p^* n_y \\ p^* n_z \\ p^* S_m \end{matrix} \right\} \tag{4.12}$$

and the corresponding intermediate right state vector and right flux vector are obtained from Equations (4.11), (4.12) by interchanging the subscripts $l \rightarrow r$. Finally, the numerical HLLC flux is defined as follow

$$\mathbf{H}_c\left(\mathbf{Q}_l, \mathbf{Q}_r\right) = \left\{ \begin{matrix} \mathbf{H}_c\left(\mathbf{Q}_l\right) & if\ S_l > 0 \\ \mathbf{H}_c\left(\mathbf{Q}^*_l\right) & if\ S_l \leq 0 < S_m \\ \mathbf{H}_c\left(\mathbf{Q}^*_r\right) & if\ S_m \leq 0 \leq S_r \\ \mathbf{H}_c\left(\mathbf{Q}_r\right) & if\ S_r < 0 \end{matrix} \right\} \tag{4.13}$$

where $\mathbf{H}_c\left(\mathbf{Q}_l\right)$ and $\mathbf{H}_c\left(\mathbf{Q}_r\right)$ 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 [26]. 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$^+$ [27]. 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 [28] extended for all speed flows. The AUSM$^+$-up is implemented in the **Arion** code and it is given as follows:

$$\mathbf{F}_c\left(\mathbf{Q}_l,\mathbf{Q}_r\right) = \mathbf{p}_{1/2} + \dot{m}_{1/2}\left\{\begin{array}{ll} \boldsymbol{\psi}_l & \text{if } M_{1/2} > 0 \\ \boldsymbol{\psi}_r & \text{otherwise} \end{array}\right\} \tag{4.14}$$

where

$$\boldsymbol{\psi}_{l/r} = \left\{\begin{array}{c} 1 \\ u_{l/r} \\ v_{l/r} \\ w_{l/r} \\ H_{l/r} \end{array}\right\}, \tag{4.15}$$

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

$$\dot{m}_{1/2} = a_{1/2}M_{1/2}\left\{\begin{array}{ll} \rho_l & \text{if } M_{1/2} > 0 \\ \rho_r & \text{otherwise} \end{array},\right\} \tag{4.16}$$

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

$$\mathbf{p}_{1/2} = \begin{Bmatrix} 0 \\ p_{1/2}n_x \\ p_{1/2}n_y \\ p_{1/2}n_z \\ 0 \end{Bmatrix} \tag{4.17}$$

where

$$p_{1/2} = \mathcal{P}^{+}_{(5)}(M_l)p_l + \mathcal{P}^{-}_{(5)}(M_r)p_r - K_u \mathcal{P}^{+}_{(5)}(M_l)\mathcal{P}^{-}_{(5)}(M_r)(\rho_l + \rho_r)(f_a a_{1/2})(q_r - q_l) \tag{4.18}$$

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. The remaining functions are given below. The left/right Mach number at the interface, $M_{l/r}$, is defined as follows:

$$M_{l/r} = \frac{q_{l/r}}{a_{1/2}} \tag{4.19}$$

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:

$$a_{1/2} = \frac{a_l + a_r}{2} \tag{4.20}$$

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

$$\begin{aligned}
\overline{M}^2 &= \frac{q_l^2 + q_r^2}{2a_{1/2}^2} \\
M_o^2 &= min\left(1, max\left(\overline{M}^2, M_\infty^2\right)\right) \\
f_a\left(M_o\right) &= M_o\left(2 - M_o\right) \\
\rho_{1/2} &= \frac{\rho_l + \rho_r}{2} \\
M_{1/2} &= \mathcal{M}^{+}_{(4)}\left(M_l\right) + \mathcal{M}^{-}_{(4)}\left(M_r\right) - \frac{K_p}{f_a} max\left(1 - \sigma\overline{M}^2, 0\right)\frac{p_r - p_l}{\rho_{1/2}a_{1/2}^2}
\end{aligned} \tag{4.21}$$

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:

$$\begin{aligned}
\mathcal{M}_1^{\pm} &= \frac{1}{2}\left(M \pm |M|\right) \\
\mathcal{M}_2^{\pm} &= \pm\frac{1}{4}\left(M \pm 1\right)^2 \\
\mathcal{M}_{(4)}^{\pm}(M) &= \begin{Bmatrix} \mathcal{M}_{(1)}^{\pm} & if\ |M| > 0 \\ \mathcal{M}_{(2)}^{\pm}(1 \mp 16\beta\mathcal{M}_{(2)}^{\mp}) & otherwise \end{Bmatrix}
\end{aligned} \tag{4.22}$$

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

$$\mathcal{P}_{(5)}^{\pm}(M) = \begin{Bmatrix} \frac{1}{M}\mathcal{M}_{(1)}^{\pm} & if\ |M| \geq 1 \\ \mathcal{M}_{(2)}^{\pm}[(\pm 2 - M) \mp 16\alpha M \mathcal{M}_{(2)}^{\mp}) & otherwise \end{Bmatrix} \tag{4.23}$$

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

4.2.3 Passive Scalar Approach

The **Arion** code implements the passive scalar approach [22] 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:

$$\mathbf{Q}_l^* = \begin{Bmatrix} \rho_l^* \\ (\rho u)_l^* \\ (\rho v)_l^* \\ (\rho w)_l^* \\ E_l^* \\ (\rho k)^* \\ (\rho\phi)^* \end{Bmatrix} = \Omega_l \begin{Bmatrix} \rho_l\left(S_l - q_l\right) \\ \left(S_l - q_l\right)(\rho u)_l + \left(p^* - p_l\right)n_x \\ \left(S_l - q_l\right)(\rho v)_l + \left(p^* - p_l\right)n_y \\ \left(S_l - q_l\right)(\rho w)_l + \left(p^* - p_l\right)n_z \\ \left(S_l - q_l\right)E_l - p_l q_l + p^* S_m \\ \rho k \\ \rho\phi \end{Bmatrix} \tag{4.24}$$

The HLLC inviscid flux is then easily evaluated using:

$$\mathbf{H}^*_{c_l} \equiv \mathbf{H}_c\left(\mathbf{Q}^*_l\right) = \mathbf{Q}^*_l S_m + \begin{Bmatrix} 0 \\ p^* n_x \\ p^* n_y \\ p^* n_z \\ p^* S_m \\ 0 \\ 0 \end{Bmatrix} \tag{4.25}$$

4.2.4 High Order Flux Approximations

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. To obtain a higher order flux approximation, the left and right state vectors of the convective flux are evaluated using a linear reconstruction using Green's theorem or a Taylor series expansion and least squares method. A cell-wise gradient of the primitive variables is constructed, followed by a second order Taylor series expansion, the left and right states are reconstructed. Let the vector $\boldsymbol{W} = (W_m; m = 1, ...7)$ denote the primitive variables vector (for a general $k - \omega$ turbulence model),

$$\boldsymbol{W} = [\bar{\rho}, \widetilde{u}, \widetilde{v}, \widetilde{w}, \bar{p}, k, \omega] \tag{4.26}$$

then the left and right primitive variables are reconstructed as follows:

$$\begin{aligned} (W_m)_L &= (W_m)_i + (\psi_m)_i \left(\nabla W_m\right)_i \cdot \boldsymbol{d}_i^{ij} && \text{(4.27a)} \\ (W_m)_R &= (W_m)_j + (\psi_m)_j \left(\nabla W_m\right)_j \cdot \boldsymbol{d}_j^{ij} && \text{(4.27b)} \end{aligned}$$

where $\boldsymbol{d}_i^{ij}$ $\left(\boldsymbol{d}_j^{ij}\right)$ is the distance vector from the mid-point of face ij to the center of cell i (j), and ψ_m is the cell limiter that is used to suppress oscillations in the solution.

4.3 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. The vector $\mathcal{W}_{ij}$ is calculated by averaging of adjacent cell-center values (*i.e.*, $\mathcal{W}_i$ and $\mathcal{W}_j$).

4.4 Time Marching Schemes

The **Arion** code provides various possibilities for advancing Equation (4.3) in time. This section contains a brief description of the available schemes. 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
3. Multi-stage implicit schemes
   (a) Third fourth and fifth order Runge-Kutta implicit schemes

4.4.1 Explicit schemes

4.4.1.1 Explicit Euler Scheme

Consider the semi-discrete equation[1]:

$$V\frac{dQ}{dt} = R \tag{4.28}$$

[1]Equation (4.3) without the index

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

$$\Delta Q^n = \frac{\Delta t}{V} R^n \tag{4.29}$$

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

$$\Delta Q^n = Q^{n+1} - Q^n \tag{4.30}$$

4.4.1.2 Runge-Kutta Schemes

Arion provides the choice of third or fourth order Runge-Kutta schemes. Consider the semi-discrete formulation as presented in Equation (4.28), the Runge-Kutta scheme formulation is as follows:

$$\begin{aligned} Q^{(0)} &= Q^n \\ Q^{(k)} &= Q^n + \alpha_k \frac{\Delta t}{V} R^{(k-1)}, \quad k = 1, \ldots, K \\ Q^{n+1} &= Q^{(K)} \end{aligned} \tag{4.31}$$

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.

4.4.1.3 Time-Accurate Third Order Runge-Kutta TVD

Arion provides the capability to use the third order, time-accurate, Runge-Kutta TVD scheme, defined as:

$$\begin{aligned} Q^{(1)} &= Q^n + \frac{\Delta t}{V} R^n, \\ Q^{(2)} &= \frac{3}{4} Q^n + \frac{1}{4} Q^{(1)} + \frac{1}{4} \frac{\Delta t}{V} R^{(1)}, \\ Q^{n+1} &= \frac{1}{3} Q^n + \frac{2}{3} Q^{(2)} + \frac{2}{3} \frac{\Delta t}{V} R^{(2)} \end{aligned} \tag{4.32}$$

4.4.2 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:

$$\left(\frac{V}{\Delta t}I - \frac{\partial R}{\partial Q}\right)^n \Delta Q^n = R^n \tag{4.33}$$

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.

Applying Equation (4.33) 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.

4.4.2.1 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:

$$\frac{\partial R}{\partial Q}\Delta Q \approx \left(\frac{\partial R}{\partial Q}\right)^n \Delta Q^n \tag{4.34}$$

Consequently, they can be moved to the right hand side.

4.4.2.2 Runge-Kutta Schemes

Runge-Kutta implicit schemes combine the explicit Runge-Kutta schemes as described in Section 4.4.1.2 with the PGS scheme that is described in Section 4.4.2.1 to form a robust implicit, multi-stage time marching scheme. note that this scheme requires the matrix inversions warranted by the PGS scheme at each stage and therefore requires more computer time per iteration.

4.5 Convergence Acceleration Methods

4.5.1 Krylov Methods

Krylov subspace methods were originally introduced as direct methods [29] for solving linear systems, but only gained popularity after being reintroduced as iterative methods [30]. Krylov methods are actually projection (Galerkin) methods for solving a linear set of equations $Ax = b$ using the Krylov subspace, K_j, spanned by:

$$K_j = span\left(r_0, Ar_0, A^2r_0, ..., A^{j-1}r_0\right) \tag{4.35}$$

where $r_0 = b - Ax_0$. The main advantage of Krylov methods for large systems of equations, lies in the fact that they require only matrix-vector products to carry out the iteration, and not the individual elements of A. This also makes them especially suitable for use with Newton's method.

The widely used Generalized Minimal RESidual (GMRES) Krylov method [31] employs Arnoldi basis vectors to form the trial subspace out of which the solution is constructed. One matrix-vector product is required per iteration to create each new trial vector, and the iterations are terminated based on an estimate of the residual that does not require explicit construction of intermediate residual vectors or solutions, which constitutes a major benefit of the algorithm. Several studies [32–34] have established the superiority of GMRES over other Krylov methods, especially in a Jacobian-Free Newton-Krylov framework. Consequently, GMRES is used to solve the large equation set arising in each Newton iteration, given by Equation 4.33.

4.5.1.1 Preconditioning

The intensive memory-pressure typically encountered in modern CFD codes has put an increased emphasis on quality preconditioning to complement Krylov methods. In fact, Krylov methods were shown to be efficient for large-scale problem only when coupled with effective preconditioning of the matrix A [32]. The **Arion** code employs the additive Schwartz method in which the action of the pre-conditioner is broken-down to smaller sub-pre-conditioners defined on individual geometric subdomains. The primary motivation for such an approach is its parallel scalability [35, 36]. Furthermore, domain-based parallelism is recognized as the form of data parallelism that most effectively exploits modern processors that possess a multi-level memory hierarchy [37].

Specifically, the **Arion** code makes use of the restricted additive Schwarz method (RASM), which eliminates interprocess communication during either the restriction or prolongation phase of the additive Schwarz technique [38], thus further increasing its suitability for massive-scale, distributed computations. The relatively small subsets of linear equations arising in the process of applying the preconditioner are solved with a local, block incomplete lower-upper (ILU) decomposition method.

4.5.1.2 Portable, Extensible Toolkit for Scientific Computation (PETSc)

PETSc, the Portable, Extensible Toolkit for Scientific Computation is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. PETSc is developed as open-source. Among the various solution methods that are available by the PETSc Toolkit, the **Arion** code utilizes certain Krylov solvers. The reader is referred to Section 8.16 for the PETSc directives that are supported by the **Arion** code.

4.5.2 Line Search

The **Arion** code uses the back-traced lines search algorithm as briefly described in this section. Let a general implicit iterative scheme be defined as:

$$\left(\frac{V}{\Delta t} - J^n\right)\Delta Q^n = R^n \tag{4.36}$$

where V is the cell volume, ΔQ^n is the vector of dependent variables correction term at iteration n, R is the residual vector, and J is the Jacobian matrix defined as:

$$J \triangleq \frac{\partial R}{\partial Q} \tag{4.37}$$

A relaxed updated solution may be given by

$$Q^{n+1} = Q^n + \alpha \Delta Q^n \tag{4.38}$$

where α is relaxation parameter. The relaxation parameter can manually prescribed or optimally calculated using a back-traced line search algorithm [39].

4.5.2.1 Back-Traced Line Search

Let the function $f(\alpha)$ be defined as

$$f(\alpha) = \frac{1}{2} R^T R \tag{4.39}$$

Consequently,

$$\begin{aligned} f(0) &= \frac{1}{2} R^{T^n} R^n \\ f(1) &= \frac{1}{2} R^{T^{n+1}} R^{n+1} \\ f'(0) &= R^{T^n} J^n \Delta Q^n \end{aligned} \tag{4.40}$$

Let the polynomial function satisfying the conditions in Equation 4.40 be defined as:

$$f(\alpha) = a\alpha^2 + b\alpha + c \tag{4.41}$$

The optimal relaxation parameter is such that minimizes the function f.

The algorithm that is implemented in the **Arion** code is as follows:

1. Solve the linear system:
$$J(Q^n)\,\Delta Q^n = -R(R^n)$$
2. Calculate the new residual:
$$R(Q^n + \Delta Q^n)$$
3. Check if the convergence is satisfactory:
$$f(Q^n + \Delta Q^n) < f(Q^n) + \beta\alpha\nabla f(Q^n)^T \Delta Q^n$$
4. If convergence is satisfactory, move on to the next Newton iteration. If not perform a line search:
$$\alpha|\min_{\alpha} f(Q^n + \alpha\Delta Q^n)$$
5. Repeat (go to 3) until the new residual is good enough or the maximum number of line search iterations is iteration reached or the minimum value of α is obtained.

The application of the lines search algorithm in the **Arion** code is in addition to the Krylov methods (see Section 4.5.1). It is invoked through a directive (or a series thereof) to set the number of line search iterations, the convergence criterion constant, β, etc. The reader is referred to Section 8.15.1 for all of the relevant directives.

Chapter 5

Boundary Conditions

5.1 Introduction

The **Arion** code contains a wide variety of boundary conditions. Being an unstructured finite volume code, the notion of a ghost cell is utilized throughout. However, in certain cases the Jacobian and flux are explicitly dictated rather than calculated based on the ghost. In particular, the convection Jacobian and flux. 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.

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:

$$\begin{bmatrix} n_x & n_y & n_z \\ t^1_x & t^1_y & t^1_z \\ t^2_x & t^2_y & t^2_z \end{bmatrix} \begin{pmatrix} u_g \\ v_g \\ w_g \end{pmatrix} = \begin{pmatrix} 2\overline{V}_f \cdot \hat{n} - \overline{V}_r \cdot \hat{n} \\ \overline{V}_r \cdot \hat{t}^1 \\ \overline{V}_r \cdot \hat{t}^2 \end{pmatrix} \tag{5.1}$$

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

5.2.2 No-Slip Condition

The no-slip condition is much easier to implement:

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

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.

5.3 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

$$\begin{aligned}\rho_g &= \rho_\infty && (5.5)\\ \overline{V}_g &= \overline{V}_\infty && (5.6)\\ p_g &= \frac{\rho_\infty}{\rho_r}p_r && (5.7)\end{aligned}$$

5.3.1.2 Turkel Outlet

$$\begin{aligned}\rho_g &= \rho_r + \frac{p_\infty - p}{a_\infty^2} && (5.8)\\ \overline{V}_g &= \overline{V}_r && (5.9)\\ p_g &= p_\infty && (5.10)\end{aligned}$$

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:

$$\begin{aligned}R^+ &= q_r - \frac{2}{\gamma-1}a_\infty\\ R^- &= q_\infty - \frac{2}{\gamma-1}a_r && (5.11)\end{aligned}$$

The ghost Riemann invariants are obtained using:

$$R_g = \frac{1}{2}\left(R^+ + R^-\right) \qquad (5.12)$$

5.3.2.1 Riemann Inlet

$$\begin{aligned}\rho_g &= \frac{\rho_\infty}{a_\infty^{\frac{1}{\gamma-1}}}\left\{\underbrace{\left[\frac{\gamma-1}{4}\left(R^- - R^+\right)\right]^{\frac{1}{\gamma-1}}}_{a_g}\right\}^2\\ p_g &= \sqrt{a_g}\rho_g\gamma && (5.13)\end{aligned}$$

5.3.2.2 Riemann Outlet

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

$$\begin{aligned} s &= \frac{\rho^\gamma}{\gamma p} \\ \rho_g &= \left(a_g^2 s\right)^{\frac{1}{\gamma-1}} \\ p_g &= \sqrt{a_g}\rho_g\gamma \end{aligned} \tag{5.14}$$

5.3.3 Fixed (Supersonic Inlet)

$$\begin{aligned} \overline{V}_g &= \overline{V}_\infty \\ p_g &= p_\infty \\ T_g &= T_\infty \end{aligned} \tag{5.15}$$

5.3.4 Extrapolation (Supersonic Outlet)

$$\begin{aligned} \overline{V}_g &= \overline{V}_r \\ p_g &= p_r \\ T_g &= T_r \end{aligned} \tag{5.16}$$

5.3.5 Inlet

For supersonic inlet, a fixed boundary condition is applied (see Section 5.3.3). For subsonic inlet, the pressure is extrapolated from within the physical domain and the rest of the flow variables are fixed. This is automatically repeated every time step.

$$\begin{aligned} (\rho, u, v, w)_g &= (\rho, u, v, w)_\infty \\ p_g &= p_r \end{aligned} \tag{5.17}$$

5.3.6 Outlet

For supersonic outlet, an extrapolated boundary condition is applied (see Section 5.3.4). For subsonic outlet, a "Fixed," prescribed value is used for the pressure and the rest of the flow variables are extrapolated. This is automatically repeated every time step. Formally, this is a pressure outlet type boundary condition.

$$\begin{aligned} (\rho,u,v,w)_g &= (\rho,u,v,w)_r \\ p_g &= p_{bc} \end{aligned} \tag{5.18}$$

where p_{bc} is the prescribed pressure outlet. If no pressure outlet is prescribed then $p_{bc} \equiv p_\infty$.

5.3.6.1 Prescribed Mass Flow Rate

To obtain the prescribed mass flow rate at an outlet, the pressure is iteratively corrected in the following manner:

$$p_g = p_{bc} + \delta p \tag{5.19}$$

where p_{bc} is the prescribed pressure outlet (see Table 8.82). The pressure correction, δp, is calculated as follows:

$$\delta p_{new} = \omega \frac{|\dot{m}|^n \left(\dot{m}^n - \dot{m}_{bc}\right)}{\rho^n_{ave} A^2} + \delta p_{old} \tag{5.20}$$

where $\dot{m}^n$ is the computed mass flow rate at the current iteration, ρ^n_{ave} is the computed average density at the current iteration, A is the total area of the boundary, and δp_{new} and δp_{old} are the new and previous calculated pressure corrections, respectively. The parameter ω is a relaxation parameter (see Table 8.65). The mass flow rate is used in conjunction with the "outlet" boundary condition (see Table 8.64).

5.3.7 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 :**

$$\begin{aligned}(\rho, u, v, w)_g &= (\rho, u, v, w)_\infty \\ p_g &= p_r \end{aligned} \tag{5.21}$$

- **Outlet :**

$$\begin{aligned}(\rho, u, v, w)_g &= (\rho, u, v, w)_r \\ p_g &= p_\infty \end{aligned} \tag{5.22}$$

5.4 Stagnation Inlet

For subsonic flow, the stagnation inlet boundary condition is implemented as follows. The user prescribes the stagnation temperature and stagnation pressure (see Tables 8.81 and 8.82). Let the negative Riemann invariant be defined as:

$$R^- = q - \frac{2}{\gamma - 1}a \tag{5.23}$$

where $q = \overline{V} \cdot \hat{n}$ is the velocity normal to the face, a is the speed of sound, and γ is the ratio of specific heats.

Applying the Riemann invariance results in

$$R_g^- = q_g - \frac{2}{\gamma - 1}a_g = R_r^- \tag{5.24}$$

The subscript r pertains to the real flow cell, *i.e.*, extrapolated from the interior of the domain, while the subscript g pertains to the value at the ghost cell (boundary).

By assuming a calorically perfect gas one can write on the boundary

$$c_pT_0 = c_pT_g + \frac{q_g^2}{2} \tag{5.25}$$

Consequently, the speed of sound at the ghost cell can be evaluated using

$$a_g = \sqrt{(\gamma - 1)\, c_p \left(T_0 - \frac{q_g^2}{2c_p}\right)} \tag{5.26}$$

Substituting Equation 5.26 into Equation 5.24 results in a quadratic equation in q_g. The pressure, density, and temperature at the ghost cell are calculated using isentropic relations.

5.5 Symmetry Boundary Conditions

Symmetric boundary conditions are treated exactly as an adiabatic impermeable wall.

Chapter 6

Computational Mesh

The **Arion** code is considered a hybrid code since it supports various cell element types. The code supports tetrahedra, hexahedra, prism, and pyramids. Mesh generation may be conducted by any unstructured grid generator, however, users must export the mesh using a Star-CD export or a CGNS export.

6.1 Star-CD Export

The Star-CD export results in three files. The first, a file containing the vertex information, having the extension ".vrt." The second, a file containing the cell elements, having the extension ".cel." And finally, a file containing the boundary elements (triangles or quads), having the extension ".bnd." The ".bnd" file contains the boundary elements as well as a name for each element. The naming is conducted by the user using the grid generation package (*e.g.,* Pointwise). The code supports two variations of the Star-CD format, the export by Pointwise and the export by CENTAUR (only 3-D export is supported).

6.2 CGNS Export

The CGNS export results in a single file. The file extension of the CGNS file is ".cgns."

6.3 Automated Mesh Adaptation

An automated mesh adaptation procedure is available in the **Arion** code. It requires the use of the Pointwise mesh generation package and can be facilitated through the EZOpt optimization software package. A feature based adaptation sensor that is proportional to an interpolation error is utilized. Using a series of options, a point cloud data file is generated and the mesh is re-generated using the previously generated computational mesh and the point cloud data. The method preserves the boundary layer mesh and it maintains smooth mesh size. The sensor function is given by:

$$S = K\left|\vec{h}\right|^{p}\left|\left[\hat{\vec{h}}\cdot\frac{\partial\hat{\Phi}}{\partial x_2} - \hat{\vec{h}}\cdot\frac{\partial\hat{\Phi}}{\partial x_1}\right]\right| \tag{6.1}$$

where $\hat{\vec{h}}$ is the distance vector between adjacent points and Φ is problem dependent. The parameters K and p may be prescribed by the user. Currently the **Arion** code supports sensor functions that depend on the Mach number, pressure, density, and velocity magnitude (Φ). The available adaptation options are described in Section 8.20.

Chapter 7

Parallelization

The **Arion** flow solver is designed to work in a distributed memory architecture using the MPI interface. The design of the code distinguishes between managing a simulation and solving the flow field. Within the distributed MPI universe, the first rank is responsible for managing the simulation and henceforth named 'manager rank'. The rest of the ranks are responsible for the flow solution and are named 'worker ranks'. The manager rank responsibility starts with the input analysis, and continuing with reading the initial geometric problem (the grid files), splitting the computational domain into parts and sending those parts to the worker ranks. Therefore, the worker ranks know only part of the computational domain, and pass boundary data among themselves. The manager rank is responsible for the the assembly of the 'restart' files, for timing all the worker threads, and for log output. Since there is a distinction between the manager rank and the worker ranks, one must have at least two ranks running, even if they reside on a single shared memory machine.

Chapter 8

Input File, Run Preparation, and Execution

8.1 Preface

The **Arion** code is driven through the command line with optionally additional input files having a certain syntax. An input file may be constructed using a simple text editor. The solver is invoked by typing the MPI command (depending on the MPI version of the actual machine): “mpirun [mpi options]... arion [−−f input_file] [command-line arguments]”

The input file is made of groups of directives, each group is marked by two consecutive − signs. Each group has a series of options, with each option marked with one − sign. Within the input file the sign ! means a remark until the end of the line. A brief help of all the options may be printed to the screen by using the −−h option. The latest additions to the code may be printed to the screen using the −−new option.

8.1.1 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 group of directives starts with two consecutive − signs, *i.e.*, −− followed by the group name and a series of options.
- Each option is marked with one − sign.
- An option 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.

8.1.2 Basic Input File: Examples

8.1.2.1 Single Component Gas Example

Listing 8.1 contains a simple, single component gas, basic input file for a simple simulation of the ideal-gas flow about a two-dimensional airfoil.

```
--equation.of.state
   -name air
   -type eos.ideal.gas
   -ref.p 101325
   -ref.T 288

--bc
   -name RIE_FREE
   -type riemann

--bc
   -name IWALL
   -type impermeable.wall

--bc
   -name XZSYM
   -type 2d

--system
   -name ideal.gas.mf.inviscid
   -type ideal.gas.mf.inviscid
   -time.step 5 100 cfl.exponential 250
        -cell.gradient green.gauss.node
        -limiter mlp.2d
   -log convergence cfl residual log.residual

--solve
   -time.march           implicit.rk3.R.pgs
   -convection.flux      hllc.roe
   -convection.jacobian  hllc.roe
   -sweeps 4
   -spatial.order 2

   -iterations  2000
   -save      100
   -save.path ./save/

   -eos air
   -p      101325
   -T      288
   -Mach   0.8
   -alpha  1.25
   -beta   0
   -log convergence iter

--log
   -name convergence
   -prefix ./logs/convergence
   -per iter
   -screen

--plot
   -name naca0012
   -prefix ./plots/plot
   -interval 100
```

```
--uns
       -name naca0012-j129-uns-regular
       -prefix naca0012-j129-uns-regular/naca0012-j129-uns-regular
       -log convergence cl cd

--table
       -name P_table
       -prefix ./tables/table
       -interval 10
       -plane.bc 0 0 0 0 0.5 0 IWALL
```

Listing 8.1: Example of a simple **Arion** input file

8.1.2.2 Multi Component Gas Example

8.2 Grid Files

Currently, **Arion** supports the Star-CD and CGNS Version 3.4 exports only. A detailed description of the files is given in Chapter 6. Starting from Version 1.31, **Arion** uses the Metis open source library to convert the Star-CD or CGNS export to binary format. In addition, Metis assists in decomposing the computational domain into several partitions for the purpose of efficient parallel computations. The following section describes the use of the Metis library for conversion of the files to binary format. The conversion results in a new set of files with "+32" added to all file extensions, signifying that the integers are 32 bit wide.

The new set of files now contains 5 files. The vertex information has the extension".vrt+32." The cell elements file has the extension ".cel+32." The boundary elements file has the extension ".bnd+32." In addition, two new files are generated, one with the extension "nam+32" and the other with the extension "fac+32." As a result, the grid is read by the code in a fast manner.

8.3 Macros

8.3.1 Def

Def is a macros that starts with the directive −− def followed by a key and a value. The syntax is as described in Table 8.1.

Def		
Syntax	−−def $key $value	
Description	Sets a macro. Any '$key' or '${key}' in the options will be replaces by the $value .	
Parameters	$key	Key.
	$value	Value.
Examples	−−def mypath /Users/data/restart/	

Table 8.1: Def macro

8.4 Conversion

The conversion is conducted using the −−star2metis or −−cgns2metis directive as described in Table 8.2. Note that the Metis options that are described in Tables 8.2 and 8.3 are taken directly from the Metis runtime help. For further information the user is referred to the complete Metis User's Manual.

The conversion generates a new set of files, replacing the original files that were generated by the grid generation software package (*e.g.*, Pointwise). The actual files are not split but are reordered in a manner that prepares the files for a parallel read that is efficient in terms of domain decomposition.

Conversion		
Syntax	arion {−−star2metis | −−cgns2metis}$prefix #ranks $output [metis-options]...	
Description	Convert the Star-CD or CGNS files to the Arion binary format and use Metis to decompose the computational domain. See Table 8.3 for Metis options.	
Parameters	star2metis	Convert Star-CD files.
	star2cgns	Convert CGNS files.
	$prefix	Prefix of grid files.
	#ranks	Number of partitions.
	$output	Prefix of output files.
Examples	arion −−star2metis Star-CD-file-name 64 New-file-name	
	arion −−cgns2metis CGNS-file-name 64 New-file-name	

Table 8.2: Conversion

Metis Options

Metis Option	Description
-ptype {rb | kway}	Specifiy the scheme to be used for computing the k-way partitioning: rb - Recursive bisectioning, kway - Direct k-way partitioning [default].
-ctype {rm | shem}	Specify the scheme to be used to match the vertices of the graph; rm - Random matching, shem - Sorted heavy-edge matching [default].
-iptype {grow | random}	Specify the scheme to be used to compute the initial partitioning, (applies only when -ptype = rb); grow - Grow a bisection using a greedy scheme [default], random - Compute a bisection at random.
-objtype {cut | vol}	Specify the objective that the partitioning routines will optimize, (applies only when -ptype = kway); cut - Minimize the edgecut [default]; vol - Minimize the total communication volume.
-no2hop	Specify that the coarsening will not perform any 2-hop matchings when the standard matching fails to sufficiently contract the graph.
-contig	Specify that the partitioning routines should try to produce (applies only when -ptype = kway) partitions that are contiguous. Note that if the input graph is not connected this option is ignored.
-minconn	Specify that the partitioning routines should try to minimize the (applies only when -ptype = kway) maximum degree of the subdomain graph, i.e., the graph in which each partition is a node, and edges connect subdomains with a shared interface.

Table 8.3: Metis options

Metis Options (continued)

Metis Option	Description
-ufactor #x (where x is an integer)	Specify the maximum allowed load imbalance among the partitions. A value of x indicates that the allowed load imbalance is 1+x/1000. For -ptype=rb, the load imbalance is measured as the ratio of the 2*max(left,right)/(left+right), where left and right are the sizes of the respective partitions at each bisection. For -ptype=kway, the load imbalance is measured as the ratio of max_i(pwgts[i])/avgpwgt, where pwgts[i] is the weight of the ith partition and avgpwgt is the sum of the total vertex weights divided by the number of partitions requested. For -ptype=rb, the default value is 1 (i.e., load imbalance of 1.001) For -ptype=kway, the default value is 30 (i.e., load imbalance of 1.03).
-niter #i	Specify the number of iterations for the refinement algorithms at each stage of the uncoarsening process. Default is 10.
-ncuts #c	Specify the number of different partitionings that it will compute The final partitioning is the one that achieves the best edgecut or communication volume. Default is 1.
-seed #s	Select the seed of the random number generator.

Table 8.3: Metis options (continued)

8.4.1 Conversion Between Different Orderings

Arion provides the capabilities to start a simulation from a certain ordering and continue the simulation with a different ordering. This is conducted using the −−cnv2cnv directive as described in Table 8.4.

cnv2cnv		
Syntax	arion −−cnv2cnv $cnv1 $cnv2 $load-prefix $save-prefix {$system1-name}...	
Description	Converts **Arion** solution files between different orderings; creates new restart files.	
Parameters	$cnv1	Ordering type to convert from.
	$cnv2	Ordering type to convert to.
	$load-prefix	Prefix of input files.
	$save-prefix	Prefix of output files.
	$system1-name	Name of converted system.
Examples	arion −−cnv2cnv bin arion ./newsolution/oldfiles ./newsolution/newfiles ./restart/oldfiles	

Table 8.4: Conversion between different ordering (cnv2cnv) option

8.5 Log Control Options

Log control options are used to set up the log outputs. For example, the options are used to set up the log file name. The log control options section starts with the −−log directive, followed by a series of options that are described in the following tables. It is important to note that the number of logs (and thus generated log files) is unlimited.

Name		
Syntax	−name $name	
Description	Set the name of the log. This name is referred to by the specific −log options. It can be the predefined keywords: ‘stdout’, ‘screen’, or ‘display’: where it will be only shown on the screen. It can also be: ‘stderr’ where it will be only directed to the standard error stream.	
Parameters	$name	Log name.
Examples	−name convergence	

Table 8.5: Log name option

Prefix	
Syntax	−prefix $prefix
Description	The log file path and name prefix. The '.log' suffix is added to obtain the actual log file name.
Parameters	$prefix Prefix.
Examples	−prefix ./logs/convergence

Table 8.6: Log prefix option

Per	
Syntax	−per $type
Description	Set when the log is printed based on the $type. The possibilities are: iter, produced every iteration (virtual time step); step, produced every step (physical time step); or pos, produced every 'pos' (depending on the simulation type).
Parameters	$type Type (iter, step, or pos).
Examples	−per pos

Table 8.7: Log per option

Stdout	
Syntax	−stdout
Description	Adds echo to the standard output as well as to the log file. One can also use the -screen or -display (the outcome is exactly the same).
Parameters	N/A
Examples	−screen

Table 8.8: Log stdout option

Stderr	
Syntax	−stderr
Description	Adds echo to the standard error as well as to the log file.
Parameters	N/A
Examples	−stderr

Table 8.9: Log stderr option

8.6 Plot Control Options

Plot control options are used to set up the plot outputs. The plot control options section starts with the −−plot directive, followed by a series of options that are described in the following tables.

Name		
Syntax	−name $name	
Description	Set the name of the plot. This name is referred to by the specific −plot options.	
Parameters	$name	Plot name.
Examples	−name Alpha10	

Table 8.10: Plot name option

Prefix		
Syntax	−prefix $prefix	
Description	The plot file path and name prefix. The actual file number is composed from the prefix, and the $name as described in Table 8.183. The '.fvuns' suffix is added to the plot file name. If the computational domain has been decomposed, the files are split as well and the number of the partition is added to the file name.	
Parameters	$prefix	Prefix.
Examples	−prefix ./plots/	

Table 8.11: Plot prefix option

Interval	
Syntax	−interval #interval
Description	Set the interval (of iterations/steps) to create the plot files.
Parameters	#interval Plot interval.
Examples	−interval 100

Table 8.12: Plot interval option

Sequential/overwrite	
Syntax	−sequential / −overwrite
Description	Include position stamp in the plot files (or overwrite, without position stamp, default).
Parameters	N/A
Examples	−sequential

Table 8.13: Plot sequential/overwrite option

FVUNS/EPPIC/CGNS	
Syntax	−fvuns / −eppic / −cgns
Description	Select the file format to use, default is:-fvuns.
Parameters	N/A
Examples	−eppic −cgns

Table 8.14: Plot fvuns/eppic/cgns option

8.7 Table Control Options

Table control options are used to set up the table outputs. The table control options section starts with the −−table directive, followed by a series of options that are described in the following tables.

Name		
Syntax	−name $name	
Description	Set the name of the table. This name is referred to by the specific −table options.	
Parameters	$name	Table name.
Examples	−name Alpha10	

Table 8.15: Table name option

Prefix		
Syntax	−prefix $prefix	
Description	The table file path and name prefix. The '.tbl' suffix is added to obtain the actual table file name.	
Parameters	$prefix	Prefix.
Examples	−prefix ./tables/surface-pressure	

Table 8.16: Table prefix option

Interval	
Syntax	−interval #interval
Description	Set the interval (of iterations/steps) to create the table files.
Parameters	#interval Table interval.
Examples	−interval 100

Table 8.17: Table interval option

Sequential/overwrite	
Syntax	−sequential / -overwrite
Description	Include position stamp in the table files (or overwrite, without position stamp, default).
Parameters	N/A
Examples	−sequential

Table 8.18: Table sequential/overwrite option

Horizontal/vertical	
Syntax	−horizontal / -vertical
Description	The table type (vertical is the default).
Parameters	N/A
Examples	−horizontal

Table 8.19: Table horizontal/vertical option

Ray		
Syntax	−ray #x #y #z #vx #vy #vz	
Description	Create a table along a predefined ray.	
Parameters	#x #y #z	Ray origin.
	#vx #vy #vz	Ray direction.
Examples	−ray 0 0 0 1 1 1	

Table 8.20: Table ray option

Plane.bc		
Syntax	−plane.bc #x #y #z #nx #ny #nz $bc-name	
Description	Create a table along an intersection between a plane and a boundary surface.	
Parameters	#x #y #z	Plane.bc origin.
	#nx #ny #nz	Plane.bc direction.
	$bc-name	The boundary surface name as assigned by the −−bc −name directive (see Table 8.61).
Examples	−plane.bc 0 0 0 1 1 1 WALL	

Table 8.21: Table plane.bc option

BC		
Syntax	−bc $bc	
Description	Create the table from a boundary condition.	
Parameters	$bc	Boundary conidtion name.
Examples	−bc STRUT	

Table 8.22: Table bc option

8.8 Parallel Options

The parallel input options section starts with the −−parallel directive, followed by a series of parallel options.

Cache		
Syntax	−cache #	
Description	Size of the cache memory (in bytes).	
Parameters	#	Cache size.
Examples	−cache 1000	

Table 8.23: Parallel cache option

Rank		
Syntax	[{{-rank #rank} | -all.ranks} | {-rank.span #rank_min #rank_-max} | {-rank.head #head} | {-rank.tail #tail}} {-threads | -parts | -load | -read.threads} #]...	
Description	Define how many OpenMP threads will be open on each rank.	
Parameters	#rank	Rank number.
	-threads	Number of threads for the current rank.
	-all.ranks	Means that all subsequent options hold for all the ranks.
	-rank.span	Means that all subsequent options hold for the span of ranks.
	-rank.head	Means that all subsequent options hold for the starting ranks.
	-rank.tail	Means that all subsequent options hold for the ending ranks.
	-threads	Set the number of solver threads on a specific (or all) rank.
	-read.threads	Use when memory is insufficient to read all the part in parallel (a significant amount of memory is needed to read a zone part, that is deallocated at the end of the read).
	-parts	Number of partitions.
	-load	Changes the load of a rank, by default all ranks have load 1.
	#	Load level for the current rank.
Examples	−rank 1 -threads 1	
	-rank 2 -threads 8	
	-all.ranks -threads 48	
	-rank 2 -load 4	

Table 8.24: Parallel rank option

8.9 Equation of State Options

The "Equation of State" input options section starts with the −−equation.of.state directive, followed by a series of equation.of.state input options. Each section is used to declare a specific fluid with its equation of state and constitutive laws (*e.g.* Sutherland's law). There could be more than one section, one per fluid type. It is first assigned a name to be used by the user. The current version of **Arion** provides the means to solve either a single component perfect gas fluid or a multi-component gas under the assumption of chemical equilibrium. Future versions will support multi-component gas under non-equilibrium conditions.

Name		
Syntax	−name $fluid	
Description	Name of flow medium.	
Parameters	$fluid	Flow medium.
Examples	−name air	

Table 8.25: equation.of.state name option

Type	
Syntax	−type $eos.type.fluid
Description	Choose the type of equation of state. Available types are listed in Table 8.27
Parameters	$eos.type.fluid Type of equation of state.
Examples	−type eos.ideal.gas

Table 8.26: equation.of.state type option

EOS Types	
EOS type	Description
eos.ideal.gas	Perfect gas (see Section 2.2.3 and Equation 2.10).
eos.table	Perfect gas under equilibrium assumption (required for multi-component.gas eqb system).

Table 8.27: EOS types

R	
Syntax	−R #
Description	Set the specific gas constant (applicable to the eos.ideal.gas type only). Default is 287.22 (air).
Parameters	# Specific gas constant.
Examples	−R 287.22

Table 8.28: equation.of.state R option

Gamma		
Syntax	−gamma #	
Description	Set the heat capacity ratio (γ, applicable to the eos.ideal.gas type only). Default is 1.4 (air).	
Parameters	#	Heat capacity ratio.
Examples	−gamma 1.4	

Table 8.29: equation.of.state gamma option

Ref		
Syntax	−ref.$ #	
Description	Set reference properties. These reference values are relevant in particular for the limiters.	
Parameters	-ref.p	Reference pressure. Default is 1.
	-ref.T	Reference temperature. Default is 1.
	-ref.length	Reference length. Default is 1.
	#	Reference value.
Examples	−ref.p 101325	
	-ref.length 0.5	

Table 8.30: equation.of.state ref option

8.10 Molecular Options

The "Molecular" input options section starts with the −−molecular directive, followed by a series of molecular input options. Each section is used to declare specific type or constants for Sutherland's law. The Sutherland formulae can be set in two ways: The fist way, called molecular.sutherland.air.1, is direct coefficients for μ and κ (see Equation 2.12)

Name		
Syntax	−name $name	
Description	Set the name of the molecular thermodynamic relations.	
Parameters	$name	Name of molecular thermodynamic relations.
Examples	−name therm1	

Table 8.31: Molecular name option

Type		
Syntax	−type $type	
Description	Set the type of the molecular thermodynamic relations. Available types are: molecular.sutherland.air.1 and molecular.sutherland.air.2 (see Section 2.2.4, Equation 2.12).	
Parameters	$type	Type of the molecular thermodynamic relations.
Examples	−type molecular.sutherland.air.1	

Table 8.32: molecular type option

Mu1	
Syntax	−Mu1 #Cmu1
Description	Set the first coefficient for the Sutherland molecular viscosity formula; Default is 1.458e-06 (air, see Equation 2.12 and Table 2.1).
Parameters	#Cmu1 First coefficient for the Sutherland formula
Examples	−Mu1 1.458e-06

Table 8.33: Molecular Mu1 option

Mu2	
Syntax	−Mu2 #Cmu2
Description	Set the second coefficient for the Sutherland molecular viscosity formula; Default is 110.3 (air, see Equation 2.12 and Table 2.1).
Parameters	#Cmu2 Second coefficient for the Sutherland formula
Examples	−Mu2 110.3

Table 8.34: Molecular Mu2 option

Ka1		
Syntax	−Ka1 #Cka1	
Description	Set the first coefficient for the Sutherland thermal conductivity formula; Default is 2.495e-03 (air, see Equation 2.12 and Table 2.1).	
Parameters	#Cka1	First coefficient for the Sutherland formula
Examples	−Ka1 2.495e-03	

Table 8.35: Molecular Ka1 option

Ka2		
Syntax	−Ka2 #Cka2	
Description	Set the second coefficient for the Sutherland heat conductivity formula; Default is 194.0 (air, see Equation 2.12 and Table 2.1).	
Parameters	#Cka2	Second coefficient for the Sutherland formula
Examples	−Ka2 194.0	

Table 8.36: Molecular Ka2 option

Eta0		
Syntax	−Eta0 #Eta0	
Description	Set η_0 for the Sutherland molecular viscosity formula; Default is 1.827×10^{-5} (air, see Equation 2.13 and Table 2.2).	
Parameters	#Eta0	Coefficient for the Sutherland formula
Examples	−Eta0 1.827×10^{-5}	

Table 8.37: Molecular Eta0 option

C0	
Syntax	−C0 #C0
Description	Set C_0 for the Sutherland molecular viscosity formula; Default is 120 (air, see Equation 2.13 and Table 2.2).
Parameters	#C0 Coefficient for the Sutherland formula
Examples	−C0 120

Table 8.38: Molecular C0 option

T0	
Syntax	−T0 #T0
Description	Set T_0 for the Sutherland molecular viscosity formula; Default is 291.15 (air, see Equation 2.13 and Table 2.2).
Parameters	#T0 Coefficient for the Sutherland formula
Examples	−T0 291.15

Table 8.39: Molecular T0 option

Prandtl	
Syntax	−Prandtl #Prandtl
Description	Set the Prandtl number; Default is 0.72 (air, see Equation 2.13 and Table 2.2).
Parameters	#Prandtl Prandtl number
Examples	−Prandtl 0.72

Table 8.40: Molecular Prandtl option

8.11 Tabulate Options

The “Tabulate” input options section starts with the −−tabulate directive, followed by a series of tabulation input options. Tabulation is used in support of the chemical equilibrium model. Creating the eqb tables requires having a −−multi.component.gas section (see Section 8.12).

Name		
Syntax	−name $name	
Description	Set the name of the tabulation.	
Parameters	$name	Table name.
Examples	−name chem	

Table 8.41: Tabulate name option

Type		
Syntax	−type $type	
Description	Set tabulation type.	
Parameters	$type	Tabulation type, options are:
	create	Create new arion tables. Required: −distribution (see Table 8.43), −key (see Table 8.44), and −pT.chemistry as well (see Table 8.45).
	read.arion	Read an existing arion tables. Required: −key as well (see Table 8.44).
	read.outer	Read existing tabulation from non-arion engine. Required: −key as well (see Table 8.44).
Examples	−type create	

Table 8.42: Tabulate type option

Distribution		
Syntax	−distribution $distribution	
Description	Set the table distribution type.	
Parameters	$distribution	Distribution type, options are:
	linear	Linear table distribution.
	log	Logarithmic table distribution.
	tanh	Hyperbolic tangent table distribution (default).
Examples	−distribution tanh	

Table 8.43: Tabulate distribution option

Key		
Syntax	−key $key	
Description	Set the table path for saving or loading.	
Parameters	$key	Table path.
Examples	−key /path/data/tables	

Table 8.44: Tabulate key option

pT.chemistry		
Syntax	−pT.chemistry	
	< -name p -n #n -min #min -max #max >	
	< -name T -n #n -min #min -max #max >	
Description	Create all tables using the given pressure and Temperature data.	
Parameters	-name	Paramemter name (p or T).
	-n #n	Parameter resolution.
	-min #min	Parameter minimum value.
	-max #max	Parameter maximum value.
Examples	−pT.chemistry	
	< -name p -n 400 -min 10000 -max 1e7 >	
	< -name T -n 600 -min 250 -max 700 >	

Table 8.45: Tabulate pT.chemistry option

8.12 Multi Component Gas Options

The “Multi Component Gas” input options section starts with the −−multi.component.gas directive, followed by a series of tabulation input options.

Name		
Syntax	−name $name	
Description	Set the name of the mcg (multi component gas) object.	
Parameters	$name	Multi component gas object name.
Examples	−name air	

Table 8.46: Multi component gas name option

Composition		
Syntax	−composition $composition composition.end	
Description	Set the composition mass fractions.	
Parameters	$compostion	The composition in mass fraction.
	composition.end	End of the composition.
Examples	−composition O2=0.23, NO=0, N=0, O=0, N2=0.77 composition.end	

Table 8.47: Multi component gas composition option

Composition molar		
Syntax	−composition molar $composition composition.end	
Description	Set the composition molar fractions.	
Parameters	$compostion	The composition in mass fraction.
	composition.end	End of the composition.
Examples	−composition molar O2=0.21, NO=0, N=0, O=0, N2=0.79 composition.end	

Table 8.48: Multi component gas composition molar option

cy.eq	
Syntax	−cy.eq
Description	Equilibrate mcg composition based on initial conditions (for applicable systems).
Parameters	N/A
Examples	−cy.eq

Table 8.49: Multi component gas cy.eq option

cy.cutoff	
Syntax	−cy.cutoff
Description	Employ a cutoff to the mass fractions so that they are between 0 and 1.
Parameters	N/A
Examples	−cy.cutoff

Table 8.50: Multi component gas cy.cutoff option

Thrm.db		
Syntax	−thrm.db $thrm.db	
Description	Set the path to the thermal database.	
Parameters	$thrm.db	Path to the thermal database.
Examples	−thrm.db /path/data/thermo.inp	

Table 8.51: Multi component gas thrm.db option

Trns.db		
Syntax	−trns.db $trns.db	
Description	Set the path to the transport database.	
Parameters	$trns.db	Path to the transport database.
Examples	−trns.db /path/data/transport.dat	

Table 8.52: Multi component gas trns.db option

React.db		
Syntax	−react.db $react.db	
Description	Set the path to the reactions database.	
Parameters	$react.db	Path to the reactions database.
Examples	−react.db /path/data/reactions.dat	

Table 8.53: Multi component gas react.db option

Cantera.db		
Syntax	−cantera.db $cantera.db	
Description	Set the path to the Cantera database file.	
Parameters	$cantera.db	Path to the Cantera database.
Examples	−cantera.db /path/data/cantera_file.yaml	

Table 8.54: Multi component gas cantera.db option

Thrm		
Syntax	−thrm $thrm	
Description	Selects the method of calculating the thermodynamic properties.	
Parameters	$thrm	The method of calculating the thermodynamic properties. Possible options are:
	nasa	Use the NASA database to calculate thermodynamic properties (default).
	cantera	Use the cantera file to calculate thermodynamic properties.
Examples	−thrm cantera	
	−thrm nasa	

Table 8.55: Multi component gas thrm option

Trns		
Syntax	−trns $trns	
Description	Select the method of calculating the transport properties.	
Parameters	$trns	Transport properties, options are:
	wilke	Use Wilke's mixture rule to calculate the transport properties, and a constant Schmidt number for mass diffusivity (default).
	cantera	Use the Cantera mixture-averaged model to calculate the transport properties.
	cantera.multi	Use the Cantera multi-component model to calculate the transport properties.
Examples	−trns cantera	
	−trns cantera.multi	

Table 8.56: Multi component gas trns option

Kinetics		
Syntax	−kinetics $kins	
Description	Select the method of calculating the reaction rates,.	
Parameters	$kins	Kinetics properties, options are:
	none	Disable chemical reactions.
	laminar	Use laminar model for chmeical reactions (default).
	cantera	Use Cantera to calculate the chemical reactions based on the reaction set in the Cantera file.
Examples	−kinetics none	
	−kinetics cantera	

Table 8.57: Multi component gas kinetics option

Eqconst		
Syntax	−eqconst $eqconst	
Description	Select the method of calculating the reverse reaction rates.	
Parameters	$eqconst	Method of calculating the reverse reaction rates, options are:
	arrhenius	Directly evaluate reverse reaction rates based on coefficients supplied with the -react.db option. If the reaction file contains the reverse rates, the code will automatically use them and override any other option.
	gibbs	calculate equilibrium constants for each reaction with Gibbs' free energy (default when no reverse rates are available).
Examples	−eqconst gibbs	

Table 8.58: Multi component gas eqconst option

Sc		
Syntax	−Sc #Sc	
Description	Set the Schmidt number.	
Parameters	#Sc	Schmidt number (default is 0.7).
Examples	−Sc 0.8	

Table 8.59: Multi component gas Sc option

Sc.trb		
Syntax	−Sc.trb #Sc	
Description	Set the turbulent Schmidt number.	
Parameters	#Sc	Turbulent Schmidt number (default is 1.0).
Examples	−Sc.trb 0.8	

Table 8.60: Multi component gas Sc.trb option

8.13 BC Options

Boundary conditions are set for each boundary cell, based on the name that has been assigned by the user (see Chapter 6). For each name, representing a group of boundary cells, the boundary conditions are set in two steps. First, assign the name, and second, assign the type. Each group starts with the −−bc directive, followed by the name and type as follows:.

Name		
Syntax	−name $name	
Description	Name of the boundary as set in Pointwise.	
Parameters	$name	Name of boundary.
Examples	−name WALL	

Table 8.61: BC name option

Type		
Syntax	−type $type	
Description	Type of boundary condition. See Chapter 5 and Table 8.63 for available types.	
Parameters	$type	Type of boundary condition.
Examples	−type noslip.wall	

Table 8.62: BC type option

Boundary Condition Types

BC type	Description (see Chapter 5 for details)
impermeable.wall	Impermeable wall.
noslip.wall	No slip wall.
automatic.wall.function	Automatic wall function.
nichols.wall.function	Nichols type wall function.
riemann.inlet	Riemann type inlet boundary conditions.
riemann.outlet	Riemann type outlet boundary conditions.
riemann	Riemann type outer boundary conditions (automatic detection between riemann.inlet and riemann.outlet).
turkel.inlet	Turkel type inlet boundary conditions.
turkel.outlet	Turkel type outlet boundary conditions.
turkel	Turkel type outer boundary conditions (automatic detection between riemann.inlet and riemann.outlet).
inlet	Inlet type outer boundary conditions.
outlet	Pressure outlet type outer boundary conditions.
inout	Inout outer boundary conditions (automatic detection between inlet and outlet).
extrap	Zero order extrapolation.
fixed	Fixed boundary conditions.
stagnation.inlet	Stagnation inlet.
symmetry	Symmetry type conditions.
2d	Two dimensional domain conditions.
axis	Axis for axisymmetric model.

Table 8.63: Boundary condition types

Mass.flow	
Syntax	−mass.flow #mdot
Description	Targeted mass flow rate [kg/s]. Automatically adjusts the pressure boundary value in an attempt to yield the specified mass flow rate (see Section 5.3.6.1). Works only for the “outlet” type boundary condition. Can be used with −relax (see Table 8.65).
Parameters	#mdot — Mass flow rate.
Examples	−mass.flow 10

Table 8.64: BC mass.flow option

Relax	
Syntax	−relax #val
Description	A relaxation parameter for the mass flow rate pressure adjustment (see Table 8.64). Use values smaller than 1 if the pressure changes too rapidly (see Section 5.3.6.1).
Parameters	#val — Relaxation parameter.
Examples	−relax 0.5

Table 8.65: BC relax option

Reference.pressure		
Syntax	−reference.pressure #ref	
Description	Set the mean-flow reference pressure (for force calculation), used to override free-stream reference pressure.	
Parameters	#ref	Reference pressure.
Examples	−reference.pressure 101325	

Table 8.66: BC reference.pressure option

Composition		
Syntax	−composition $composition composition.end	
Description	Override the composition mass fractions (for applicable systems, can specify only species fractions that are not zero).	
Parameters	$compostion	The composition in mass fraction.
	composition.end	End of the composition.
Examples	−composition O2=0.23, N2=0.77 composition.end	

Table 8.67: BC gas composition option

Composition Molar		
Syntax	−composition molar $composition composition.end	
Description	Override the composition molar fractions (for applicable systems, can specify only species fractions that are not zero).	
Parameters	$compostion	The composition in mass fraction.
	composition.end	End of the composition.
Examples	−composition molar O2=0.21, N2=0.79 composition.end	

Table 8.68: BC gas composition molar option

cy.eq	
Syntax	−cy.eq
Description	Equilibrate and override mcg composition based on inlet conditions (for applicable systems).
Parameters	N/A
Examples	−cy.eq

Table 8.69: BC gas cy.eq option

8.13.1 BC Log Options

Log		
Syntax	−log $log [log-options] ...	
Description	Report in log file. See Table 8.71 for log options list.	
Parameters	$log	Log name.
Examples	−log coefficients cl cd	

Table 8.70: Boundary conditions log option

BC Log Options

Log type	Description
wet.surface	Wet surface area.
p.center.sum	Pressure center sum.
p.center.x	Center of pressure x coordinate.
p.center.y	Center of pressure x coordinate.
p.center.z	Center of pressure x coordinate.
p.center.xFy	Center of pressure x coordinate.
p.center.xFz	Center of pressure x coordinate.
p.center.yFx	Pressure center sum(y * dFx) / Fx coordinate.
p.center.yFz	Pressure center sum(y * dFz) / Fz coordinate.
p.center.zFx	Pressure center sum(z * dFx) / Fx coordinate.
p.center.zFy	Pressure center sum(z * dFy) / Fy coordinate .
fx	X coordinate direction force.

fy	Y coordinate direction force.
fz	Z coordinate direction force.
fx.pressure	Force in x direction due to pressure.
fy.pressure	Force in y direction due to pressure.
fz.pressure	Force in z direction due to pressure.
fx.friction	Force in x direction due to friction.
fy.friction	Force in y direction due to friction.
fz.friction	Force in z direction due to friction.
lift	Lift force.
drag	Drag force.
mx	Moment about X coordinate direction.
my	Moment about Y coordinate direction.
mz	Moment about Z coordinate direction.
cfx	X coordinate direction force coefficient.
cfy	Y coordinate direction force coefficient.
cfz	Z coordinate direction force coefficient.
cmx	Moment coefficient about X coordinate direction.
cmy	Moment coefficient about Y coordinate direction.
cmz	Moment coefficient about Z coordinate direction.
cl	Lift coefficient.
cd	Drag coefficient.
mass.flow	Mass flow rate.

Table 8.71: BC log options

8.13.2 Wall Distance Calculations

The following directives pertain to wall type boundary conditions. Normally, one would like to calculate wall distance based on whether a specific boundary has been assigned wall conditions. However, sometimes it is required to override this default. Note that the wall distance is used only for turbulence models that require wall distance. The following two directives are exactly for that. They should be used along with a wall boundary condition.

Wall.distance	
Syntax	−wall.distance
Description	Explicitly set wall distance search for a specific boundary. Default: for all walls.
Parameters	N/A
Examples	−wall.distance

Table 8.72: BC wall.distance option

No.wall.distance	
Syntax	−no.wall.distance
Description	Explicitly disable wall distance search for a specific boundary. Default: for all boundaries other than walls.
Parameters	N/A
Examples	−no.wall.distance

Table 8.73: BC no.wall.distance option

8.13.3 Free-Stream Conditions Override

Mach		
Syntax	−Mach #M	
Description	Override the free-stream Mach number.	
Parameters	#M	Free-stream Mach number.
Examples	−Mach 0.95	

Table 8.74: BC Mach option

Velocity.magnitude		
Syntax	−velocity.magnitude #V	
Description	Override the free-stream velocity [m/s].	
Parameters	#V	Free-stream velocity [m/s].
Examples	−velocity.magnitude 200	

Table 8.75: BC velocity.magnitude option

Alpha		
Syntax	−alpha #aoa	
Description	Override the free-stream angle of attack [degrees].	
Parameters	#aoa	Free-stream angle of attack [degrees].
Examples	−alpha 5	

Table 8.76: BC alpha option

Beta		
Syntax	−beta #beta	
Description	Override the free-stream side slip angle [degrees].	
Parameters	#beta	Free-stream side slip angle [degrees].
Examples	−beta 3	

Table 8.77: BC beta option

U		
Syntax	−u #u	
Description	Override the free-stream x coordinate direction velocity [m/s].	
Parameters	#u	Free-stream x coordinate direction velocity [m/s].
Examples	−u 150	

Table 8.78: BC u option

V		
Syntax	−v #v	
Description	Override the free-stream y coordinate direction velocity [m/s].	
Parameters	#v	Free-stream y coordinate direction velocity [m/s].
Examples	−v 20	

Table 8.79: BC v option

W		
Syntax	−w #w	
Description	Override the free-stream z coordinate direction velocity [m/s].	
Parameters	#w	Free-stream z coordinate direction velocity [m/s].
Examples	−w 10	

Table 8.80: BC w option

T		
Syntax	−T #T	
Description	Override the free-stream temperature [K]. In the case of the stagnation inlet (see Section 5.4) this directive pertains to the free-stream stagnation temperature.	
Parameters	#T	Free-stream temperature [K].
Examples	−T 300	

Table 8.81: BC T option

p		
Syntax	−p #p	
Description	Override the free-stream pressure [Pa]. In the case of the stagnation inlet (see Section 5.4) this directive pertains to the free-stream stagnation pressure.	
Parameters	#p	Free-stream pressure [Pa].
Examples	−p 105000	

Table 8.82: BC p option

Trb.intensity		
Syntax	−trb.intensity #ti	
Description	Override the free-stream turbulent intensity (in absolute fraction, not percentage).	
Parameters	#ti	Free-stream turbulent intensity (in absolute fraction, not percentage). Default is 0.001.
Examples	−trb.intensity 0.01	

Table 8.83: BC trb.intensity option

Trb.dnu		
Syntax	−trb.dnu #dnu	
Description	Override the free-stream turbulent Spalart-Allmaras model variable ($\tilde{\nu}$).	
Parameters	#dnu	Free-stream turbulent Spalart-Allmaras model variable.
Examples	−trb.dnu 0.1	

Table 8.84: BC trb.dnu option

8.14 System Options

A system refers to a "System of Equations," *e.g.,* Euler or Reynolds Averaged Navier-Stokes equations. Other examples may include, turbulence model equations or various physical model equations (not included in the current revision of the code). Each system starts with the directive −−system, followed by a series of options.

Certain systems require that an additional, complimentary system would be defined. For example, when simulating turbulent flows it is required to define a mean flow system that is consistent with the turbulence model selected and a second system to solve the actual turbulence model equations.

Type		
Syntax	−type $type	
Description	Type of equation set to be solved.	
Parameters	$type	Type of equation set. See Table 8.86 for equation system types.
Examples	−type ideal.gas.mf.inviscid −type ideal.gas.mf.tnt −type turbulent.kw.sst	

Table 8.85: System type option

System Types

System type	Description
ideal.gas.mf.inviscid	Solve the Euler equations assuming inviscid, perfect gas flow.
ideal.gas.mf.viscous	Solve the Navier-Stokes equations assuming laminar perfect gas flow.
ideal.gas.mf.tnt	Solve the Navier-Stokes equations assuming turbulent ideal gas flow (using the $k-\omega$-TNT turbulence model). This assumes an additional system for solving the $k-\omega$-TNT turbulence model equations.
ideal.gas.mf.sst	Solve the Navier-Stokes equations assuming turbulent ideal gas flow (using the $k-\omega$-SST turbulence model). This assumes an additional system for solving the $k-\omega$-SST turbulence model equations.
ideal.gas.mf.sa	Solve the Navier-Stokes equations assuming turbulent ideal gas flow (using the Spalart-Allmaras turbulence model) This assumes an additional system for solving the Spalart-Allmaras turbulence model equation.
turbulent.kw.tnt	Solve the $k-\omega$-TNT turbulence model equations.
turbulent.kw.sst	Solve the $k-\omega$-SST turbulence model equations.
turbulent.kw.sst.sas	Solve the $k-\omega$-SST-SAS turbulence model equations.
turbulent.dnu.sa	Solve the Spalart-Allmaras turbulence model equation.
eqb.inviscid	Solve the Euler equations assuming inviscid perfect gas flow under chemical equilibrium.
eqb.viscous	Solve the Navier-Stokes equations assuming laminar perfect gas flow under chemical equilibrium.
eqb.tnt	Solve the Navier-Stokes equations assuming turbulent perfect gas flow under chemical equilibrium (requires subsequent mean-flow $k-\omega$-TNT model system).
eqb.sst	Solve the Navier-Stokes equations assuming turbulent perfect gas flow under chemical equilibrium (requires subsequent mean-flow $k-\omega$-SST model system).
eqb.sa	Solve the Navier-Stokes equations assuming turbulent perfect gas flow under chemical equilibrium (requires subsequent mean-flow Spalart-Allmaras model system).

Table 8.86: System types

Cell.gradient		
Syntax	−cell.gradient $method	
Description	Method to calculate the cell center gradient that is required for high order approximations.	
Parameters	$method	Method for cell gradient calculation. See Table 8.88 for a list of available cell gradient methods.
Examples	−cell.gradient green.gauss.node	

Table 8.87: System cell.gradient option

Cell Gradient Types

Cell Gradient type	Description
green.gauss.node	Calculate cell gradient based on the Green-Gauss Theorem using node reconstruction (default).
green.gauss.cell	Calculate cell gradient based on the Green-Gauss Theorem using cell center reconstruction.
least.square.1	Calculate cell gradient based on least square approximations using r (the distance) as the weight.
least.square.1.5	Calculate cell gradient based on least square approximations using $r^{1.5}$ as the weight.
least.square.2	Calculate cell gradient based on least square approximations using r^2 as the weight.

Table 8.88: Cell gradient types

Face.gradient		
Syntax	−face.gradient $method	
Description	Method to calculate the face gradient that is required for viscous fluxes calculations.	
Parameters	$method	Method for face gradient calculation. See Table 8.90 for a list of available face gradient methods.
Examples	−face.gradient diamond	

Table 8.89: System face.gradient option

Face Gradient Types

Face Gradient type	Description
diamond	Calculate face gradient based on the Green-Gauss Theorem using a diamond-like shape (default).
thin.layer	Calculate face gradient based on the thin layer assumption.
hasselbacher	Calculate face gradient based on defect correction by Hasselbacher.

Table 8.90: Face gradient types

Limiter		
Syntax	−limiter $limiter	
Description	Type of limiter.	
Parameters	$limiter	Type of limiter. See Table 8.92 for limiter types.
Examples	−limiter venka.2d	

Table 8.91: Limiter option

Limiter Types

Limiter type	Description
none	No limiter.
1st	First order.
venka.2d	Venkatakrishnan limiter (2D domain).
venka.3d	Venkatakrishnan limiter (3D domain).
mlp.2d	MLP-u2 limiter (2D domain).
mlp.3d	MLP-u2 limiter (3D domain).

Table 8.92: Limiter types

Venka.k	
Syntax	−venka.k #k
Description	Change the Venkatakrishnan limiter coefficient. Default is 5.
Parameters	#k Venkatakrishnan limiter coefficient.
Examples	−venka.k 5

Table 8.93: System venka.k option

Time.step		
Syntax	−time.step #initial #iterations {cfl.linear | cfl.exponential | cfl.tangential | dt.linear | dt.exponential | dt.tangential } #final	
Description	Set the CFL number or time step (first order single time stepping).	
Parameters	#initial	Initial CFL number or time step.
	#iterations	Number of iteration for which the CFL number or time step grows from #initial to #final.
	cfl.linear	Linear growth of the CFL number.
	cfl.exponentia	Exponential growth of the CFL number.
	cfl.tangential	Hyperbolic tangent growth of the CFL number.
	dt.linear	Linear growth of the time step.
	dt.exponentia	Exponential growth of the time step.
	dt.tangential	Hyperbolic tangent growth of the time step.
	#final	Final CFL number or time step.
Examples	−time.step 5 50 cfl.linear 10	

Table 8.94: System CFL option

Implicit.jacobi

Syntax	−implicit.jacobi
Description	Use Jacobi instead of the default Gauss-Seidel.
Parameters	N/A
Examples	−implicit.jacobi

Table 8.95: System implicit.jacobi option

Convection.noslip.diagonal

Syntax	−convection.noslip.diagonal {normal | analytic | implicit.on.noslip}	
Description	Select the way the diagonal convective jacobian is estimated on no-slip boundaries.	
Parameters	normal	The inviscid diagonal is evaluated directly based on the scheme.
	analytic	Use analytical Jacobian
	implicit.on.no	Implicit treatment of wall boundary conditions through the Jacobian.
Examples	−convection.noslip.diagonal analytic	

Table 8.96: System convection.noslip.diagonal option

Convection.impermeable.diagonal

Syntax	−convection.impermeable.diagonal {normal | analytic | implicit.on.noslip}	
Description	Selects the way the diagonal convective jacobian is estimated on impermeable boundaries.	
Parameters	normal	The inviscid diagonal is evaluated directly based on the scheme.
	analytic	Use analytical Jacobian
	implicit.on.no	Implicit treatment of wall boundary conditions through the Jacobian.
Examples	−convection.impermeable.diagonal analytic	

Table 8.97: System convection.impermeable.diagonal option

Convection.symmetry.diagonal		
Syntax	−convection.symmetry.diagonal {normal | analytic | implicit.on.noslip}	
Description	Selects the way the diagonal convective jacobian is estimated on symmetry boundaries.	
Parameters	normal	The inviscid diagonal is evaluated directly based on the scheme.
	analytic	Use analytical Jacobian
	implicit.on.no	Implicit treatment of wall boundary conditions through the Jacobian.
Examples	−convection.symmetry.diagonal analytic	

Table 8.98: System convection.symmetry.diagonal option

Realizability.trb.w	
Syntax	−realizability.trb.w
Description	Adds realizability condition for omega (applies only to $k-\omega$-SST turbulence model).
Parameters	N/A
Examples	−realizability.trb.w

Table 8.99: System realizability.trb.w option

Source.mpk	
Syntax	−source.mpk #mpk
Description	Limit turbulence model production (applies only to $k-\omega$ turbulence models).
Parameters	#mpk Upper limit for production term.
Examples	−source.mpk 5

Table 8.100: System source.mpk option

Damp	
Syntax	−damp #damp-factor
Description	Set damping to the convection Jacobian. Values that are less than unity increase stability. Not recommended for turbulence model equations. The recommended damping factor is 0.5-1.0. Default is 1.
Parameters	#damp-factor Damping factor.
Examples	−damp 0.75

Table 8.101: System damp option

Relax	
Syntax	−relax #relax-factor
Description	Set relaxation to the solution increment. The recommended relaxation factor is 0.5-1.0. Default is 1.
Parameters	#relax-factor Relaxation factor.
Examples	−relax 0.75

Table 8.102: System relaxation option

iteration.convergence	
Syntax	−iteration.convergence #criterion
Description	Set the convergence criterion (log(Res/Res0)), default is ignored. For unsteady flows this would be the convergence of the whole simulation. For dual time this would be convergence of the iterations.
Parameters	#criterion Convergence criterion in terms of [log(Res/Res0)].
Examples	−iteration.convergence -5

Table 8.103: System iteration.convergence option

2d.tolerance	
Syntax	−2d.tolerance #tolerance
Description	Tolerance for two-dimensional detection.
Parameters	#tolerance Set the tolerance.
Examples	−2d.tolerance 1e-8

Table 8.104: System 2d.tolerance option

Log		
Syntax	−log $log [log-options] ...	
Description	Report in log file. See Table 8.106 for log options list.	
Parameters	$log	Log name.
Examples	−log coefficients cl cd	

Table 8.105: System log option

System Log Options

Log type	Description
cfl	System CFL.
dt	System Δt.
time	Time.
residual	System residual.
log.residual	System log of the residual.
minimal.timestep	Minimal time step (for non sp.petsc).
orig(minimal.timestep)	Minimal time step original cell index (for non sp.petsc).
cell(minimal.timestep)	Minimal time step zonal cell index (for non sp.petsc).
zone(minimal.timestep)	Minimal time step zone index (for non sp.petsc).
orig(res)	Max residual original cell number.
cell(res)	Max residual zonal cell number.
zone(res)	Max residual zone number.
max(mt)	Max of the turbulence viscosity (where applicable).
cell(mt)	Max turbulent viscosity cell number (where applicable).
zone(mt)	Max turbulent viscosity zone number (where applicable).
petsc.iters	Petsc iteration number.
petsc.residual	Petsc residual.

Table 8.106: System log options

Plot		
Syntax	−plot $plot [plot-options] ...	
Description	Functions to include in FV-UNS or CGNS file. See Table 8.108 for plot functions list.	
Parameters	$plot	Plot name.
Examples	−plot NACA0012 r p velocity residual	

Table 8.107: System plot option

Plot Functions

Plot functions	Description
cy	Species mass fraction (where applicable).
residual	Residual field.
relax	Relaxation field.
velocity	Velocity vector field (where applicable).
{u | v | w }	Velocity vector field components (scalars, where applicable).
dudx, dudy, dudz	Cell centered u-velocity derivatives.
dvdx, dvdy, dvdz	Cell centered v-velocity derivatives.
dwdx, dwdy, dwdz	Cell centered w-velocity derivatives.
r	Density field (where applicable).
p	Pressure field (where applicable).
Cp	Pressure coefficient (where applicable).
Cf	Skin-friction coefficient (where applicable).
T	Temperature field (where applicable).
{lim.r | lim.p | lim.u | lim.v | lim.w}	Limiter fields (where applicable).
wall.distance	Wall distance field (where applicable).
mt	Turbulent viscosity field (where applicable).
qflux	Normal heat flux (where applicable).
mach	Mach number.
trb.k	Turbulent kinetic energy field (where applicable).
trb.w	Turbulent specific dissipation rate field (where applicable).
{trb.lim.k | trb.lim.w}	Turbulence model limiters (where applicable).
yplus	y^+ (where applicable).
Tw	Magnitude of $\bar{\tau}_{wall}$ (where applicable).
Tw.vec	The vector $\bar{\tau}_{wall}$ (where applicable).
Tw{x | y | z }	X | Y | Z direction component of $\bar{\tau}_{wall}$.

Table 8.108: Plot functions

8.15 Solve Options

The Solve input options section starts with the −−solve directive, followed by a series of solve options.

Convection.flux		
Syntax	−convection.flux $method	
Description	Set the convection flux approximation method.	
Parameters	$method	Flux approximation method. See Table 8.110 for available convection flux types.
Examples	−convection.flux hllc.roe	

Table 8.109: Solve convection.flux option

Convection Flux Types

Convection flux type	Description
hllc.roe	HLLC Roe.
hllc.davis	HLLC Davis.
ausm	AUSM.
ausm.up	AUSM$^+$-up
ausm.dv	AUSM-DV

Table 8.110: Convection flux types

Convection.jacobian	
Syntax	−convection.jacobian $jacobian
Description	Set the convection Jacobian. See Table 8.112 for available convection Jacobian types.
Parameters	$jacobian Jacobian.
Examples	−convection.jacobian hllc.roe

Table 8.111: Solve convection.jacobian option

Convection Jacobian Types

Convection Jacobian type	Description
hllc.roe	HLLC Roe.
hllc.davis	HLLC Davis.
van.leer	van Leer.
rossow.low.mach	Low Mach number Rossow.
matrix.free	Matrix free.

Table 8.112: Convection Jacobian types

Spatial.order		
Syntax	−spatial.order #	
Description	Set the convection flux spatial order. Available options are either 1 for first order or 2 for second order.	
Parameters	#	Convection flux approximation order.
Examples	−spatial.order 2	

Table 8.113: Solve spatial.order option

Sweeps		
Syntax	−sweeps #	
Description	Set the number of sweeps within an iteration.	
Parameters	#	Number of sweeps. Can be any even number. Default is 6.
Examples	−sweeps 4	

Table 8.114: Solve sweeps option

Time.march	
Syntax	−time.march $method
Description	Set the time marching method. See Table 8.116 for a list of available time marching methods.
Parameters	$method Time marching method.
Examples	−time.march implicit.pgs

Table 8.115: Solve time.march option

Time Marching Methods

Time marching method	Description
explicit.euler	Explicit Euler time marching scheme.
explicit.rk3	Third order Runge-Kutta explicit time marching scheme.
explicit.rk4	Fourth order Runge-Kutta explicit time marching scheme.
explicit.rk5	Fifth order Runge-Kutta explicit time marching scheme.
implicit.pgs	Implicit point Gauss-Seidel time marching scheme.
implicit.b2.pgs	Second order implicit B2 time marching scheme.
implicit.heun.R.pgs	Second order Runge-Kutta implicit point Gauss-Seidel time marching scheme.
implicit.rk3.R.pgs	Third order Runge-Kutta implicit point Gauss-Seidel time marching scheme.
implicit.rk4.R.pgs	Fourth order Runge-Kutta implicit point Gauss-Seidel time marching scheme.
implicit.rk5.R.pgs	Fifth order Runge-Kutta implicit point Gauss-Seidel time marching scheme.
implicit.rk3.R.petsc	Krylov-based Implicit (Runge-Kutta R, 3rd order).
implicit.rk4.R.petsc	Krylov-based Implicit (Runge-Kutta R, 4th order).
implicit.b2.petsc	Krylov-based Implicit (B2).

Table 8.116: Time marching methods

Time.accurate.explicit		
Syntax	−time.accurate.explicit #steps #time-step	
Description	Use 3rd order Runge-Kutta TVD (see Section 4.4.1.3) or Explicit Euler as a time-accurate method. Requires to set the time marching method (explicit.euler or explicit.rk3) as in Table 8.116.	
Parameters	#steps	Number of steps.
	#time-step	Time step.
Examples	−time.march explicit.rk3	
	−time.accurate.explicit 1000 0.0001	

Table 8.117: Solve time.accurate.explicit option

Iterations		
Syntax	−iterations #	
Description	Set the number of iterations. For unsteady flows this would be the number of iterations of the whole simulation. For dual time this would be the maximum iterations of the current time step.	
Parameters	#	Number of iterations.
Examples	−iterations 1000	

Table 8.118: Solve iterations option

Time.step.reduction		
Syntax	−time.step.reduction #attempts	
Description	Set time the number of consecutive CFL reduction attempts when non-physical values are detected in the solution.	
Parameters	#attempts	Number of attempts.
Examples	−time.step.reduction 3	

Table 8.119: Solve time.step.reduction option

Dual.time		
Syntax	−dual.time #steps #time-step	
Description	Set dual time stepping simulation.	
Parameters	#steps	Number of physical time steps.
	#time-step	Physical time step (Δt).
Examples	−dual.time 200 0.0001	

Table 8.120: Solve dual.time option

Global.minimal.timestep	
Syntax	−global.minimal.timestep
Description	Forces all the cells to use the global minimal time step that was calculated.
Parameters	N/A
Examples	−global.minimal.timestep

Table 8.121: Solve global.minimal.timestep option

Save		
Syntax	−save #	
Description	Set the intermittency of output of restart files.	
Parameters	#	Number of steps between output.
Examples	−save 10	

Table 8.122: Solve save option

Save.path		
Syntax	−save.path $path	
Description	Set the path for output of log and restart files.	
Parameters	$path	Path of log and restart files.
Examples	−save.path ./save/	

Table 8.123: Solve save.path option

Save.sequential	
Syntax	−save.sequential
Description	Adds iteration signature to the saves.
Parameters	N/A
Examples	−save.sequential

Table 8.124: Solve save.sequential option

Load.path		
Syntax	−load.path $path	
Description	Set the path for restart files.	
Parameters	$path	Path of restart files.
Examples	−load.path ./load/	

Table 8.125: Solve load.path option

Load.sequence		
Syntax	−load.sequence #iteration	
Description	Load a specific iteration.	
Parameters	#iteration	Iteration for load.
Examples	−load.sequence 1000	

Table 8.126: Solve load.sequence option

EOS		
Syntax	−eos $eos	
Description	Set the equation of state.	
Parameters	$eos	Equation of state. Currently, only perfect gases are supported.
Examples	−eos air	

Table 8.127: Solve eos option

Molecular		
Syntax	−molecular $molecular	
Description	Set the name of the molecular viscosity and thermal conductivity to use.	
Parameters	$molecular	Molecular relations name.
Examples	−molecular air-suther	

Table 8.128: Solve molecular option

P		
Syntax	−p #	
Description	Set the free stream pressure.	
Parameters	#	Free stream pressure.
Examples	−p 101325	

Table 8.129: Solve p option

T		
Syntax	−T #	
Description	Set the free stream temperature.	
Parameters	#	Free stream temperature.
Examples	−p 288	

Table 8.130: Solve T option

Mach		
Syntax	−Mach #	
Description	Set the free stream Mach number.	
Parameters	#	Free stream Mach number.
Examples	−Mach 0.8	

Table 8.131: Solve Mach option

Velocity.magnitude		
Syntax	−velocity.magnitude #	
Description	Set the free stream velocity magnitude.	
Parameters	#	Free stream velocity magnitude.
Examples	−velocity.magnitude 200	

Table 8.132: Solve velocity magnitude option

U		
Syntax	−u #	
Description	Set the free stream X coordinate direction velocity component.	
Parameters	#	Free stream X coordinate direction velocity component.
Examples	−u 200	

Table 8.133: Solve u option

V		
Syntax	−v #	
Description	Set the free stream Y coordinate direction velocity component.	
Parameters	#	Free stream Y coordinate direction velocity component.
Examples	−v 0	

Table 8.134: Solve v option

W		
Syntax	−w #	
Description	Set the free stream Z coordinate direction velocity component.	
Parameters	#	Free stream Z coordinate direction velocity component.
Examples	−w 20	

Table 8.135: Solve w option

Alpha		
Syntax	−alpha #	
Description	Set the free stream angle of attack.	
Parameters	#	Free stream angle of attack in degrees.
Examples	−alpha 10	

Table 8.136: Solve alpha option

Beta		
Syntax	−beta #	
Description	Set the free stream side slip angle.	
Parameters	#	Free stream side slip angle in degrees.
Examples	−beta 0	

Table 8.137: Solve beta option

Trb.intensity		
Syntax	−trb.intensity #	
Description	Set the free stream turbulence intensity.	
Parameters	#	Free stream turbulence intensity (in absolute fraction, not percentage).
Examples	−trb.intensity 0.01	

Table 8.138: Solve trb.intensity option

Trb.mt		
Syntax	−trb.mt #mt	
Description	Set the free stream turbulence viscosity.	
Parameters	#mt	Free stream turbulence viscosity (in absolute fraction from the molecular viscosity, not percentage).
Examples	−trb.mt 0.01	

Table 8.139: Solve trb.mt option

Reference.velocity	
Syntax	−reference.velocity #ref
Description	Set the reference velocity. Default is the mean flow free stream velocity.
Parameters	#ref Reference velocity.
Examples	−reference.velocity 300

Table 8.140: Solve reference.velocity option

Reference.mach	
Syntax	−reference.mach #ref
Description	Set the reference Mach number. Default is the mean flow free stream Mach number.
Parameters	#ref Reference Mach number.
Examples	−reference.mach 0.75

Table 8.141: Solve reference.mach option

Reference.pressure	
Syntax	−reference.pressure #ref
Description	Set the mean-Flow reference pressure, used to override free-stream pressure.
Parameters	#ref Reference pressure.
Examples	−reference.pressure 101325

Table 8.142: Solve reference.pressure option

Reference.velocity.trb		
Syntax	−reference.velocity.trb #ref	
Description	Set the reference velocity for the turbulence model (used to override the free-stream velocity).	
Parameters	#ref	Free-stream velocity [m/s].
Examples	−reference.velocity.trb 210	

Table 8.143: Solve reference.velocity.trb option

Reference.mach.trb		
Syntax	−reference.mach.trb #ref	
Description	Set the reference Mach number for the turbulence model (used to override the free-stream Mach number).	
Parameters	#ref	Free-stream Mach number.
Examples	−reference.mach.trb 0.7	

Table 8.144: Solve reference.mach.trb option

Reference.longitudinal		
Syntax	−reference.longitudinal #ref	
Description	Set the reference length for Reynolds number evaluation. Unused by the current version.	
Parameters	#ref	Reference length.
Examples	−reference.length 1.5	

Table 8.145: Solve reference.length option

Ref		
Syntax	−ref.$ # { # # }	
Description	Set the reference length, area, and reference point for force and moment coefficient calculations.	
Parameters	-reference.area	Reference area (scalar).
	-reference.len	Reference length (vector).
	-reference.point	Reference point (vector).
	#	Reference value.
Examples	−reference.len 0.325 0.1 0.325	
	−reference.point 1 0 0	

Table 8.146: Solve ref option

Plot	
Syntax	−plot [plot-options] ...
Description	Add to the fvuns file. See Table 8.148 for plot options list.
Parameters	N/A
Examples	−plot rank owner orig

Table 8.147: Solve plot option

Solve Plot Options	
Log type	Description
rank	Add the rank to the fvuns plot file.
part	Add the part to the fvuns plot file.
owner	Add ownership to the fvuns plot file.
orig	Add original node index to the fvuns plot file.

Table 8.148: Solve plot options

Log		
Syntax	−log $log [log-options] ...	
Description	Report in log file. See Table 8.150 for log options list.	
Parameters	$log	Log name.
Examples	−log progress-logs iter step time	

Table 8.149: Solve log option

Solve Log Options

Log type	Description
iter	Iteration number.
step	Step number.
pos	Position # (iteration for steady state, step for dual time simulation).
time	Physical time for dual time simulation.
exec.time	Cumulative execution time.
iter.avg.time	iteration average time.
iteration.time	Iteration time.
step.time	Step time.
petsc.mem.usage	Petsc memory usage in mebibyte.
petsc.mem.malloc	Petsc memory malloc in mebibyte.

Table 8.150: Solve log options

Wall.distance		
Syntax	−wall.distance {quick [#fx #fy #fz]} | {{exact | exact.hiorder}[#fx #fy #fz #cx #cy #cz]}	
Description	Set the wall distance algorithm, fx, fy, fz determine the boundary division allocation (surface) while cx, cy, cz determine the cell division allocation (volume).	
Parameters	quick	Use the quick algorithm.
	exact	Exact wall distance calculation.
	exact.hiorder	High order, exact wall distance calculation.
	#fx #fy #fz	Number of division of surface faces in x, y, and, z directions.
	#cx #cy #cz	Number of division of volume cells in x, y, and, z directions.
Examples	−wall.distance exact	

Table 8.151: Solve wall distance option

Source.mom		
Syntax	−source.mom #fx #fy #fz	
Description	Set a constant momentum source term of the form r × [fx, fy, fz] Where r is the local density.	
Parameters	#fx #fy #fz	Momentum density???.
Examples	−source.mom 1.0 1.0 1.0	

Table 8.152: Solve source.mom option

Axisymmetric	
Syntax	−axisymmetric
Description	Set an axisymmetric simulation. The 2-D grid must be positioned in the $x - z$ plane and extruded by one node into the y-direction. Here, x represents the axis of rotation and z corresponds to the radius of rotation.
Parameters	N/A
Examples	−axisymmetric

Table 8.153: Solve axisymmetric option

8.15.1 Line Search

Line.search iters	
Syntax	−line.search iters #iterations
Description	Set the number of line.search iterations default is 0 (no line search iterations).
Parameters	#iterations Number of line search iterations.
Examples	−line.search iters 1

Table 8.154: Solve line search iterations option

Line.search min.dq	
Syntax	−line.search min.dq #dq
Description	Set the minimal relaxation factor that the line search can use (optional, default is 0.1).
Parameters	#dq Minimal relaxation factor.
Examples	−line.search min.dq 1e-2

Table 8.155: Solve line search minimum dq option

Line.search poly.order		
Syntax	−line.search poly.order #order	
Description	Set the polynomial order of the back-traced line search curve-fitting, can be 2 or 3 (optional, default is 3).	
Parameters	#order	Order of back-traced line search curve-fitting polynomial.
Examples	−line.search poly.order 3	

Table 8.156: Solve line search polynomial order option

Line.search beta		
Syntax	−line.search beta #beta	
Description	Set the convergence check criterion constant in the line search algorithm (optional, default is 1.000000e-04).	
Parameters	#beta	Line search convergence criterion constant.
Examples	−line.search beta 1e-3	

Table 8.157: Solve line search beta option

8.16 PETSc Toolkit

The PETSc input option section starts with the −−petsc directive, followed by a series of petsc options. For advanced usage of the PETSc package the user is referred to the command line help or the online PETSc manual.

KSP_view	
Syntax	−ksp_view
Description	View internal Krylov solver iterations.
Parameters	N/A
Examples	−ksp_view

Table 8.158: Petsc ksp_view option

KSP_monitor_true_residual	
Syntax	−ksp_monitor_true_residual
Description	Monitor true (un-preconditioned) residuals.
Parameters	N/A
Examples	−ksp_monitor_true_residual

Table 8.159: Petsc ksp_monitor_true_residual option

KSP_norm_type		
Syntax	−ksp_norm_type [type]	
Description	Select type of norm used to determine convergence of Krylov solver.	
Types	none	Use no norm calculations.
	preconditioned	Use the preconditioned residual norm.
	unpreconditioned	Use the unpreconditioned residual norm..
	natural	Use the norm induced by the linear operator.
Examples	−ksp_norm_type none	

Table 8.160: Petsc ksp_norm_type option

KSP_max_it		
Syntax	−ksp_max_it #	
Description	Set maximum number of Krylov solver iterations.	
Parameters	#	Maximum number of Krylov iterations.
Examples	−ksp_max_it 20	

Table 8.161: Petsc ksp_max_it option

KSP_rtol	
Syntax	−ksp_rtol #
Description	Set relative tolerance threshold for convergence (reduction in orders of magnitude).
Parameters	# Relative tolerance threshold for convergence.
Examples	−ksp_rtol 2e-2

Table 8.162: Petsc ksp_rtol option

KSP_atol	
Syntax	−ksp_atol #
Description	Set absolute tolerance threshold for convergence (absolute size of residual, use when residuals approach machine accuracy).
Parameters	# Aelative tolerance threshold for convergence.
Examples	−ksp_atol 1e-10

Table 8.163: Petsc ksp_atol option

KSP_type	
Syntax	−ksp_type [type]
Description	Select Krylov solver (Linear Solvers and Krylov Methods (KSP)).
Types	Selected type list can be found in Table 8.165. For a complete list the user is referred to Petsc relevant user guide page.
Examples	−ksp_type fgmres

Table 8.164: Petsc ksp_type option

KSP Types	
Type	Description
gmres	Generalized Minimal Residual method.
fgmres	Flexible Generalized Minimal Residual method.

Table 8.165: Petsc KSP Types

KSP__gmres__restart		
Syntax	−ksp__gmres__restart #	
Description	Set GMRES restart length (increase to improve convergence on account of memory and cpu time, default: 30).	
Parameters	#	GMRES restart length.
Examples	−ksp__gmres__restart 30	

Table 8.166: Petsc ksp__gmres__restart option

PC__type	
Syntax	−pc__type [type]
Description	Set pre-conditioner (for additional information the reader is referred to Petsc relevant user guide page).
Types	Selected type list can be found in Table 8.169. For a complete list the user is referred to Petsc list of pre-conditioners.
Examples	−pc__type asm

Table 8.167: Petsc pc__type option

Sub_pc_type

Syntax	−sub_pc_type [type]
Description	Select method used to invert the sub-domains created by the additive Schwartz pre-conditioner.
Types	Selected type list can be found in Table 8.169. For a complete list the user is referred to Petsc list of pre-conditioners.
Examples	−sub_pc_type ilu −sub_pc_type preonly

Table 8.168: Petsc sub_pc_type option

PC Types

Type	Description
asm	Additive Schwarz.
ilu	Incomplete LU
preonly	Disable Krylov solver in computing inverse of asm sub-domains.

Table 8.169: Petsc PC Types

Sub__pc__factor__levels	
Syntax	−sub_pc_factor_levels #
Description	Set sub-domains pre-conditioner factor level.
Parameters	# Sub-domains pre-conditioner factor level.
Examples	−sub_pc_factor_levels 2

Table 8.170: Petsc sub_pc_factor_levels option

Sub__pc__factor__fill	
Syntax	−sub_pc_factor_fill #
Description	Indicate the amount of fill you expect in the factored matrix (for the sbu-domain).
Parameters	# Amount of fill you expect in the factored matrix.
Examples	−sub_pc_factor_fill 1

Table 8.171: Petsc sub_pc_factor_fill option

8.17 Domain Options

The Domain input options section starts with the −−domain directive, followed by a series of domain options. It is used to define a volumetric domain.

Name		
Syntax	−name $name	
Description	Set the name of the domain.	
Parameters	$name	Domain name.
Examples	−name my_domain_name	

Table 8.172: Domain name option

Init		
Syntax	−init $init_name	
Description	Set the name of the init (see Section 8.18, Table 8.175).	
Parameters	$init_name	Init name (see Table 8.175).
Examples	−init my_init_name	

Table 8.173: Domain init option

-type box

Type		
Syntax	−type $type {[x.min #val] [x.max #val] [y.min #val] [y.max #val] [z.min #val] [z.max #val]}	
Description	Set the type of the domain. Currently available types: 'box'. The domain 'box' is a 3-D rectangular cuboid defined by min / max value on each axis. If one axis limits is not defined, then it assumed to be $+\infty$ or $-\infty$	
Parameters	$type	domain type.
	box	3D rectangular cuboid.
	x.min	set x min.
	x.max	set x max.
	y.min	set y min.
	y.max	set y max.
	z.min	set z min.
	z.max	set z max.
	#val	Value.
Examples	−type box x.min 0 x.max 5 y.min -2 y.max 2 z.min -1 z.max 1	
	−type box x.min 0 y.min -3 z.min -5	

Table 8.174: Domain type option

8.18 Init Options

The Init input options section starts with the −−init directive, followed by a series of domain options. It is used to define a volumetric domain.

Name		
Syntax	−name $name	
Description	Set the name of the init.	
Parameters	$name	Init name.
Examples	−init my_init_name	

Table 8.175: Init name option

T		
Syntax	−T #T	
Description	Override the existing temperature [K].	
Parameters	#T	Temperature.
Examples	−T 300	

Table 8.176: Init T option

P		
Syntax	−p #p	
Description	Override the existing pressure [pa].	
Parameters	#p	pressure.
Examples	−p 67000	

Table 8.177: Init p option

U		
Syntax	−u #u	
Description	Override the existing x direction velocity [m/s].	
Parameters	#u	velocity.
Examples	−u 150	

Table 8.178: Init u option

V		
Syntax	−v #v	
Description	Override the existing y direction velocity [m/s].	
Parameters	#v	velocity.
Examples	−v 0	

Table 8.179: Init v option

W		
Syntax	−w #w	
Description	Override the existing z direction velocity [m/s].	
Parameters	#w	velocity.
Examples	−w 10	

Table 8.180: Init w option

Composition		
Syntax	−-composition $composition composition.end	
Description	Override the existing mcg composition (for applicable systems) mass fraction (can specify only species fractions that are not zero).	
Parameters	$compostion	The composition in mass fraction.
	composition.end	End of the composition.
Examples	−composition O2=0.23, NO=0, N=0, O=0, N2=0.77 composition.end	

Table 8.181: Init composition option

Composition Molar		
Syntax	−-composition molar $composition composition.end	
Description	Override the existing mcg composition (for applicable systems) molar fraction (can specify only species fractions that are not zero).	
Parameters	$compostion	The composition in molar fraction.
	composition.end	End of the composition.
Examples	−composition molar O2=0.21, N2=0.79 composition.end	

Table 8.182: Init composition molar option

8.19 UNS Options

The UNS input options section starts with the −−uns directive, followed by a series of uns options.

Name		
Syntax	−name $name	
Description	Name of flow case.	
Parameters	$name	Flow case name.
Examples	−name naca0012_laminar	

Table 8.183: UNS name option

Prefix		
Syntax	{−prefix | −prefix.starcd | −prefix.cgns} $prefix	
Description	Set prefix of the flow case.	
Parameters	prefix	Used for manual conversion (conversion and decomposition has been conducted using star2metis or cgns2metis).
	prefix.starcd	Use original Star-CD export grid files (conversion and decomposition is conducted).
	prefix.cgns	Use original CGNS export grid files (conversion and decomposition is conducted).
	$prefix	Prefix of flow case.
Examples	−prefix ./naca0012/NACA0012-Str-Laminar-Rey500	

Table 8.184: UNS prefix option

Block		
Syntax	−block #block	
Description	Select block from within a multi-block Stat-CD file.	
Parameters	#block	Star-CD file block number.
Examples	−block 2	

Table 8.185: UNS block option

Base		
Syntax	−base #base	
Description	Use only the specified base within the CGNS file. The user is referred to the CGNS format for further details.	
Parameters	#base	Base number.
Examples	−base 1	

Table 8.186: UNS base option

Zone		
Syntax	−zone #zone	
Description	Use only the specified zone within the CGNS file. The user is referred to the CGNS format for further details.	
Parameters	#zone	Zone number.
Examples	−zone 5	

Table 8.187: UNS zone option

CRC.Sections		
Syntax	−crc.sections #sections	
Description	Parallelize the initial CRC calculations. Different number of sections results in a different CRC (default value is 10).	
Parameters	#sections	Number of CRC sections.
Examples	−crc.sections 5	

Table 8.188: UNS crc.sections option

CRC.Version		
Syntax	−crc.version #version	
Description	Use a different crc calculation to match a specific **Arion** version.	
Parameters	#version	**Arion** version number.
Examples	−crc.version 2.43	

Table 8.189: UNS crc.version option

Key		
Syntax	−key $prefix	
Description	Save/load a key file. The file contains pre-calculated wall distance. If the file exists, the code reads it prior to execution. If the file does not exist, the code generates one.	
Parameters	$prefix	Prefix of the key file.
Examples	−key ./NACA0012-Str-Laminar-Rey500	

Table 8.190: UNS key option

Offset		
Syntax	−offset #x #y #z	
Description	Offset the grid.	
Parameters	#x	Offset in the x direction.
	#y	Offset in the y direction.
	#z	Offset in the z direction.
Examples	−offset 5 0 0	

Table 8.191: UNS offset option

Scale		
Syntax	−scale #x #y #z	
Description	Scale the grid.	
Parameters	#x	Scale in the x direction.
	#y	Scale in the y direction.
	#z	Scale in the z direction.
Examples	−scale 0.001 0.001 0.001	

Table 8.192: UNS scale option

Rotate [rad]		
Syntax	−rotate.{x|y|z}.rad #angle	
Description	Rotate the grid around an axis at an angle [rad] (can concatenate rotations) .	
Parameters	#angle	Rotation angle [rad].
Examples	−rotate.x.rad 0.1	
	−rotate.y.rad 0.1	

Table 8.193: UNS rotate option [rad]

Rotate [deg]		
Syntax	−rotate.{x|y|z}.deg #angle	
Description	Rotate the grid around an axis at an angle [deg] (can concatenate rotations).	
Parameters	#angle	Rotation angle [deg].
Examples	−rotate.x.deg 10	
	−rotate.z.deg -10	

Table 8.194: UNS rotate option [deg]

Log	
Syntax	−log $log [log-options] ...
Description	Report in log file. See Tables 8.196 and 8.204 for log options list.
Parameters	$log Log name.
Examples	−log coefficients cl cd

Table 8.195: UNS log option

UNS Log Options

Log type	Description
wet.surface	Wet surface area.
p.center.sum	Center of pressure.
p.center.x	Center of pressure x coordinate.
p.center.y	Center of pressure x coordinate.
p.center.z	Center of pressure x coordinate.
p.center.xFy	Center of pressure x coordinate.
p.center.xFz	Center of pressure x coordinate.
p.center.yFx	Pressure center sum(y * dFx) / Fx coordinate.
p.center.yFz	Pressure center sum(y * dFz) / Fz coordinate.
p.center.zFx	Pressure center sum(z * dFx) / Fx coordinate.
p.center.zFy	Pressure center sum(z * dFy) / Fy coordinate .
fx	X coordinate direction force.
fy	Y coordinate direction force.
fz	Z coordinate direction force.
fx.pressure	Force in x direction due to pressure.
fy.pressure	Force in y direction due to pressure.
fz.pressure	Force in z direction due to pressure.
fx.friction	Force in x direction due to friction.
fy.friction	Force in y direction due to friction.
fz.friction	Force in z direction due to friction.
lift	Lift force.
drag	Drag force.

mass.flow	Mass flow rate.
mx	Moment about X coordinate direction.
my	Moment about Y coordinate direction.
mz	Moment about Z coordinate direction.
cfx	X coordinate direction force coefficient.
cfy	Y coordinate direction force coefficient.
cfz	Z coordinate direction force coefficient.
cmx	Moment coefficient about X coordinate direction.
cmy	Moment coefficient about Y coordinate direction.
cmz	Moment coefficient about Z coordinate direction.
cl	Lift coefficient.
cd	Drag coefficient.

Table 8.196: UNS log options

8.19.1 Motion: DOF Options

This section describes the directive to control motion. The current version of **Arion** supports a user prescribed motion only. The motion can be prescribed as a constant motion, a harmonic motion, or a motion prescribed using a discrete table. Table values can be loaded from a separate text file using a `file` directive. If a discrete table is utilized, the values of the table are interpolated as required.

DOF Angular Motion		
Syntax	{−dof.angular.roll.rate.motion | −dof.angular.pitch.rate.motion | −dof.angular.yaw.rate.motion} {rad | deg} {#rate [sin #amplitude #frequency #shift] | table.t [#t0 #v0 #t1 #v1 ... #tN #vN | file #filename] table.end}	
Description	Set angular rate motion where rate = #rate + #amplitude * cos(#frequency * t + #shift).	
Parameters	rad	Set rate in radians per second.
	deg	Set rate in degrees per second.
	#rate	Angular rate.
	#amplitude	Oscillation amplitude.
	#frequency	Oscillation frequnecy.
	#shift	Oscillation phase shift.
	#t0 #t1 ... #tN	Discrete time.
	#v0 #v1 ... #vN	Discrete rate.
Examples	−dof.angular.roll.rate.motion deg 5	

Table 8.197: UNS DOF angular rate motion option

DOF Velocity Motion

Syntax	−dof.velocity.motion {#u #v #w | table.t [#t0 #u0 #v0 #w0 #t1 #u1 #v1 #w1 ... #tN #uN #vN #wN | file #filename] table.end}	
Description	Set motion velocity components.	
Parameters	#u	*X* coordinate direction velocity component.
	#v	*Y* coordinate direction velocity component.
	#w	*Z* coordinate direction velocity component.
	#t0 #t1 ... #tN	Discrete time.
	#u0 #u1 ... #uN	Discrete velocity.
	#v0 #v1 ... #vN	Discrete velocity.
	#w0 #w1 ... #wN	Discrete velocity.
Examples	−dof.velocity.motion 100 0 0	

Table 8.198: UNS DOF velocity motion option

DOF Acceleration Motion		
Syntax	−dof.acceleration.motion {#u #v #w | table.t [#t0 #ax0 #ay0 #az0 #t1 #ax1 #ay1 #az1 ... #tN #axN #ayN #azN | file #file-name] table.end}	
Description	Set motion acceleration components.	
Parameters	#ax	X coordinate direction acceleration component.
	#ay	Y coordinate direction acceleration component.
	#az	Z coordinate direction acceleration component.
	#t0 #t1 ... #tN	Discrete time.
	#ax0 #ax1 ... #axN	Discrete acceleration.
	#ay0 #ay1 ... #ayN	Discrete acceleration.
	#az0 #az1 ... #azN	Discrete acceleration.
Examples	−dof.acceleration.motion 0 0 10	

Table 8.199: UNS DOF acceleration motion option

DOF Angular Init		
Syntax	{−dof.angular.roll.rate.init | −dof.angular.pitch.rate.init | −dof.angular.yaw.rate.init} {rad | deg} #rate	
Description	Set initial value of angular rate.	
Parameters	rad	Set rate in radians per second.
	deg	Set rate in degrees per second.
	#rate	Angular rate.
Examples	−dof.angular.roll.rate.init deg 5	

Table 8.200: UNS DOF angular rate init option

DOF Velocity Init		
Syntax	−dof.velocity.init #u #v #w	
Description	Set initial velocity components.	
Parameters	#u	X coordinate direction velocity component.
	#v	Y coordinate direction velocity component.
	#w	Z coordinate direction velocity component.
Examples	−dof.velocity.init 100 20 0	

Table 8.201: UNS DOF velocity init option

DOF Acceleration Init		
Syntax	−dof.acceleration.init #u #v #w	
Description	Set initial acceleration components.	
Parameters	#ax	X coordinate direction acceleration component.
	#ay	Y coordinate direction acceleration component.
	#az	Z coordinate direction acceleration component.
Examples	−dof.acceleration.init 100 20 0	

Table 8.202: UNS DOF acceleration init option

DOF Motion Origin		
Syntax	−dof.origin.of.motion #x #y #z	
Description	Set the origin of motion.	
Parameters	#x	X coordinate direction component.
	#y	Y coordinate direction component.
	#z	Z coordinate direction component.
Examples	−dof.origin.of.motion 0 0 0	

Table 8.203: UNS DOF motion origin option

DOF Log Options

Log type	Description
dof.phi	Phi-euler angle in body frame.
dof.theta	Theta - euler angle in body frame.
dof.psi	Psi - euler angle in body frame.
dof.p	p - angular velocity in body frame.
dof.q	q - angular velocity in body frame.
dof.r	r - angular velocity in body frame.
dof.p.dot	dp/dt - angular acceleration in body frame.
dof.q.dot	dq/dt - angular acceleration in body frame.
dof.r.dot	dr/dt - angular acceleration in body frame.
dof.all.angular	All angualr information.
dof.dx	x - translational offset in observer frame.
dof.dy	y - translational offset in observer frame.
dof.dz	z - translational offset in observer frame.
dof.u	u - translational velocity in observer frame.
dof.v	v - translational velocity in observer frame.
dof.w	w - translational velocity in observer frame.
dof.u.dot	du/dt - translational acceleration in observer frame.
dof.v.dot	dv/dt - translational acceleration in observer frame.
dof.w.dot	dw/dt - translational acceleration in observer frame.
dof.all.translational	All translational information.
dof.origin.of.motion	Origin of motion vector.
dof.origin.of.motion.x	Origin of motion x - coordinate.

dof.origin.of.motion.y	Origin of motion y - coordinate.
dof.origin.of.motion.z	Origin of motion z - coordinate.

Table 8.204: DOF log options

8.20 Point Cloud Data

The point cloud data input options section starts with the −−pcd directive, followed by a series of point cloud data options. This section describes the point cloud data options. A simple example of the point cloud section is brought in Listing 8.2. The reader is referred to Section 6.3 for a brief description of the adaptation method that is implemented in the **Arion** code.

```
--pcd
  -name mach-sensor
  -prefix mach-cycle-0
  -mach
  -power 1
  -k 1
  -percentage 2.5
  -decay 0.3
  -spacing 0.5
```

Listing 8.2: Example of a pcd section

Name		
Syntax	−name $name	
Description	Set the name of the point cloud data.	
Parameters	$name	Point cloud data name.
Examples	−name pcd1	

Table 8.205: PCD name option

Prefix	
Syntax	−prefix $prefix
Description	The point cloud data plot-file will be saved using the path prefix with '.fvuns' suffix.
Parameters	$prefix Point cloud data prefix.
Examples	−prefix ./pcfd/pcd

Table 8.206: PCD prefix option

Mach	
Syntax	−mach
Description	Set the grid adaptation criterion to Mach number.
Parameters	N/A
Examples	−mach

Table 8.207: PCD mach option

Pressure	
Syntax	−pressure
Description	Set the grid adaptation criterion to pressure.
Parameters	N/A
Examples	−pressure

Table 8.208: PCD pressure option

Density	
Syntax	−density
Description	Set the grid adaptation criterion to density.
Parameters	N/A
Examples	−density

Table 8.209: PCD density option

Velocity.magnitude	
Syntax	−velocity.magnitude
Description	Set the grid adaptation criterion to velocity magnitude.
Parameters	N/A
Examples	−velocity.magnitude

Table 8.210: PCD velocity.magnitude option

Power		
Syntax	−power #power	
Description	Set the sensor power value (p). See Equation 6.1.	
Parameters	#power	Sensor power value.
Examples	−power 1	

Table 8.211: PCD power option

K		
Syntax	−k #k	
Description	Set the sensor constant (K). See Equation 6.1.	
Parameters	#k	Sensor constant.
Examples	−k 1	

Table 8.212: PCD K option

Percentage		
Syntax	−percentage #percentage	
Description	Set the refinement percentage (refine the top percentage% of the cells according to the value of sensor).	
Parameters	#percentage	Percentage of refined cells.
Examples	−percentage 2	

Table 8.213: PCD percentage option

Spacing		
Syntax	−spacing #spacing	
Description	Set the spacing.	
Parameters	#decay	Spacing (used by Pointwise to refine the mesh).
Examples	−spacing 0.5	

Table 8.214: PCD spacing option

Decay		
Syntax	−decay #decay	
Description	Set the decay.	
Parameters	#decay	Decay (used by Pointwise to refine the mesh).
Examples	−decay 0.3	

Table 8.215: PCD decay option

8.21 Run-time Options

Arion provides the capability to control the run while the application is running using the run-time options. This can be achieved by creating an ascii text file named 'arion.ins' containing one of the following run-time options:

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.
plot	Dump the FVUNS file as requested by the user at the next available opportunity.
save	Save a restart.
time.step [$system] #cfl	Change to the specified constant CFL or virtual time step of all the systems or the specified $system.
dt #dt	Change to the specified physical time step (for dual time simulations).

Table 8.216: Run-time options

8.21.1 Run-time Options Examples

The examples in Table 8.217 provide the complete Unix/Linux command to create the 'arion.ins' file with the required content. Alternatively, the user may use an editor to create the file and enter the content. Note, once **Arion** identifies the existence of the file 'arion.ins' and reads its content, the file is erased.

Run-time Options Examples

Command	Result
echo stop > arion.ins	Stop and save a restart.
echo plot > arion.ins	Dump the FVUNS file as requested by the user at the next available opportunity.
echo save > arion.ins	Save a restart.
echo time.step 50.0 > arion.ins	Change the CFL or time step to 50.0 (all systems).
echo time.step turbulent.kw.tnt 30.0 > arion.ins	Change the CFL or virtual time step for the k-ω-TNT turbulence model equations to 30.0.
echo dt 0.001	Change the physical time step to 0.001

Table 8.217: Run-time options examples

Bibliography

[1] Chul Park. Review of chemical-kinetic problems of future NASA missions. I - Earth entries. *Journal of Thermophysics and Heat Transfer*, 7(3):385–398, January 1993.

[2] Leonardo C Scalabrin. *Numerical Simulation of Weakly Ionized Hypersonic Flow Over Reentry Capsules*. PhD thesis, University of Michigan, 2007.

[3] Peter A Gnoffo, Roop N Gupta, and Judy L Shinn. Conservation equations and physical models for hypersonic air flows in thermal and chemical nonequilibrium, 1989.

[4] Alexander Burcat and Branko Ruscic. Third Millenium Ideal Gas and Condensed Phase Thermochemical Database for Combustion with Updates from Active Thermochemical Tables. Technical Report TAE 960, 2005.

[5] C R Wilke. A viscosity equation for gas mixtures. *Journal of Chemical Phys*, 18:517, 1950.

[6] F G Blottner, M Johnson, and M Ellis. Chemically Reacting Viscous Flow Program For Multi-Component Gas Mixtures. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA (United States), January 1971.

[7] Bonnie J Mcbride, Sanford Gordon, and Martin A Reno. Coefficients for calculating thermodynamic and transport properties of individual species. Technical Report NASA TM-4513, 1993.

[8] Joseph Oakland Hirschfelder and Charles F Curtiss. *Molecular theory of gases and liquids*. Wiley, 1964.

[9] Roop N Gupta, Jerrold M Yos, and Richard A Thompson. A review of reaction rates and thermodynamic and transport properties for the 11-species air model for chemical and thermal nonequilibrium calculations to 30000 K. 1989.

[10] Sanford Gordon and Bonnie J Mcbride. Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications. Part 1: Analysis, 1994.

[11] Michael G Dunn and Sang-Wook Kang. Theoretical and Experimental Studies of Reentry Plasmas. NASA Langley Research Center, 1973.

[12] Chul Park. *Nonequilibrium hypersonic aerothermodynamics*. John Wiley & Sons, New York, January 1989.

[13] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. 1992. AIAA paper 92 - 0439.

[14] 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.

[15] 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.

[16] 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.

[17] Philippe R Spalart and Christopher L Rumsey. Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA journal*, 45(10):2544–2553, 2007.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[25] SF Davis. Simplified second-order godunov-type methods. *SIAM Journal on Scientific and Statistical Computing*, 9(3):445–473, 1988.

[26] Meng-Sing Liou and Christopher J. Steffen Jr. A new flux splitting scheme. *Journal of Computational Physics*, 107(1):23–39, 1993.

[27] Meng-Sing Liou. A sequel to AUSM: $AUSM^{+}-up$. *Journal of Computational Physics*, 129:364–382, 1996.

[28] Meng-Sing Liou. A sequel to AUSM, part II: $AUSM^{+}-up$ for all speeds. *Journal of Computational Physics*, 214:137–170, 2006.

[29] M R Hestenes and E Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49, 1952.

[30] J K Reid. On the method of conjugate gradients for the solution of large sparse systems of linear equations. *Pro. the Oxford conference of institute of mathmatics and its applications*, 5(4):231–254, 1971.

[31] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[32] D A Knoll and D E Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193:357–397, 2004.

[33] P McHugh and D E Knoll. Inexact newton's method solutions to the incompressible navier-stokes and energy equations using standard and matrix-free implementations. In *11th Computational Fluid Dynamics Conference*, page 3332, 1993.

[34] Marian Nemec and David W Zingg. Newton-krylov algorithm for aerodynamic design using the navier-stokes equations. *AIAA journal*, 40(6):1146–1154, 2002.

[35] H Alcin, B Koobus, O Allain, and A Dervieux. Efficiency and scalability of a two-level Schwarz algorithm for incompressible and compressible flows. *International Journal for Numerical Methods in Fluids*, 72(1):69–89, May 2013.

[36] T K Sengupta, A Dipankar, and A K Rao. A new compact scheme for parallel computing using domain decomposition . *Journal of Computational Physics*, pages 654–677, 2007.

[37] G Wang and Danesh K Tafti. Performance Enhancement on Microprocessors with Hierarchical Memory Systems for Solving Large Sparse Linear Systems:.

The International Journal of High Performance Computing Applications, 13:63–79, 1999.

[38] Xiao-Chuan Cai and Marcus Sarkis. A Restricted Additive Schwarz Preconditioner for General Sparse Linear Systems. *SIAM Journal on Scientific Computing*, 21:792–797, 1999.

[39] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.