THE EZNSS CFD CODE
THEORETICAL AND USER'S MANUAL

Revision 330

ISCFDC Report 2015-06

Prepared by

Yuval Levy and Daniella Raveh

COPYRIGHT © October 2015

by

The authors and the Israeli CFD Center

Contents

List of Figures

List of Tables

Abstract

This theoretical manual describes the algorithms, methods, and input and output files of the EZNSS code. The current revision of the code, revision 262, contains a flow solver, a Chimera suite, an elliptic collar grid generator, a six degrees of freedom motion simulation module, an aeroelasticity module, and a spline suite to support the aeroelasticity module. The flow solver contains a central differencing method (Beam and Warming), a flux vector splitting method (Steger-Warming), a partially flux vector splitting method (F3D), and three flux difference splitting methods (HLLC, AUSM+up, and MAPS). It has explicit and implicit time marching, with our without dual-time-stepping, and with or without Runge-Kutta (3rd and 5th order). The flow solver has 6 RANS turbulence models, the algebraic Baldwin-Lomax, its Degani-Schiff variant, the R_t by Goldberg, the Spalart-Allmaras turbulence model, the $k-\omega$-TNT, and the $k-\omega$-SST. The $k-\omega$-TNT model contains flags that turns it into various types of a hybrid model, named the $XLES$, the $DDES$, and the $X-DDES$ models. The solver also has three Reynolds Stress models, the JH model, the stress omega model, and the MCL model (although included in the production version of the code, Reynolds stress models are not fully tested yet, use with care). The Chimera suite contains virtual body hole cutting and fail safe mechanisms for interpolations. The aeroelasticity module is based on the Karpel-Raveh modal approach, and the spline suite is due to Dr. Daniella Raveh. The code is fully parallel, using multi-level parallelism for the flow solver.

Chapter 1

Introduction

The EZNSS code is a multi-zone Euler/Navier-Stokes flow solver. The EZNSS code is capable of simulating complex, time-accurate flows about dynamically deforming geometries. This includes relative motion between surfaces as well as deformations caused due to aeroelastic effects. The code contains a number of implicit algorithms and a number of turbulence models. The code handles complex geometries using patched grids or the Chimera overset grid topology [1]. The code automatically handles various grid topologies such as C-C, C-H, and C-O grid topologies. When the grid topology is identified, the appropriate boundary conditions are set. To provide higher flexibility, the user may override the boundary conditions using the input file.

The code is written using Fortran 77 and Fortran 90. The use of Fortran 90 allows to use dynamic memory allocation. The program is parallelized using OpenMP and multi-level parallelism and may be run on any shared memory parallel computer with relative ease. Version 2.5.2 of the code contains the dual-time step capability, aeroelastic capabilities, six degrees of freedom motion simulation, and real gas effects.

The report is arranged in the following manner: Chapter 2 contains a description of the governing equations, numerical algorithms, and the turbulence models that are used in the code. Chapter 3 describes the supported mesh topologies, with examples. Chapter 4 contains a brief description of the boundary conditions. Chapter 5 entails the six degrees of freedom simulation module while Chapter 6 entails the aeroelasticity module. Parallelization is described in Chapter 7.

The appendices provide additional information as follows: Appendices A and B include details of the Jacobians. Appendices C and D include detailed descriptions of the input and data files, along with some usage guidance.

Chapter 2

Computational Methods

2.1 Introduction

Computer simulations are generally based upon the numerical solution of the model equations in a discretized mode. The accuracy of the computations depends mainly on the physical modeling, the numerical algorithm, and the quality of the computational mesh. This chapter contains a description of the governing equations for high Reynolds number fluid flow, the numerical algorithms that are used in the solution of the Euler or the Navier-Stokes equations, the boundary conditions associated with them, and turbulence models that are used to model the Reynolds stress tensor. The Reynolds stress tensor may be modeled by adopting the Boussinesq approximation or by using Reynolds stress models.

2.2 Governing Equations

The equations governing fluid flow are derived from the laws of conservation of mass, momentum, and 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

$$\frac{\partial Q}{\partial t} + \frac{\partial \left(E - E_v\right)}{\partial x} + \frac{\partial \left(F - F_v\right)}{\partial y} + \frac{\partial \left(G - G_v\right)}{\partial z} = 0 \tag{2.1}$$

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

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

The inviscid flux vectors, E, F, and G, are

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

and the viscous flux vectors, E_v, F_v, and G_v, are

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

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).

For computational purposes these equations are made dimensionless. [2] Characteristic values of all the variables can be formed from six basic reference quantities: a reference length, L, *e.g.* the chord of a wing or the diameter of a body of revolution; a reference velocity, a_∞, the speed of sound of the undisturbed flow; reference density and temperature, characteristic of the undisturbed flow, ρ_∞ and T_∞, respectively; and similarly, reference values for the coefficients of viscosity and thermal conductivity, μ_∞ and κ_∞, characteristic of the undisturbed flow. In addition, for real gas effects, there is a need to define a reference value for specific heat capacity in constant pressure c_p, c_{p_∞}.

The other reference values are derived from the basic ones. Thus time t is normalized by L/a_∞ and the pressure p and the energy e are normalized by ρa_∞^2. The temperature T is normalized by the reference value $\gamma_\infty T_\infty$. Table 2.1 contains a list of normalization relations. These relations assist in normalizing the equations.

2.2.1 Normalizing the Continuity Equation

The continuity equation in Cartesian coordinates is given by:

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u}{\partial x}+\frac{\partial \rho v}{\partial y}+\frac{\partial \rho w}{\partial z}=0 \tag{2.6}$$

Variable	Relation
Length	$x_i = \overline{x}_i L$
Velocity	$u_i = \overline{u}_i a_\infty$
Time	$t = \overline{t}\frac{L}{a_\infty}$
Density	$\rho = \overline{\rho}\rho_\infty$
Temperature	$T = \overline{T}\gamma T_\infty$
Pressure	$p = \overline{p}\rho_\infty a_\infty^2$
Energy	$e = \overline{e}\rho_\infty a_\infty^2$
Viscosity coefficient	$\mu = \overline{\mu}\mu_\infty$
Thermal conductivity coefficient	$\kappa = \overline{\kappa}\kappa_\infty$
Specific heat capacity in constant pressure	$c_p = \overline{c}_p c_{p_\infty}$

Table 2.1: List of dimensionless variables

Substituting the relations from Table 2.1, the continuity equation reads:

$$\frac{\partial\overline{\rho}\rho_\infty}{\partial\overline{t}\frac{L}{a_\infty}}+\frac{\partial\overline{\rho}\rho_\infty\overline{u}a_\infty}{\partial\overline{x}L}+\frac{\partial\overline{\rho}\rho_\infty\overline{v}a_\infty}{\partial\overline{y}L}+\frac{\partial\overline{\rho}\rho_\infty\overline{w}a_\infty}{\partial\overline{z}L}=0 \tag{2.7}$$

or:

$$\frac{\rho_\infty a_\infty}{L}\left(\frac{\partial\overline{\rho}}{\partial\overline{t}}+\frac{\partial\overline{\rho u}}{\partial\overline{x}}+\frac{\partial\overline{\rho v}}{\partial\overline{y}}+\frac{\partial\overline{\rho w}}{\partial\overline{z}}\right)=0 \tag{2.8}$$

Finally, the normalized continuity equation is the same as the original one where all variables, dependent and independent, are dimensionless.

$$\frac{\partial\overline{\rho}}{\partial\overline{t}}+\frac{\partial\overline{\rho u}}{\partial\overline{x}}+\frac{\partial\overline{\rho v}}{\partial\overline{y}}+\frac{\partial\overline{\rho w}}{\partial\overline{z}}=0 \tag{2.9}$$

2.2.2 Normalizing the Momentum Equation

As an example, the momentum equation in the x direction is considered:

$$\begin{aligned}\frac{\partial\rho u}{\partial t}+\frac{\partial\rho u^2}{\partial x}+\frac{\partial\rho uv}{\partial y}+\frac{\partial\rho uw}{\partial z} \;=\; & -\frac{\partial p}{\partial x}\\ & +\frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x}-\frac{2}{3}\mu\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)\right]\\ & +\frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]\\ & +\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]\end{aligned}$$

Substituting the relations from Table 2.1, the momentum equation reads:

$$\begin{aligned}\frac{\partial\overline{\rho}\rho_\infty\overline{u}a_\infty}{\partial\overline{t}\frac{L}{a_\infty}}+\frac{\partial\overline{\rho}\rho_\infty\overline{u}^2a_\infty^2}{\partial\overline{x}L}+\frac{\partial\overline{\rho}\rho_\infty\overline{u}a_\infty\overline{v}a_\infty}{\partial\overline{y}L}+\frac{\partial\overline{\rho}\rho_\infty\overline{u}a_\infty\overline{w}a_\infty}{\partial\overline{z}L} &= -\frac{\partial\overline{p}\rho_\infty a_\infty^2}{\partial\overline{x}L}\\ +\frac{\partial}{\partial\overline{x}L}\left[2\overline{\mu}\mu_\infty\frac{\partial\overline{u}a_\infty}{\partial\overline{x}L}-\frac{2}{3}\overline{\mu}\mu_\infty\left(\frac{\partial\overline{u}a_\infty}{\partial\overline{x}L}+\frac{\partial\overline{v}a_\infty}{\partial\overline{y}L}+\frac{\partial\overline{w}a_\infty}{\partial\overline{z}L}\right)\right]&\\ +\frac{\partial}{\partial\overline{y}L}\left[\overline{\mu}\mu_\infty\left(\frac{\partial\overline{u}a_\infty}{\partial\overline{y}L}+\frac{\partial\overline{v}a_\infty}{\partial\overline{x}L}\right)\right]&\\ +\frac{\partial}{\partial\overline{z}L}\left[\overline{\mu}\mu_\infty\left(\frac{\partial\overline{u}a_\infty}{\partial\overline{z}L}+\frac{\partial\overline{w}a_\infty}{\partial\overline{x}L}\right)\right]&\end{aligned}$$

Multiplying both sides of the equation with $\frac{L}{\rho_\infty a_\infty^2}$ results in:

$$\begin{aligned}\frac{\partial\overline{\rho u}}{\partial\overline{t}}+\frac{\partial\overline{\rho u^2}}{\partial\overline{x}}+\frac{\partial\overline{\rho uv}}{\partial\overline{y}}+\frac{\partial\overline{\rho uw}}{\partial\overline{z}} &= -\frac{\partial\overline{p}}{\partial\overline{x}}+\frac{\mu_\infty}{\rho_\infty a_\infty L}\times \\ &\left\{\frac{\partial}{\partial\overline{x}}\left[2\overline{\mu}\frac{\partial\overline{u}}{\partial\overline{x}}-\frac{2}{3}\overline{\mu}\left(\frac{\partial\overline{u}}{\partial\overline{x}}+\frac{\partial\overline{v}}{\partial\overline{y}}+\frac{\partial\overline{w}}{\partial\overline{z}}\right)\right]\right. \\ &+\frac{\partial}{\partial\overline{y}}\left[\overline{\mu}\left(\frac{\partial\overline{u}}{\partial\overline{y}}+\frac{\partial\overline{v}}{\partial\overline{x}}\right)\right] \\ &\left.+\frac{\partial}{\partial\overline{z}}\left[\overline{\mu}\left(\frac{\partial\overline{u}}{\partial\overline{z}}+\frac{\partial\overline{w}}{\partial\overline{x}}\right)\right]\right\}\end{aligned}$$

Realizing that the term $\frac{\mu_\infty}{\rho_\infty a_\infty L}$ equals $\frac{M}{Re}$ where $Re=\frac{\rho_\infty u_\infty L}{\mu_\infty}$ results in:

$$\begin{aligned}\frac{\partial\overline{\rho u}}{\partial\overline{t}}+\frac{\partial\overline{\rho u^2}}{\partial\overline{x}}+\frac{\partial\overline{\rho uv}}{\partial\overline{y}}+\frac{\partial\overline{\rho uw}}{\partial\overline{z}} &= -\frac{\partial\overline{p}}{\partial\overline{x}}+\frac{M}{Re}\times \\ &\left\{\frac{\partial}{\partial\overline{x}}\left[2\overline{\mu}\frac{\partial\overline{u}}{\partial\overline{x}}-\frac{2}{3}\overline{\mu}\left(\frac{\partial\overline{u}}{\partial\overline{x}}+\frac{\partial\overline{v}}{\partial\overline{y}}+\frac{\partial\overline{w}}{\partial\overline{z}}\right)\right]\right. \\ &+\frac{\partial}{\partial\overline{y}}\left[\overline{\mu}\left(\frac{\partial\overline{u}}{\partial\overline{y}}+\frac{\partial\overline{v}}{\partial\overline{x}}\right)\right] \\ &\left.+\frac{\partial}{\partial\overline{z}}\left[\overline{\mu}\left(\frac{\partial\overline{u}}{\partial\overline{z}}+\frac{\partial\overline{w}}{\partial\overline{x}}\right)\right]\right\}\end{aligned}$$

2.2.2.1 Turbulent Viscosity Coefficient

In turbulent flows the viscosity coefficient is composed of the molecular viscosity coefficient, μ_l, and the turbulent viscosity coefficient, μ_t such that:

$$\mu=\mu_l+\mu_t \tag{2.10}$$

with the dimensionless viscosity coefficient being:

$$\overline{\mu}=\overline{\mu}_l+\overline{\mu}_t \tag{2.11}$$

The molecular viscosity coefficient is evaluated using Sutherland's Law while the turbulent viscosity coefficient is obtained through the solution of the turbulence model

equations.

2.2.3 Normalizing the Energy Equation

In Cartesian coordinates, the energy equation reads:

$$\begin{aligned}\frac{\partial e}{\partial t}+\frac{\partial\left(eu+p\right)}{\partial x}+\frac{\partial\left(ev+p\right)}{\partial y}+\frac{\partial\left(ew+p\right)}{\partial z}=\\ \frac{\partial}{\partial x}\left(u\tau_{xx}+v\tau_{xy}+w\tau_{xz}+\kappa\frac{\partial T}{\partial x}\right)\\ \frac{\partial}{\partial y}\left(u\tau_{yx}+v\tau_{yy}+w\tau_{yz}+\kappa\frac{\partial T}{\partial y}\right)\\ \frac{\partial}{\partial z}\left(u\tau_{zx}+v\tau_{zy}+w\tau_{zz}+\kappa\frac{\partial T}{\partial z}\right)\end{aligned} \tag{2.12}$$

Based on the normalization conducted for the momentum equation, the shear stresses' dimensionless form is:

$$\tau_{ij}=\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{ij} \tag{2.13}$$

Substituting the relations from Table 2.1, the energy equation reads:

$$\begin{aligned}\frac{\partial\overline{e}\rho_\infty a_\infty^2}{\partial\overline{t}\frac{L}{a_\infty}}+\frac{\partial\left(\overline{e}\rho_\infty a_\infty^2\overline{u}a_\infty+\overline{p}\rho_\infty a_\infty^2\right)}{\partial\overline{x}L}+\\ \frac{\partial\left(\overline{e}\rho_\infty a_\infty^2\overline{v}a_\infty+\overline{p}\rho_\infty a_\infty^2\right)}{\partial\overline{y}L}+\frac{\partial\left(\overline{e}\rho_\infty a_\infty^2\overline{w}a_\infty+\overline{p}\rho_\infty a_\infty^2\right)}{\partial\overline{z}L}=\\ \frac{\partial}{\partial\overline{x}L}\left(\overline{u}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{xx}+\overline{v}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{xy}+\overline{w}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{xz}+\overline{\kappa}\kappa_\infty\frac{\partial\overline{T}\gamma T_\infty}{\partial\overline{x}L}\right)\\ \frac{\partial}{\partial\overline{y}L}\left(\overline{u}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{yx}+\overline{v}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{yy}+\overline{w}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{yz}+\overline{\kappa}\kappa_\infty\frac{\partial\overline{T}\gamma T_\infty}{\partial\overline{y}L}\right)\\ \frac{\partial}{\partial\overline{z}L}\left(\overline{u}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{zx}+\overline{v}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{zy}+\overline{w}a_\infty\frac{\mu_\infty a_\infty}{L}\overline{\tau}_{zz}+\overline{\kappa}\kappa_\infty\frac{\partial\overline{T}\gamma T_\infty}{\partial\overline{z}L}\right)\end{aligned} \tag{2.14}$$

Multiplying both sides of the equation with $\frac{L}{\rho_\infty a_\infty^3}$ results in:

$$\frac{\partial \overline{e}}{\partial \overline{t}} + \frac{\partial\left(\overline{eu}+\overline{p}\right)}{\partial \overline{x}} + \frac{\partial\left(\overline{ev}+\overline{p}\right)}{\partial \overline{y}} + \frac{\partial\left(\overline{ew}+\overline{p}\right)}{\partial \overline{z}} = \frac{\mu_\infty}{\rho_\infty a_\infty L} \times \left[\frac{\partial}{\partial \overline{x}}\left(\overline{u\tau}_{xx} + \overline{v\tau}_{xy} + \overline{w\tau}_{xz} + \frac{\kappa_\infty \gamma T_\infty}{a_\infty^2 \mu_\infty}\overline{\kappa}\frac{\partial \overline{T}}{\partial \overline{x}}\right)\right. \frac{\partial}{\partial \overline{y}}\left(\overline{u\tau}_{yx} + \overline{v\tau}_{yy} + \overline{w\tau}_{yz} + \frac{\kappa_\infty \gamma T_\infty}{a_\infty^2 \mu_\infty}\overline{\kappa}\frac{\partial \overline{T}}{\partial \overline{y}}\right) \left.\frac{\partial}{\partial \overline{z}}\left(\overline{u\tau}_{zx} + \overline{v\tau}_{zy} + \overline{w\tau}_{zz} + \frac{\kappa_\infty \gamma T_\infty}{a_\infty^2 \mu_\infty}\overline{\kappa}\frac{\partial \overline{T}}{\partial \overline{z}}\right)\right] \tag{2.15}$$

As with the momentum equation, the term $\frac{\mu_\infty}{\rho_\infty a_\infty L} = \frac{M}{Re}$. Realizing that $a_\infty^2 = \gamma R T_\infty$, the term $\frac{\kappa_\infty \gamma T_\infty}{a_\infty^2 \mu_\infty}$ becomes $\frac{\kappa_\infty}{R\mu_\infty}$. The dimensionless energy equation then becomes:

$$\frac{\partial \overline{e}}{\partial \overline{t}} + \frac{\partial\left(\overline{eu}+\overline{p}\right)}{\partial \overline{x}} + \frac{\partial\left(\overline{ev}+\overline{p}\right)}{\partial \overline{y}} + \frac{\partial\left(\overline{ew}+\overline{p}\right)}{\partial \overline{z}} = \frac{M}{Re} \times \left[\frac{\partial}{\partial \overline{x}}\left(\overline{u\tau}_{xx} + \overline{v\tau}_{xy} + \overline{w\tau}_{xz} + \frac{\kappa_\infty}{R\mu_\infty}\overline{\kappa}\frac{\partial \overline{T}}{\partial \overline{x}}\right)\right. \frac{\partial}{\partial \overline{y}}\left(\overline{u\tau}_{yx} + \overline{v\tau}_{yy} + \overline{w\tau}_{yz} + \frac{\kappa_\infty}{R\mu_\infty}\overline{\kappa}\frac{\partial \overline{T}}{\partial \overline{y}}\right) \left.\frac{\partial}{\partial \overline{z}}\left(\overline{u\tau}_{zx} + \overline{v\tau}_{zy} + \overline{w\tau}_{zz} + \frac{\kappa_\infty}{R\mu_\infty}\overline{\kappa}\frac{\partial \overline{T}}{\partial \overline{z}}\right)\right] \tag{2.16}$$

2.2.3.1 Turbulent Heat Conduction Coefficient

In turbulent flows the heat conduction coefficient is composed of the thermal heat conduction coefficient, κ_l, and the turbulent heat conduction coefficient, κ_t such that:

$$\kappa = \kappa_l + \kappa_t \tag{2.17}$$

with the dimensionless conductivity coefficient being:

$$\overline{\kappa} = \overline{\kappa}_l + \overline{\kappa}_t \tag{2.18}$$

The thermal heat conduction coefficient is evaluated using a relation that is similar to the Sutherland Law while the turbulent heat conduction coefficient is evaluated based on the turbulent viscosity coefficient, μ_t and the turbulent Prandtl number:

$$Pr_t = \frac{\mu_t c_p}{\kappa_t} \tag{2.19}$$

The turbulent Prandtl number itself may be assumed constant ($Pr_t \approx 0.9$) or it can be evaluated using empirical relations. Hence, the turbulent heat conduction coefficient may be evaluated using:

$$\kappa_t = \frac{\mu_t c_p}{Pr_t} \tag{2.20}$$

When using dimensionless form, one has to evaluate $\overline{\kappa}_t$ as follows:

$$Pr_t = \frac{\overline{\mu}_t \mu_\infty \overline{c_p} c_{p_\infty}}{\overline{\kappa}_t \kappa_\infty} = \frac{\mu_\infty c_{p_\infty}}{\kappa_\infty} \frac{\overline{\mu}_t \overline{c_p}}{\overline{\kappa}_t} \tag{2.21}$$

or:

$$Pr_t = Pr_\infty \frac{\overline{\mu}_t \overline{c_p}}{\overline{\kappa}_t} \tag{2.22}$$

and therefore:

$$\overline{\kappa}_t = Pr_\infty \frac{\overline{\mu}_t \overline{c_p}}{Pr_t} \tag{2.23}$$

2.2.4 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 e_I. For most problems in gas dynamics, it is possible to assume a perfect gas. This assumption is also adopted in the EZNSS code. Hence, in Equations (2.3) and (2.5), the pressure and temperature are obtained from the equation of state for a perfect gas

$$p = \rho RT = \rho \left(\gamma - 1\right) e_I \tag{2.24}$$

where R is the gas constant ($R = 287.0$ for air), e_I is the internal energy of the gas, and γ is the ratio of specific heats (c_p/c_v). In terms of the flow variables, the pressure

and temperature are calculated using:

$$\begin{aligned} p &= (\gamma-1)\left[e-\frac{1}{2}\rho\left(u^2+v^2+w^2\right)\right] \\ T &= \frac{\gamma-1}{R}\left[\frac{e}{\rho}-\frac{1}{2}\left(u^2+v^2+w^2\right)\right] \end{aligned} \tag{2.25}$$

Following the normalization process that is described in the previous sections, the equation of state is normalized as follows:

$$\overline{p}\rho_\infty a_\infty^2 = \overline{\rho}\rho_\infty R\overline{T}\gamma T_\infty \tag{2.26}$$

Once again, realizing that $a_\infty^2 = \gamma R T_\infty$, the normalized equation of state becomes:

$$\overline{p} = \overline{\rho}\overline{T} \tag{2.27}$$

and the normalized pressure and temperature are evaluated using:

$$\begin{aligned} \overline{p} &= (\gamma-1)\left[\overline{e}-\frac{1}{2}\overline{\rho}\left(\overline{u}^2+\overline{v}^2+\overline{w}^2\right)\right] \\ \overline{T} &= \frac{\overline{p}}{\overline{\rho}} \end{aligned} \tag{2.28}$$

2.2.5 Constitutive Relations

In addition to the equation of state, it is also necessary to establish relations for the coefficients of viscosity, μ, and thermal conductivity, κ. The EZNSS code utilizes the Sutherland Formulae to evaluate the coefficients as follows:

$$\begin{aligned} \mu &= 1.458\times10^{-6}\frac{T^{\frac{3}{2}}}{T+110.4} \\ \kappa &= 2.495\times10^{-3}\frac{T^{\frac{3}{2}}}{T+194} \end{aligned} \tag{2.29}$$

Using the normalization definitions from Table 2.1, the normalized Sutherland Formulae become:

$$\begin{aligned} \overline{\mu} &= \left(1+\frac{110.4}{T_\infty}\right)\frac{\left(\gamma\overline{T}\right)^{\frac{3}{2}}}{\gamma\overline{T}+\frac{110.4}{T_\infty}} \\ \overline{\kappa} &= \left(1+\frac{194}{T_\infty}\right)\frac{\left(\gamma\overline{T}\right)^{\frac{3}{2}}}{\gamma\overline{T}+\frac{194}{T_\infty}} \end{aligned} \tag{2.30}$$

2.2.6 Real Gas Effects

For high speed flows, air no longer behaves as a calorically perfect gas. The simplest model is to account for the changes in specific heats by using polynomial relations. The following model uses two polynomials, one for temperatures bellow $T = 1000K$ and one for temperatures above $T = 1000K$. Both polynomials take the form:

$$\frac{c_p}{R} = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4 \tag{2.31}$$

The default polynomial coefficients are given by:

$$\begin{aligned} a_0 &= 3.56839620E+00 \\ a_1 &= -6.78729429E-04 \\ a_2 &= 1.55371476E-06 \\ a_3 &= -3.29937060E-12 \\ a_4 &= -4.66395387E-13 \end{aligned} \tag{2.32}$$

for temperatures bellow $T = 1000K$ and

$$\begin{aligned} a_0 &= 3.08792717E+00 \\ a_1 &= 1.24597184E-03 \\ a_2 &= -4.23718945E-07 \\ a_3 &= 6.74774789E-11 \\ a_4 &= -3.97076972E-15 \end{aligned} \tag{2.33}$$

for temperatures above $T = 1000K$. Once c_p is calculated, the ratio of specific heats may be evaluated using:

$$\gamma = \frac{c_p}{c_p - R} \tag{2.34}$$

The default polynomial coefficients may be changed using a user input file as described in Appendix D.8.

2.2.7 Normalized Set of Equation

The normalization process results in a set of dimensionless equations that is similar to the dimensional ones with two exceptions. The factor M/Re appears in front of the viscous terms, where Re is the Reynolds number

$$Re = \frac{\rho_\infty u_\infty L}{\mu_\infty} \tag{2.35}$$

Also, the term κ in Eq. (2.5) becomes $\frac{\kappa_\infty}{R\mu_\infty}\overline{\kappa}$. The rest of the variables appear in the same way with the addition of a bar and the set of normalized equations takes the form

$$\frac{\partial \overline{Q}}{\partial \overline{t}} + \frac{\partial \overline{E}}{\partial \overline{x}} + \frac{\partial \overline{F}}{\partial \overline{y}} + \frac{\partial \overline{G}}{\partial \overline{z}} = \frac{M}{Re}\left(\frac{\partial \overline{E}_v}{\partial \overline{x}} + \frac{\partial \overline{F}_v}{\partial \overline{y}} + \frac{\partial \overline{G}_v}{\partial \overline{z}}\right) \tag{2.36}$$

Note: From here on the bars above all variables are omitted and a normalized physical domain with normalized flow variables is assumed.

2.3 General Curvilinear Coordinates

The Cartesian form of the equations is not suitable for handling complex body geometries. For example, application of the boundary conditions are not compatible with the Cartesian form. To enhance compatibility the physical domain is transformed into a computational one by introducing a coordinate transformation.[2–4] In general

terms, this transformation takes the form

$$\begin{aligned} \tau &= t \\ \xi &= \xi\left(x, y, z, t\right) \\ \eta &= \eta\left(x, y, z, t\right) \\ \zeta &= \zeta\left(x, y, z, t\right) \end{aligned} \tag{2.37}$$

The transformation brings the body surface $f\left(x, y, z\right) = 0$ onto one computational plane (in the current formulation $\zeta = 1$). Therefore the coordinate ζ extends radially from the body surface and the other two coordinates, ξ and η lie on the surface normal to ζ. The computational domain is equi-spaced (usually chosen to be $\delta\xi = \delta\eta = \delta\zeta = 1$, for convenience), so the differencing is simplified. To apply the transformation to the model equations the chain rule is used

$$\begin{bmatrix} \frac{\partial}{\partial t} \\ \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix} = \begin{bmatrix} 1 & \xi_t & \eta_t & \zeta_t \\ 0 & \xi_x & \eta_x & \zeta_x \\ 0 & \xi_y & \eta_y & \zeta_y \\ 0 & \xi_z & \eta_z & \zeta_z \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \tau} \\ \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{bmatrix} \tag{2.38}$$

The Jacobian of the transformation J is given by the determinant

$$\begin{aligned} J^{-1} = \frac{\partial(t,x,y,z)}{\partial(\tau,\xi,\eta,\zeta)} &= \begin{vmatrix} 1 & x_\tau & y_\tau & z_\tau \\ 0 & x_\xi & y_\xi & z_\xi \\ 0 & x_\eta & y_\eta & z_\eta \\ 0 & x_\zeta & y_\zeta & z_\zeta \end{vmatrix} \\ J^{-1} &= x_\xi\left(y_\eta z_\zeta - y_\zeta z_\eta\right) - y_\xi\left(x_\eta z_\zeta - x_\zeta z_\eta\right) + z_\xi\left(x_\eta y_\zeta - x_\zeta y_\eta\right) \end{aligned} \tag{2.39}$$

The metrics of the transformation are given by

$$\begin{aligned}
\xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z \\
\xi_x &= J\left(y_\eta z_\zeta - z_\eta y_\zeta\right) \\
\xi_y &= -J\left(x_\eta z_\zeta - x_\zeta z_\eta\right) \\
\xi_z &= J\left(x_\eta y_\zeta - x_\zeta y_\eta\right) \\
\eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z \\
\eta_x &= -J\left(y_\xi z_\zeta - y_\zeta z_\xi\right) \\
\eta_y &= J\left(x_\xi z_\zeta - x_\zeta z_\xi\right) \\
\eta_z &= -J\left(x_\eta y_\zeta - x_\zeta y_\eta\right) \\
\zeta_t &= -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z \\
\zeta_x &= J\left(y_\xi z_\eta - y_\eta z_\xi\right) \\
\zeta_y &= -J\left(x_\xi z_\eta - x_\eta z_\xi\right) \\
\zeta_z &= J\left(x_\xi y_\eta - x_\eta y_\xi\right)
\end{aligned} \tag{2.40}$$

Applying the coordinate transformation results in a new set of equations that maintains the conservation-law form of the original equations. Equation (2.36) becomes

$$\frac{\partial \hat{Q}}{\partial \tau} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} + \frac{\partial \hat{G}}{\partial \zeta} = \frac{1}{Re}\left(\frac{\partial \hat{E}_v}{\partial \xi} + \frac{\partial \hat{F}_v}{\partial \eta} + \frac{\partial \hat{G}_v}{\partial \zeta}\right) \tag{2.41}$$

where

$$\begin{aligned}
\hat{Q} &= \tfrac{1}{J} \quad Q \\
\hat{E} &= \tfrac{1}{J} \quad \left(Q\xi_t + E\xi_x + F\xi_y + G\xi_z\right) \\
\hat{F} &= \tfrac{1}{J} \quad \left(Q\eta_t + E\eta_x + F\eta_y + G\eta_z\right) \\
\hat{G} &= \tfrac{1}{J} \quad \left(Q\zeta_t + E\zeta_x + F\zeta_y + G\zeta_z\right) \\
\hat{E}_v &= \tfrac{1}{J} \quad \left(E_v\xi_x + F_v\xi_y + G_v\xi_z\right) \\
\hat{F}_v &= \tfrac{1}{J} \quad \left(E_v\eta_x + F_v\eta_y + G_v\eta_z\right) \\
\hat{G}_v &= \tfrac{1}{J} \quad \left(E_v\zeta_x + F_v\zeta_y + G_v\zeta_z\right)
\end{aligned} \tag{2.42}$$

Alternatively, the new dependent variables, the inviscid flux vectors, and the viscous flux vectors can be expressed in terms of the original dependent variables and metrics

as

$$\hat{Q} = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \quad \hat{E} = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (e+p)\,U - \xi_t p \end{bmatrix}$$

$$\hat{F} = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ (e+p)\,V - \eta_t p \end{bmatrix} \quad \hat{G} = \frac{1}{J}\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ (e+p)\,W - \zeta_t p \end{bmatrix} \tag{2.43}$$

$$\hat{E}_v = \tfrac{1}{ReJ}\begin{bmatrix} 0 \\ \xi_x\tau_{xx} + \xi_y\tau_{xy} + \xi_z\tau_{xz} \\ \xi_x\tau_{yx} + \xi_y\tau_{yy} + \xi_z\tau_{yz} \\ \xi_x\tau_{zx} + \xi_y\tau_{zy} + \xi_z\tau_{zz} \\ \xi_x\beta_x + \xi_y\beta_y + \xi_z\beta_z \end{bmatrix}$$

$$\hat{F}_v = \tfrac{1}{ReJ}\begin{bmatrix} 0 \\ \eta_x\tau_{xx} + \eta_y\tau_{xy} + \eta_z\tau_{xz} \\ \eta_x\tau_{yx} + \eta_y\tau_{yy} + \eta_z\tau_{yz} \\ \eta_x\tau_{zx} + \eta_y\tau_{zy} + \eta_z\tau_{zz} \\ \eta_x\beta_x + \eta_y\beta_y + \eta_z\beta_z \end{bmatrix} \tag{2.44}$$

$$\hat{G}_v = \tfrac{1}{ReJ}\begin{bmatrix} 0 \\ \zeta_x\tau_{xx} + \zeta_y\tau_{xy} + \zeta_z\tau_{xz} \\ \zeta_x\tau_{yx} + \zeta_y\tau_{yy} + \zeta_z\tau_{yz} \\ \zeta_x\tau_{zx} + \zeta_y\tau_{zy} + \zeta_z\tau_{zz} \\ \zeta_x\beta_x + \zeta_y\beta_y + \zeta_z\beta_z \end{bmatrix}$$

where U, V, and W are the contravariant velocities and are given by the relation

$$\begin{bmatrix} U-\xi_t \\ V-\eta_t \\ W-\zeta_t \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.45}$$

Each element containing derivatives with respect to the original independent variables t, x, y, z has to be expanded according to the chain rule so that the final form of the equations includes only derivatives with respect to the transformed independent variables τ, ξ, η, and ζ.

2.4 Implicit Numerical Methods

The fine grid spacing required to resolve the normal viscous terms close to the body surface 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. The linearization procedure is simple since the equations are written in conservation-law form. It is done by utilizing local Taylor series expansions of the vectors $\hat{E}$, $\hat{F}$, and $\hat{G}$ about $\hat{Q}$ [5, 6]

$$\begin{aligned} \hat{E}^{n+1} &= \hat{E}^n + \hat{A}^n\left(\hat{Q}^{n+1}-\hat{Q}^n\right) + O\left(\Delta t^2\right) \\ \hat{F}^{n+1} &= \hat{F}^n + \hat{B}^n\left(\hat{Q}^{n+1}-\hat{Q}^n\right) + O\left(\Delta t^2\right) \\ \hat{G}^{n+1} &= \hat{G}^n + \hat{C}^n\left(\hat{Q}^{n+1}-\hat{Q}^n\right) + O\left(\Delta t^2\right) \end{aligned} \tag{2.46}$$

where $\hat{A}$, $\hat{B}$, and $\hat{C}$ are the Jacobian matrices. The superscript n denotes evaluation at the n^{th} time step where $t = n\Delta t$. The elements of $\hat{A}$, $\hat{B}$, and $\hat{C}$, are given in Appendix A.

A similar Taylor series expansion of the viscous flux vectors results in the viscous

Jacobian matrices, $\hat{A}_v$, $\hat{B}_v$, and $\hat{C}_v$. However, the viscous Jacobians contain mixed derivatives and therefore the block tridiagonal form of the linear systems is spoiled. In first order, the mixed derivatives can be neglected with no loss of accuracy and simpler matrices, that are in fact based on the thin-layer approximation [7] can be used. The new matrices, denoted by $\hat{A}_v^\xi$, $\hat{B}_v^\eta$, and $\hat{C}_v^\zeta$, contain derivatives only in the direction denoted by the superscript. The elements of $\hat{A}_v^\xi$, $\hat{B}_v^\eta$, and $\hat{C}_v^\zeta$ are also given in Appendix A.

Applying the first-order Euler implicit formula to Eq.(2.36) and incorporating the linearizations results in a linear system with first-order time accuracy

$$\left\{ I + h\left[\frac{\partial}{\partial\xi}\hat{A} + \frac{\partial}{\partial\eta}\hat{B} + \frac{\partial}{\partial\zeta}\hat{C} - \frac{1}{Re}\left(\frac{\partial}{\partial\xi}\hat{A}_v^\xi + \frac{\partial}{\partial\eta}\hat{B}_v^\eta + \frac{\partial}{\partial\zeta}\hat{C}_v^\zeta \right) \right] \right\}^n \Delta\hat{Q}^n = -\Delta t \left[\frac{\partial\hat{E}}{\partial\xi} + \frac{\partial\hat{F}}{\partial\eta} + \frac{\partial\hat{G}}{\partial\zeta} - \frac{1}{Re}\left(\frac{\partial\hat{E}_v}{\partial\xi} + \frac{\partial\hat{F}_v}{\partial\eta} + \frac{\partial\hat{G}_v}{\partial\zeta} + \right) \right]^n + O\left(\Delta t^2\right) \quad (2.47)$$

where I is the identity matrix, $h = \Delta t$, $\Delta\hat{Q}^n = \hat{Q}^{n+1} - \hat{Q}^n$, and $\partial/\partial\xi$, $\partial/\partial\eta$, and $\partial/\partial\zeta$ are approximated by finite differencing.

2.5 Beam and Warming Algorithm

The linear system, formed after replacing the spatial derivatives in Eq. (2.47) with central finite difference approximations, is a block hepta-diagonal matrix with nonadjacent diagonals. A direct solution of this system requires inversion of a block matrix of the size of the computational mesh. Because of the large band of the system, the direct inversion process is a very costly one and calls for a simplification. Approximate factorization of the left-hand-side operator reduces the inversion to a sequence of one-dimensional inversions, without altering the formal accuracy of Eq. (2.47) [5]. If central differencing is used to approximate the spatial operators, the resulting one-dimensional operators are block tridiagonal matrices. Beam and Warming [5] developed a factored algorithm applicable to the Euler gasdynamic equations in two dimensions. They later included the viscous flux terms and applied the scheme to the two-dimensional compressible Navier-Stokes equations [6]. The following form of the

algorithm is its extension to three dimensions.

$$\left(I + h\delta_\xi \hat{A} - hRe^{-1}\bar{\delta}_\xi \hat{A}_v^\xi\right)^n \left(I + h\delta_\eta \hat{B} - hRe^{-1}\bar{\delta}_\eta \hat{B}_v^\eta\right)^n \left(I + h\delta_\zeta \hat{C} - hRe^{-1}\bar{\delta}_\zeta \hat{C}_v^\zeta\right)^n \Delta\hat{Q}^n = \hat{R}^n \tag{2.48}$$

where $\hat{R}^n$ is

$$\hat{R}^n = -\Delta t \left[\delta_\xi \hat{E} + \delta_\eta \hat{F} + \delta_\zeta \hat{G} - Re^{-1}\left(\bar{\delta}_\xi \hat{E}_v + \bar{\delta}_\eta \hat{F}_v + \bar{\delta}_\zeta \hat{G}_v\right)\right]^n \tag{2.49}$$

The above formulation applies to both Euler implicit first-order and trapezoidal second-order time accuracy as follows: setting the quantity h equal to Δt yields the first-order-accurate Euler implicit form, while setting h equal to $\Delta t/2$ yields the second-order-accurate trapezoidal form. The δ operators denote central differencing and the $\bar{\delta}$ operators denote a midpoint operator used in order to preserve the block tridiagonal form. The numerical scheme then has second-order spatial accuracy and either first- or second-order time accuracy.

The procedure of advancing the solution from time-step n to time-step $n+1$ requires a series of three one-dimensional block-tridiagonal inversions

$$\begin{aligned}
\left(I + h\delta_\xi \hat{A} - hRe^{-1}\bar{\delta}_\xi \hat{A}_v^\xi\right)^n \Delta\hat{Q}^1 &= \hat{R}^n \\
\left(I + h\delta_\eta \hat{B} - hRe^{-1}\bar{\delta}_\eta \hat{B}_v^\eta\right)^n \Delta\hat{Q}^2 &= \Delta\hat{Q}^1 \\
\left(I + h\delta_\zeta \hat{C} - hRe^{-1}\bar{\delta}_\zeta \hat{C}_v^\zeta\right)^n \Delta\hat{Q}^n &= \Delta\hat{Q}^2 \\
\hat{Q}^{n+1} &= \hat{Q}^n + \Delta\hat{Q}^n
\end{aligned} \tag{2.50}$$

Each inversion process is set up in a way that takes advantage of the pipelining capability of supercomputers. Although the inversion is in itself recursive, the factorized scheme can be optimized by performing concurrent multiple line inversions, further reducing the computation time per time step.

2.6 Flux Vector Splitting

Finite difference schemes that are based on centered spatial difference operators are simultaneously stable for both the positive and the negative characteristic speeds associated with the convective flux vectors. The Beam and Warming algorithm is a good example. It is constructed from Eq. (2.47) by using central differences to approximate all of the spatial derivatives. Although the scheme is unstable in three dimensions, the instability is a weak one and can be controlled by numerical dissipation that is added to damp the growth of high-frequency waves and nonlinear instabilities. On the other hand, one-sided difference operators lead to schemes that are stable only for equations with single-signed eigenvalues. However, these schemes have better dissipative and dispersive properties. One-sided difference operators also lead to a lower-banded matrix than the block-tridiagonal matrix that is usually formed with central differencing, and therefore lead to an easier inversion. The gasdynamic equations have characteristic speeds (eigenvalues) of mixed signs in subsonic flow regimes; therefore the use of one-sided spatial-difference operators (upwind schemes) requires splitting the flux terms.

Upwinding requires that the flux vector, (for example $\hat{E}$ in Eq. (2.47), and its Jacobian matrix (in this case $\hat{A}$) be split into sub-vectors and sub-matrices associated with its positive and negative eigenvalues. This can be done by realizing that the Jacobian matrices $\hat{A}$, $\hat{B}$, and $\hat{C}$ have a complete set of eigenvalues and eigenvectors and therefore can be written as

$$\hat{A} = T_\xi \hat{\Lambda}_\xi T_\xi^{-1}, \quad \hat{B} = T_\eta \hat{\Lambda}_\eta T_\eta^{-1}, \quad \hat{C} = T_\zeta \hat{\Lambda}_\zeta T_\zeta^{-1} \tag{2.51}$$

The eigenvalues $\hat{\Lambda}_\xi$, $\hat{\Lambda}_\eta$, and $\hat{\Lambda}_\zeta$ and the eigenvectors T_ξ, T_η, and T_ζ are given in Appendix B.

Using the result of Eq. (2.51) and the fact that $\hat{E}$ is a homogeneous function of degree one in $\hat{Q}$, Steger and Warming [8] rewrote $\hat{E}$ as

$$\hat{E} = T_\xi \hat{\Lambda}_\xi T_\xi^{-1} \hat{Q} \tag{2.52}$$

where the diagonal elements of the matrix $\hat{\Lambda}_\xi$ are given by Eq. (B.2). Any eigenvalue λ_l, $l = 1 \ldots 5$ can be rewritten as

$$\lambda_l = \lambda_l^+ + \lambda_l^- \tag{2.53}$$

where

$$\lambda_l^+ = \frac{\lambda_l + |\lambda_l|}{2}, \qquad \lambda_l^- = \frac{\lambda_l - |\lambda_l|}{2} \tag{2.54}$$

Using this formula one can rewrite the diagonal matrix $\hat{\Lambda}_\xi$

$$\hat{\Lambda}_\xi = \hat{\Lambda}_\xi^+ + \hat{\Lambda}_\xi^- \tag{2.55}$$

where $\hat{\Lambda}_\xi^+$ has the diagonal elements λ_l^+ and $\hat{\Lambda}_\xi^-$ has the diagonal elements λ_l^-. Thus Eq. (2.52) can be rewritten as

$$\begin{aligned} \hat{E} &= T_\xi \left(\hat{\Lambda}_\xi^+ + \hat{\Lambda}_\xi^- \right) T_\xi^{-1} \hat{Q} \\ \hat{E} &= \left(\hat{A}^+ + \hat{A}^- \right) \hat{Q} \\ \hat{E} &= \left(\hat{E}^+ + \hat{E}^- \right) \end{aligned} \tag{2.56}$$

with

$$\hat{A} = \hat{A}^+ + \hat{A}^-, \quad \hat{A}^+ = T_\xi \hat{\Lambda}_\xi^+ T_\xi^{-1}, \quad \hat{A}^- = T_\xi \hat{\Lambda}_\xi^- T_\xi^{-1}$$
$$\hat{E}^+ = \hat{A}^+ \hat{Q}, \qquad \hat{E}^- = \hat{A}^- \hat{Q}$$

2.6.1 Steger Warming Flux Splitting

Using the flux vector splitting as described above, the following three-factored algorithm has been devised by Steger and Warming. [8]

$$\left(I + h\delta_\xi^b \hat{A}^+ + h\delta_\xi^f \hat{A}^- - hRe^{-1}\bar{\delta}_\xi \hat{A}_v^\xi\right)^n$$
$$\left(I + h\delta_\eta^b \hat{B}^+ + h\delta_\eta^f \hat{B}^- - hRe^{-1}\bar{\delta}_\eta \hat{B}_v^\eta\right)^n \tag{2.57}$$
$$\left(I + h\delta_\zeta^b \hat{C}^+ + h\delta_\zeta^f \hat{C}^- - hRe^{-1}\bar{\delta}_\zeta \hat{C}_v^\zeta\right)^n \quad \Delta\hat{Q}^n = \hat{R}^n \tag{2.58}$$

where

$$\hat{R}^n = -\Delta t\left[\delta_\xi^b \hat{E}^+ + \delta_\xi^f \hat{E}^- + \delta_\eta^b \hat{F}^+ + \delta_\eta^f \hat{F}^- + \delta_\zeta^b \hat{G}^+ + \delta_\zeta^f \hat{G}^- - Re^{-1}\left(\bar{\delta}_\xi \hat{E}_v + \bar{\delta}_\eta \hat{F}_v + \bar{\delta}_\zeta \hat{G}_v\right)\right]^n \tag{2.59}$$

Here $h = \Delta t$ or $\Delta t/2$ for first- or second-order time accuracy, δ^b is a backward-difference operator and δ^f is a forward-difference operator. This scheme is also approximately factored, based on the same principles that led to the Beam and Warming algorithm, in order to obtain a block-tridiagonal linear system. The procedure for advancing the solution using the Steger Warming algorithm is similar to the procedure used in the Beam and Warming algorithm. Both algorithms have three factors, one for each coordinate direction. This provides the means to devise a method that combines the two types of differencing is different directions. The EZNSS code contains the capability to choose either central or upwind differencing in each coordinate separately.

2.6.2 F3D Algorithm

Steger *et al* [9] proposed another alternative to using central differencing or flux vector splitting in all directions. By splitting and upwind-differencing the convective flux vector in the streamwise direction, while maintaining a central difference operator for

the crossflow fluxes, a two-factored, partially flux-split algorithm is formed

$$\begin{matrix}\left[I + h\delta_\xi^b \hat{A}^+ + h\delta_\zeta \hat{C} - hRe^{-1}\bar{\delta}_\zeta \hat{C}_v^\zeta\right]^n \\ \left[I + h\delta_\xi^f \hat{A}^- + h\delta_\eta \hat{B} - hRe^{-1}\bar{\delta}_\zeta \hat{B}_v^\eta\right]^n \end{matrix} \quad \Delta\hat{Q}^n = \hat{R}^n \tag{2.60}$$

where

$$\hat{R}^n = -\Delta t \left[\delta_\xi^b \hat{E}^+ + \delta_\xi^f \hat{E}^- + \delta_\eta \hat{F} + \delta_\zeta \hat{G} - Re^{-1}\left(\bar{\delta}_\xi \hat{E}_v + \bar{\delta}_\eta \hat{F}_v + \bar{\delta}_\zeta \hat{G}_v\right)\right]^n \tag{2.61}$$

This scheme is also approximately factored in order to obtain a block-tridiagonal linear system. The procedure for advancing the solution using the two-factor method is as follows:

$$\begin{aligned}\left[I + h\delta_\xi^b \hat{A}^+ + h\delta_\zeta \hat{C} - hRe^{-1}\bar{\delta}_\zeta \hat{C}_v^\zeta\right]^n \Delta\hat{Q}^1 &= \hat{R}^n \\ \left[I + h\delta_\xi^f \hat{A}^- + h\delta_\eta \hat{B} - hRe^{-1}\bar{\delta}_\zeta \hat{B}_v^\eta\right]^n \Delta\hat{Q}^n &= \Delta\hat{Q}^1 \\ \hat{Q}^{n+1} &= \hat{Q}^n + \Delta\hat{Q}^n \end{aligned} \tag{2.62}$$

Due to the upwinding, the first step is solved by a forward sweep, from $\xi = \xi_{min}$ to $\xi = \xi_{max}$, followed by a backward sweep, from $\xi = \xi_{max}$ to $\xi = \xi_{min}$ for the second step.

The two-factored algorithm has better stability properties and was found to be unconditionally stable when applied to a linear model wave equation [10]. It also has better dissipative and dispersive properties due to the flux splitting in the streamwise direction. Numerical dissipation terms are still needed in the crossflow directions, however, since central-difference operators are used there to approximate the spatial derivatives.

2.7 Flux Difference Splitting

An exact solution of the Riemann problem is computationally expensive and therefore approximate Riemann solvers are commonly used. Flux difference splitting utilizes

the solution of the approximate Riemann problem to evaluate the fluxes at cell faces. In what follows, the three FDS methods that are used in EZNSS are described in detail.

2.7.1 HLLC

The concept of average-state approximations was introduced by Harten, Lax and van-Leer [11] 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* [12] discussed this limitation, and proposed a modified three wave solver, named HLLC, where the contact is explicitly present. This HLLC schemes is found to have the following properties: 1) exact preservation of isolated contact and shear waves, 2) positivity preserving of scalar quantity, and 3) enforcement of the entropy condition. The resulting scheme greatly improves contact resolution and has been successfully used to compute compressible viscous and turbulent flows [13].

The HLLC Riemann solver as proposed by Batten *et al* [13] is implemented in the EZNSS code. The HLLC scheme assumes two intermediate states, $\mathbf{U}_L^*$ and $\mathbf{U}_R^*$ 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 Riemann solver, respectively). The states $\mathbf{U}_L^*$ and $\mathbf{U}_R^*$ are split by the contact wave, which moves with the velocity S_M (see Figure 2.1).

The wave speeds S_L and S_R are computed according to Einfeldt *et al* [14] as follows:

$$S_L = Min[\lambda_l, \lambda_l^{Roe}] \tag{2.63}$$

$$S_R = Max[\lambda_m, \lambda_m^{Roe}] \tag{2.64}$$

where λ_l is the smallest eigenvalue and λ_m is the largest eigenvalue. Similarly, λ_l^{Roe} and λ_m^{Roe} are the smallest and largest eigenvalues of the Roe matrix [15], respectively. The normal velocity to the interface is denoted by q and is defined as $q = un_x + vn_y + wn_z$.

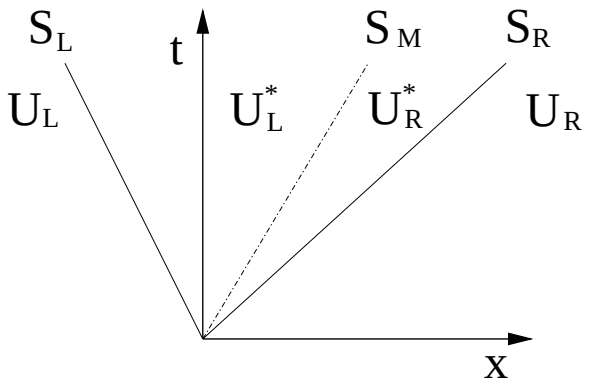

Figure 2.1: Wave structure of the HLLC Riemann solver

The contact wave speed S_M is calculated according to Batten *et al* [13] by

$$S_M = \frac{\rho_r q_r \left(S_R - q_r\right) - \rho_l q_l \left(S_L - q_l\right) + p_l - p_r}{\rho_r \left(S_R - q_r\right) - \rho_l \left(S_L - q_l\right)} \tag{2.65}$$

This choice of S_M enforces the equality of the two star pressures, *i.e.*, $p^* = p_L^* = p_R^*$ which is obtained from

$$p^* = \rho_l \left(q_l - S_L\right)\left(q_l - S_M\right) + P_l = \rho_R \left(q_R - S_R\right)\left(q_R - S_M\right) + P_R \tag{2.66}$$

Introducing the intermediate left state vector

$$\mathbf{U}_l^* = \begin{Bmatrix} \rho_l^* \\ (\rho u)_l^* \\ (\rho v)_l^* \\ (\rho w)_l^* \\ E_l^* \\ (\rho k)^* \\ (\rho \phi)^* \end{Bmatrix} = \Omega_l \begin{Bmatrix} \rho_l \left(S_L - q_l\right) \\ \left(S_L - q_l\right)(\rho u)_l + \left(p^* - p_l\right) n_x \\ \left(S_L - q_l\right)(\rho v)_l + \left(p^* - p_l\right) n_y \\ \left(S_L - q_l\right)(\rho w)_l + \left(p^* - p_l\right) n_z \\ \left(S_L - q_l\right) E_l - p_l q_l + p^* S_M \\ \rho k \\ \rho \phi \end{Bmatrix} \tag{2.67}$$

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

$$\mathbf{F}_l^* \equiv \mathbf{F}\left(\mathbf{U}_l^*\right) = \mathbf{U}_l^* S_M + \begin{Bmatrix} 0 \\ p^* n_x \\ p^* n_y \\ p^* n_z \\ p^* S_M \\ 0 \\ 0 \end{Bmatrix} \tag{2.68}$$

and the corresponding intermediate right state vector and flux vector are obtained from Eqs. (2.67), (2.68) by simultaneously interchanging the subscripts $l \rightarrow r$ and $L \rightarrow R$. Finally, the numerical HLLC flux is defined as follow

$$\mathbf{F}_c\left(\mathbf{U}_l, \mathbf{U}_r\right) = \begin{Bmatrix} \mathbf{F}_c\left(\mathbf{U}_l\right) & if\ S_L > 0 \\ \mathbf{F}_c\left(\mathbf{U}_l^*\right) & if\ S_L \leq 0 < S_M \\ \mathbf{F}_c\left(\mathbf{U}_r^*\right) & if\ S_M \leq 0 \leq S_R \\ \mathbf{F}_c\left(\mathbf{U}_r\right) & if\ S_R < 0 \end{Bmatrix} \tag{2.69}$$

where $\mathbf{F}_c\left(\mathbf{U}_l\right)$ (or $\mathbf{F}_c\left(\mathbf{U}_r\right)$) is the left (right) supersonic flux vector.

2.7.2 AUSM

The advection upstream splitting method (AUSM) was first introduced in the year 1993 by Liou and Steffen [16]. 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.

Although AUSM schemes enjoy a demonstrated improvement in accuracy, effi-

ciency and robustness over existing schemes, the have been found to have deficiencies in some cases. In the year 1996, Liou improved the original AUSM, termed now the $AUSM^+$ [17]. Among the improvement features of the original AUSM scheme are the following properties: (1) exact resolution of a one-dimensional contact wave 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 $AUSM^+ - up$ [18] extended for all speed flows. The $AUSM^+ - up$ is implemented in the EZNSS code and it is given as follows:

$$\mathbf{F}_c\left(\mathbf{U}_l, \mathbf{U}_r\right) = \mathbf{p}_{1/2} + \dot{m}_{1/2} \begin{Bmatrix} \psi_l & if\ \dot{m}_{1/2} > 0 \\ \psi_r & otherwise \end{Bmatrix} \tag{2.70}$$

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

$$\dot{m}_{1/2} = a_{1/2} M_{1/2} \begin{Bmatrix} \rho_l & if\ M_{1/2} > 0 \\ \rho_r & otherwise \end{Bmatrix} \tag{2.71}$$

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

$$p_{1/2} = \mathcal{P}^+_{(5)}(M_l) p_l + \mathcal{P}^-_{(5)}(M_r) p_r - K_u \mathcal{P}^+_{(5)}(M_l) \mathcal{P}^-_{(5)}(M_r)(\rho_l + \rho_r)(f_a a_{1/2})(q_r - q_l) \tag{2.72}$$

with q as the normal velocity to the interface 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:

$$M_{l/r} = \frac{q_{l/r}}{a_{1/2}} \tag{2.73}$$

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

$$\overline{M}^2 = \frac{q_l^2 + q_r^2}{2 a_{1/2}} \tag{2.74}$$

$$M_o^2 = min(1, max(\overline{M}^2, M_\infty^2) \tag{2.75}$$

$$f_a(M_o) = M_o(2 - M_o) \tag{2.76}$$

$$M_{1/2} = \mathcal{M}^+_{(4)}(M_l) + \mathcal{M}^-_{(4)}(M_r) - \frac{K_p}{f_a} max(1 - \sigma\overline{M}^2, 0)\frac{p_r - p_l}{\rho_{1/2} a^2_{1/2}}, \quad \rho_{1/2} = (\rho_l + \rho_r)/2 \tag{2.77}$$

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), given as follows:

$$\mathcal{M}^\pm_1 = \frac{1}{2}(M \pm |M|) \tag{2.78}$$

$$\mathcal{M}^\pm_2 = \pm\frac{1}{4}(M \pm 1)^2 \tag{2.79}$$

$$\mathcal{M}^\pm_{(4)}(M) = \begin{Bmatrix} \mathcal{M}^\pm_{(1)} & if\ |M| > 0 \\ \mathcal{M}^\pm_{(2)}(1 \mp 16\beta\mathcal{M}^\mp_{(2)}) & otherwise \end{Bmatrix} \tag{2.80}$$

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

$$\mathcal{P}^\pm_{(5)}(M) = \begin{Bmatrix} \frac{1}{M}\mathcal{M}^\pm_{(1)} & if\ |M| \geq 1 \\ \mathcal{M}^\pm_{(2)}[(\pm 2 - M) \mp 16\alpha M\mathcal{M}^\mp_{(2)}) & otherwise \end{Bmatrix} \tag{2.81}$$

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

2.7.3 MAPS

The Mach number-based advection pressure splitting scheme (MAPS) was developed by Cord-Christian Rossow [19]. The MAPS scheme employs elements of the LDFSS [20] and of the CUSP [21] formulations and uses the left and right Mach number at an interface to establish the flux function. Using the left and right Mach numbers, and almost completely avoiding the need of an intermediate state, leads to a very simple scheme, which despite its simplicity rivals the common, most advanced high-resolution/high -accuracy schemes such as the AUSM and LDFSS schemes.

Similar to the AUSM (and the LDFSS) scheme the MAPS scheme splits the convective flux-density vector into and advective contribution and into a contribution associated with the pressure. The convective flux vector then may be given as

$$\mathbf{F} = \mathbf{F}_{ad} + \mathbf{F}_p \tag{2.82}$$

where the the advective flux, $\mathbf{F}_{ad}$, is given by

$$\mathbf{F}_{ad} = \frac{1}{4}(q_l+q_r)(\phi_l+\phi_r) - \frac{1}{4}(\phi_l+\phi_r)\beta^M c_{av}[M_r sign(M_r) - M_l sign(M_l)] - \frac{1}{2}c_{av} max(|M_l|, |M_r|)(\phi_r - \phi_l) \tag{2.83}$$

where ϕ is the vector of advected quantities defined as

$$\phi = [\rho, \rho u, \rho v, \rho w, \rho H]^T \tag{2.84}$$

and ϕ_n is the normal ϕ at an interface. The Mach number of the interface normal velocity, denoted by M, is evaluated as

$$M = \frac{q}{c_{av}} \tag{2.85}$$

and c_{av} is the averaged speed of sound at the interface evaluated by

$$c_{av} = \frac{1}{2}(c_l + c_r) \tag{2.86}$$

The function β^M is given by

$$\beta^M = max(0, 2M_p - 1) \tag{2.87}$$

$$M_p = min[max(|M_l|, |M_r|), 1] \tag{2.88}$$

The contribution of the pressure at an interface is determined via

$$\mathbf{F}_p = \frac{1}{2}(\mathbf{p}_l + \mathbf{p}_r) - \frac{1}{2}\beta^p[\mathbf{p}_r sign(M_r) - \mathbf{p}_l sign(M_l)] \tag{2.89}$$

where

$$\mathbf{p} = [0, p \cdot n_x, p \cdot n_y, p \cdot n_z, 0] \tag{2.90}$$

and the blending function β^p is defined as

$$\beta^p = max(0, 2M_m - 1) \tag{2.91}$$

$$M_m = min[min(|M_l|, |M_r|), 1] \tag{2.92}$$

The MAPS scheme can be easily extended to all speed flows and more details can be found in Ref [22]. The extended MAPS for all speed flow is the one that is actually implemented in the EZNSS code.

2.8 Dual Time Step Formulation

To further increase the time accuracy and to decrease the time step, a dual-time step procedure may be added. Following the addition of the dual time-step term and using backward differences in time, a general second-order scheme (in time and space) takes the following form:

$$\frac{\hat{Q}^{k+1} - \hat{Q}^k}{\Delta\tau} + \frac{3\hat{Q}^{k+1} - 4\hat{Q}^n + \hat{Q}^{n-1}}{2\Delta t} + \frac{\partial \hat{E}^{k+1}}{\partial \xi} + \frac{\partial \hat{F}^{k+1}}{\partial \eta} + \frac{\partial \hat{G}^{k+1}}{\partial \zeta} = \frac{1}{Re}\left(\frac{\partial \hat{E}_v^{k+1}}{\partial \xi} + \frac{\partial \hat{F}_v^{k+1}}{\partial \eta} + \frac{\partial \hat{G}_v^{k+1}}{\partial \zeta}\right) \tag{2.93}$$

The super-script n denotes the time step while the super-script k denotes the sub-iteration step that arises from the dual time-step formulation. The time step is denoted by Δt and the sub-iteration time-step is denoted by $\Delta\tau$. The sub-iteration time-step is set based on the CFL number and on the convergence rate of the sub-iterations.

Following the linearization in time of the inviscid and viscous fluxes, Eq. (2.93)

takes the form:

$$\left\{ I + h\left[\frac{\partial}{\partial\xi}\hat{A} + \frac{\partial}{\partial\eta}\hat{B} + \frac{\partial}{\partial\zeta}\hat{C} - \frac{1}{Re}\left(\frac{\partial}{\partial\xi}\hat{A}_v^{\xi} + \frac{\partial}{\partial\eta}\hat{B}_v^{\eta} + \frac{\partial}{\partial\zeta}\hat{C}_v^{\zeta}\right)\right]^k \right\} \Delta\hat{Q}^k =$$
$$-h\left[\frac{3\hat{Q}^k - 4\hat{Q}^n + \hat{Q}^{n-1}}{2\Delta t} + \frac{\partial\hat{E}^k}{\partial\xi} + \frac{\partial\hat{F}^k}{\partial\eta} + \frac{\partial\hat{G}^k}{\partial\zeta} - \frac{1}{Re}\left(\frac{\partial\hat{E}_v^k}{\partial\xi} + \frac{\partial\hat{F}_v^k}{\partial\eta} + \frac{\partial\hat{G}_v^k}{\partial\zeta} + \right)\right]$$
$$+O\left(\Delta t^2\right) \quad (2.94)$$

where I is the identity matrix, $h = \frac{2\Delta\tau\Delta t}{3\Delta\tau+2\Delta t}$, $\Delta\hat{Q}^k = \hat{Q}^{k+1} - \hat{Q}^k$, and $\partial/\partial\xi$, $\partial/\partial\eta$, and $\partial/\partial\zeta$ are approximated by finite differencing. The matrices A, B, and C, are the inviscid Jacobian matrices while the matrices $\hat{A}_v^{\xi}$, $\hat{B}_v^{\eta}$, and $\hat{C}_v^{\zeta}$ are approximated viscous Jacobian matrices that contain derivatives only in the direction denoted by the super-script. The approximation in evaluating the viscous Jacobian matrices is required to guarantee the block-tridiagonal pattern of the factored schemes. Nevertheless, thanks to the dual-time formulation, the time accuracy is not hampered by using the approximations. The specific numerical scheme is set by the fashion in which the spatial partial derivatives are approximated. When using the Beam and Warming scheme all derivatives are evaluated by central differences. In this cases, additional explicit and implicit smoothing terms are added to damp high frequency oscillations. When using the Steger-Warming flux-vector splitting, up-winding is employed for all partial derivatives.

The sub-iteration is considered converged if the residual that is based on $\Delta\hat{Q}^k$ has dropped three orders in magnitude. This usually happens within 6-8 sub-iterations. At that point, $\Delta\hat{Q}^k \to 0$ and therefore $\hat{Q}^{k+1} \to \hat{Q}^k \to \hat{Q}^{n+1}$. The right hand side of Eq. (2.94) also approaches zero and can be rewritten as (dropping the $O\left(\Delta t^2\right)$ term):

$$\frac{3\hat{Q}^{n+1} - 4\hat{Q}^n + \hat{Q}^{n-1}}{2\Delta t} + \frac{\partial\hat{E}^{n+1}}{\partial\xi} + \frac{\partial\hat{F}^{n+1}}{\partial\eta} + \frac{\partial\hat{G}^{n+1}}{\partial\zeta} =$$
$$\frac{1}{Re}\left(\frac{\partial\hat{E}_v^{n+1}}{\partial\xi} + \frac{\partial\hat{F}_v^{n+1}}{\partial\eta} + \frac{\partial\hat{G}_v^{n+1}}{\partial\zeta}\right) \quad (2.95)$$

Having evaluated the partial derivatives using finite differences, Eq. (2.95) is the

second-order finite difference form of Eq. (2.41). This equation is satisfied at every grid point for each time step.

The importance of the dual time formulation is increased when conducting store separation simulations. Since the geometry changes require that the suite of Chimera routines for hole cutting and interpolation point search is invoked for each time step, using a regular time-accurate simulation, with a much smaller time step, becomes inefficient. The dual time step formulation allows the use of much larger time steps and therefore the overhead of the Chimera routines and other routines, such as metric coefficients calculations, is greatly reduced. This is true, even when a rather large number of sub-iterations is required for each time step to converge.

2.9 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 easy 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 flow 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 EZNSS code currently provides the choice between zero equations, *i.e.* an algebraic model (two variants), two one-equation turbulence models, two two-equation models, and three Reynolds stress models (RSM are still in their beta phase). It also provide the means to conduct detached eddy

simulations (DES) using hybrid models.

2.9.1 Baldwin-Lomax Turbulence Model

The Baldwin-Lomax algebraic turbulent model was developed originally by Baldwin and Lomax [7] for a flat-plate boundary layer, based on the two-layer model reported by Cebeci *et al* [23], and was later modified by Degani and Schiff [24] to be applicable to flow fields containing crossflow separation. The modified model has been used successfully to simulate subsonic as well as supersonic flows [24–26].

On the basis of the algebraic model, the coefficients of viscosity μ and thermal conductivity κ are replaced by the relations

$$\begin{aligned} \mu &= \mu_l + \mu_t \\ \kappa &= \kappa_l + \frac{c_p \mu_t}{Pr_t} \end{aligned} \tag{2.96}$$

The turbulent eddy viscosity coefficient μ_t is computed using the Baldwin-Lomax algebraic eddy-viscosity model [7] as follows,

$$\mu_t = \begin{cases} (\mu_t)_{inner}, & y \leq y_c \\ (\mu_t)_{outer}, & y > y_c \end{cases} \tag{2.97}$$

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

$$(\mu_t)_{inner} = \rho l^2 |\omega| \tag{2.98}$$

where l, the length scale, is

$$l = ky \left[1 - e^{-\left(y^+/A^+\right)}\right] \tag{2.99}$$

The quantity $|\omega|$ is the magnitude of the local vorticity vector, A^+ is a constant, and

y^+ is the normal law-of-the-wall coordinate defined by

$$y^+ = \frac{\sqrt{\rho_w \tau_w}}{\mu_w} y \tag{2.100}$$

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

$$(\mu_t)_{outer} = K C_{cp} \rho F_{wake} F_{kleb}(y) \tag{2.101}$$

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

$$F_{wake} = y_{max} F_{max} \tag{2.102}$$

In Eq. (2.102), F_{max} is the maximum value of $F(y)$

$$F(y) = |\omega| y \left[1 - e^{-\left(y^+/A^+\right)}\right] \tag{2.103}$$

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

$$F_{kleb}(y) = \left[1 + 5.5 \left(\frac{C_{kleb} y}{y_{max}}\right)^6\right]^{-1} \tag{2.104}$$

The constants appearing in Eqs. (2.96-2.104) were determined by Baldwin and Lomax [7]

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

2.9.2 Degani-Schiff Modification

The turbulence model described up to this point was developed for two-dimensional boundary-layer flows. Degani and Schiff extended it to simulate three-dimensional flows having crossflow separation on the basis of a similar behavior of turbulent boundary layers developed near surfaces of slender bodies of revolution at high angle of attack [24]. A close examination of a typical flow structure about an inclined body of revolution would reveal why the extension of the model to three-dimensional flows can be justified with the modifications proposed by Degani and Schiff.

The major difficulty that Degani and Schiff encountered in applying the Baldwin-Lomax model to flows with crossflow separation was the evaluation of the length scale y_{max} for the separated regions. The quantity y_{max} is found by searching for the maximum value of the function $F(y)$ [Eq. (2.103)] along rays perpendicular to the body. On the windward side, where the boundary layer is still attached, the function $F(y)$ contains only one distinct maximum. On the leeward side of the body the profile of the function $F(y)$ may have more than one maximum. The attached boundary layer creates a peak similar to the one created by the attached boundary layer on the windward side. In addition, the vortical structure causes the function $F(y)$ to have more peaks due to the local maxima of $|\omega|$. The peaks associated with the vortical structure are greater than the one associated with the attached boundary layer and, therefore, one of these will be picked as the reference for the evaluation of the length scale, y_{max}, and, in turn, F_{max} and $(\mu_t)_{outer}$.

The modifications proposed by Degani and Schiff were directed at choosing the "right" peak of the profiles of $F(y)$. The procedure that was adopted included a command that if the value of $F(y)$ dropped to 90% of the maximum value found along a certain ray, the search would be stopped and that local maximum would be picked for calculating the reference length and F_{max} for that circumferential angle. This procedure ensures that the first peak, the one associated with the boundary layer, will be chosen, but does not solve the problem at all circumferential angles. At a region of secondary separation or one close to a primary separation line, the boundary layer does not produce a clear maximum because of a merger of the maximum associated with the boundary layer and the maximum associated with the overlying vortex.

So the criterion chosen to detect the maximum associated with the boundary layer fails. In order to resolve this problem, another parameter was introduced: the cutoff distance. With the exception of the ray on the windward plane of symmetry, where the flow is attached and a simple search is sufficient to detect the lone maximum, the cutoff distance was chosen to be

$$y_{cutoff} = cy_{max}\,(\phi = 0) \tag{2.105}$$

The search from that point on is done only up to the cutoff distance. If a maximum is not encountered prior to reaching the cutoff distance, the search is stopped and F_{max} and y_{max} are taken to be those found on the previous ray. The cutoff distance parameter, the constant c in Eq. (2.105), is chosen empirically, based on a great number of numerical experiments.

The boundary conditions applied in the calculations are chosen to simulate viscous flows. The wall boundary conditions, applied at $\zeta = 1$, enforce zero slip and adiabatic wall conditions. The contravariant velocities U, V, W are all set to zero, and a zeroth-order normal pressure gradient condition is applied. The inflow boundary condition, applied at $\zeta = \zeta_{max}$, enforces free-stream conditions, while the exit boundary condition, applied at $\xi = \xi_{max}$, is a simple zeroth-order zero-axial-gradient extrapolation condition. A periodic boundary condition is applied at $\eta = \eta_{max}$, allowing the flow to become asymmetric if warranted by the flow physics. Application of the periodic boundary condition results in a system of periodic equations for the circumferential factor. Computationally, the solution of the periodic set is very expensive but it is necessary in order to capture asymmetric flow solutions as well as symmetric solutions.

2.9.3 Goldberg One-Equation Model

This one-equation turbulence model was developed by Goldberg. [27, 28] The model, called the R_t model, consists of a transport equation for the undamped eddy viscosity (denoted by R). The equation contains a convection term, a diffusion term, a

production term, and a destruction term. The equation has the following form:

$$\rho\frac{DR}{Dt} = \frac{\partial}{\partial x_i}\left[\left(\mu+\frac{\mu_t}{\sigma_R}\right)\frac{\partial R}{\partial x_j}\right] + C_1\rho\left(RP_k\right)^{0.5} - \left(C_3 f_3 - C_2\right)\rho D \quad (2.106)$$

The production term, denoted by P_k, has the form

$$P_k = \nu_t\left[\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)\frac{\partial U_i}{\partial x_j} - \frac{2}{3}\left(\frac{\partial U_k}{\partial x_k}\right)^2\right] \quad (2.107)$$

where ν_t is the kinematic eddy viscosity $\frac{\mu_t}{\rho}$. The destruction term, denoted by D, has the form

$$D = \begin{cases} \frac{\partial R}{\partial x_j}\frac{\partial R}{\partial x_j}, & \frac{\partial Q}{\partial x_j}\frac{\partial R}{\partial x_j} > 0 \\ 0, & otherwise \end{cases} \quad (2.108)$$

where Q is the velocity magnitude, $Q = (U^2+V^2+W^2)^{\frac{1}{2}}$. The equation is subjected to the following boundary conditions: $R = 0$ at solid walls and $R_\infty \leq \nu_\infty$ at free stream (and initial conditions).

The eddie viscosity field is given by:

$$\mu_t = f_\mu \rho R \quad (2.109)$$

where

$$f_\mu = \frac{\tanh\left(\alpha\chi^2\right)}{\tanh\left(\beta\chi^2\right)} \quad (2.110)$$

and

$$\chi = \frac{\rho R}{\mu} \quad (2.111)$$

The model constants and damping function f_3 are derived from asymptotic arguments at walls and limit flow regions such as pipe centerline and the logarithmic overlap of flat plate, pipe, and channel flows. The form of the closure parameters are given as

follows:

$$\begin{aligned} f_3 &= 1 + \frac{2\alpha}{3\beta C_3 \chi} \\ C_1 &= \chi^2 \left(C_3 - C_2 - \frac{1}{\sigma_R} \right) \\ C_2 &= -\frac{5\alpha}{3\beta\sigma_R} \\ C_3 &= C_2 + \frac{3}{2\sigma_R} \end{aligned} \tag{2.112}$$

where

$$\begin{aligned} \sigma_R &= 0.8 \\ \alpha &= 0.07 \\ \beta &= 0.2 \end{aligned} \tag{2.113}$$

Three important features characterize the R_t model. First, the model is topology free since it does not involve wall distance and therefore, the model can be used to calculate the eddy viscosity in boundary layers as well as in shear layers. This allows to model the turbulent flow about an airfoil and the turbulent wake behind it using the same model. Second, The model is capable of modeling the laminar sub-layer as the well as the turbulent boundary layer. And third, the evolution of the undamped eddy viscosity is such that transition from laminar to turbulent flow occurs without any additional triggers. These three important features provide an advantage compared to commonly used models that usually require a priori setting of various parameters. For example, the transition point for the Spalart-Allmaras model or the laminar sub-layer thickness for the cubic $\kappa - \epsilon$ model. Other models use various methods to decide whether transition to turbulence has occurred.

The fact that the transition occurs naturally provides the means of calculating highly complicated turbulent flows, such as flows that include transition to turbulence. Moreover, it is possible to simulate flows that include laminar separation that is followed by a reattachment of the flow and after a transition to turbulence, a turbulent

separation. Such a flow is typical to the high angle of attack flow about airfoils. The fact that the model is topology free allows to study the wake and, in turn, the effects of the wake on the airfoil. Moreover, it provides the means for simulating the flows about multi-element airfoils.

The model is implemented in a segregated manner, that is the flow is advanced one time step and then the undamped eddy viscosity is advanced using an implicit formulation of the transport equation. Great care has to be taken in generating an appropriate mesh. Note, even with a carefully generated computational mesh, it is extremely difficult to obtain a stable solution of the flow about airfoils at high angles of attack. In contrast, the model has been successfully used to simulated high angle of attack flows about slender bodies of revolution.

2.9.4 Spalart-Allmaras Turbulence Model

The curvilinear coordinate system formulation of the SA model is given as follows:

$$\frac{\partial \hat{q}}{\partial \tau} + \frac{\partial \hat{f}}{\partial \xi} + \frac{\partial \hat{g}}{\partial \eta} + \frac{\partial \hat{h}}{\partial \zeta} = \frac{\partial \hat{f}_v}{\partial \xi} + \frac{\partial \hat{g}_v}{\partial \eta} + \frac{\partial \hat{h}_v}{\partial \zeta} + \frac{S_{\tilde{\nu}}}{J} \tag{2.114}$$

where $\hat{q} = \rho\tilde{\nu}/J$ is the solution vector. The fluid density is denoted by ρ while $\tilde{\nu}$ is the undamped eddy viscosity. The terms $\hat{f}$, $\hat{g}$, and $\hat{h}$ represent the rotated inviscid fluxes:

$$\hat{f} = \frac{1}{J}\rho\tilde{\nu}U, \quad \hat{g} = \frac{1}{J}\rho\tilde{\nu}V, \quad \hat{h} = \frac{1}{J}\rho\tilde{\nu}W \tag{2.115}$$

where U, V, and W are the contravariant velocities. The terms $\hat{f}_v$, $\hat{g}_v$, and $\hat{h}_v$ represent the rotated viscous fluxes. For the sake of convenience, the viscous fluxes are split into diffusion (denoted by the d superscript) and anti-diffusion (denoted by the ad superscript) and are defined as

$$\hat{f}_v = \left(\hat{f}_v\right)^d + \left(\hat{f}_v\right)^{ad}, \quad \hat{g}_v = \left(\hat{g}_v\right)^d + \left(\hat{f}_v\right)^{ad}, \quad \hat{h}_v = \left(\hat{h}_v\right)^d + \left(\hat{h}_v\right)^{ad} \tag{2.116}$$

where

$$\left(\hat{f}_v\right)^d = \frac{1+C_{b2}}{\sigma J}\frac{\partial\left(\mu_{\tilde{\nu}}\widetilde{\nabla}\tilde{\nu}\cdot\frac{\partial\mathbf{\Theta}}{\partial x}\right)}{\partial\xi}, \quad \left(\hat{f}_v\right)^{ad} = -C_{b2}\frac{\rho\tilde{\nu}}{\sigma J}\frac{\partial\left(\widetilde{\nabla}\tilde{\nu}\cdot\frac{\partial\mathbf{\Theta}}{\partial x}\right)}{\partial\xi} \tag{2.117}$$

$$\left(\hat{g}_v\right)^d = \frac{1+C_{b2}}{\sigma J}\frac{\partial\left(\mu_{\tilde{\nu}}\widetilde{\nabla}\tilde{\nu}\cdot\frac{\partial\mathbf{\Theta}}{\partial y}\right)}{\partial\eta}, \quad \left(\hat{g}_v\right)^{ad} = -C_{b2}\frac{\rho\tilde{\nu}}{\sigma J}\frac{\partial\left(\widetilde{\nabla}\tilde{\nu}\cdot\frac{\partial\mathbf{\Theta}}{\partial y}\right)}{\partial\eta} \tag{2.118}$$

$$\left(\hat{h}_v\right)^d = \frac{1+C_{b2}}{\sigma J}\frac{\partial\left(\mu_{\tilde{\nu}}\widetilde{\nabla}\tilde{\nu}\cdot\frac{\partial\mathbf{\Theta}}{\partial z}\right)}{\partial\zeta}, \quad \left(\hat{h}_v\right)^{ad} = -C_{b2}\frac{\rho\tilde{\nu}}{\sigma J}\frac{\partial\left(\widetilde{\nabla}\tilde{\nu}\cdot\frac{\partial\mathbf{\Theta}}{\partial z}\right)}{\partial\zeta} \tag{2.119}$$

and $\mathbf{\Theta} = [\xi, \eta, \zeta]$, $\widetilde{\nabla} = [\frac{\partial}{\partial\xi}, \frac{\partial}{\partial\eta}, \frac{\partial}{\partial\zeta}]$, $\mu_{\tilde{\nu}} = (\mu + \rho\tilde{\nu})$. The source term is denoted as $S_{\tilde{\nu}}$ and is given by:

$$S_{\tilde{\nu}} = C_{b1}\tilde{S}\rho\tilde{\nu} - \frac{1}{\rho}C_{\omega1}f_\omega\left(\frac{\rho\tilde{\nu}}{y_n}\right)^2 \tag{2.120}$$

where y_n denotes the wall distance. The remaining constants, σ, C_{b1}, $C_{\omega1}$, C_{b2} and the functions $\tilde{S}$, f_ω may be found in the original publications [29, 30]

2.9.5 Unified Hybrid RANS/LES $k-\omega$-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. Three hybrid RANS/LES turbulence models are utilized in the current work, all of them use the TNT model as the base model. The first hybrid model is termed the X-LES model. Developed by Kok *et al* [31], it consists of a composite formulation incorporating both RANS and LES equations; it uses a clearly defined sub grid scale (SGS) model in the LES mode. Namely, in the LES mode, the two-equation turbulence model degenerates into one equation of the turbulence kinetic energy of the SGS model. The second hybrid model is the delayed DES model (DDES) originally developed by Spalart *et al* [32]. The DDES preserves the RANS mode in the boundary layer region, even when the grid cell scale is lower than that of the boundary layer thickness, thus avoiding a too early switch of the model from the RANS to the LES mode. The third model that

is proposed in the current work is the X-DDES. This model adopted the approach of the X-LES model, namely that the model degenerates into one equation of the turbulence kinetic energy of the SGS model. A general, unified formulation of the hybrid turbulence models is given by:

$$\frac{\partial \hat{q}}{\partial \tau}+\frac{\partial \hat{f}}{\partial \xi}+\frac{\partial \hat{g}}{\partial \eta}+\frac{\partial \hat{h}}{\partial \zeta}=\frac{\partial \hat{f}_v}{\partial \xi}+\frac{\partial \hat{g}_v}{\partial \eta}+\frac{\partial \hat{h}_v}{\partial \zeta}+\frac{S}{J} \tag{2.121}$$

where $\hat{q}=\left[\rho k,\rho\omega\right]^T/J$ is the solution vector. The fluid density is denoted by ρ while k is the turbulence kinetic energy and ω denotes the specific turbulence dissipation rate. The terms $\hat{f}$, $\hat{g}$, and $\hat{h}$ represent the inviscid rotated fluxes:

$$\hat{f}=\frac{1}{J}\left[\rho kU,\rho\omega U\right]^T \quad \hat{g}=\frac{1}{J}\left[\rho kV,\rho\omega V\right]^T \quad \hat{h}=\frac{1}{J}\left[\rho kW,\rho\omega W\right]^T \tag{2.122}$$

where U, V, and W are the contravariant velocities. The terms $\hat{f}_v$, $\hat{g}_v$, and $\hat{h}_v$ represent the rotated viscous fluxes:

$$\left(\hat{f}_v\right)=\frac{1}{J}\left[\frac{\partial\left(\mu_k\widetilde{\nabla}k\cdot\frac{\partial\mathbf{\Theta}}{\partial x}\right)}{\partial\xi},\frac{\partial\left(\mu_\omega\widetilde{\nabla}\omega\cdot\frac{\partial\mathbf{\Theta}}{\partial x}\right)}{\partial\xi}\right]^T \tag{2.123}$$

$$\left(\hat{g}_v\right)=\frac{1}{J}\left[\frac{\partial\left(\mu_k\widetilde{\nabla}k\cdot\frac{\partial\mathbf{\Theta}}{\partial y}\right)}{\partial\eta},\frac{\partial\left(\mu_\omega\widetilde{\nabla}\omega\cdot\frac{\partial\mathbf{\Theta}}{\partial y}\right)}{\partial\eta}\right]^T \tag{2.124}$$

$$\left(\hat{h}_v\right)=\frac{1}{J}\left[\frac{\partial\left(\mu_k\widetilde{\nabla}k\cdot\frac{\partial\mathbf{\Theta}}{\partial z}\right)}{\partial\zeta},\frac{\partial\left(\mu_\omega\widetilde{\nabla}\omega\cdot\frac{\partial\mathbf{\Theta}}{\partial z}\right)}{\partial\zeta}\right]^T \tag{2.125}$$

with $\mathbf{\Theta}=[\xi,\eta,\zeta]$, $\widetilde{\nabla}=[\frac{\partial}{\partial\xi},\frac{\partial}{\partial\eta},\frac{\partial}{\partial\zeta}]$, $\mu_k=(\mu+\mu_t/\sigma_k)$, and $\mu_\omega=(\mu+\mu_t/\sigma_\omega)$. The source vector is given as:

$$S=\begin{Bmatrix}P_k-\rho\frac{k^{1.5}}{l_k}\\ \alpha_\omega\frac{\omega}{k}P_k-\beta_\omega\rho\omega^2+max(\mathcal{E},0)\end{Bmatrix} \tag{2.126}$$

where $\mathcal{E}$ is the cross diffusion term, given as:

$$\mathcal{E} = \sigma_d \frac{\rho}{\omega} \left[\widetilde{\nabla} k \cdot \frac{\partial \mathbf{\Theta}}{\partial x} \widetilde{\nabla} \omega \cdot \frac{\partial \mathbf{\Theta}}{\partial x} + \widetilde{\nabla} k \cdot \frac{\partial \mathbf{\Theta}}{\partial y} \widetilde{\nabla} \omega \cdot \frac{\partial \Theta}{\partial y} + \widetilde{\nabla} k \cdot \frac{\partial \mathbf{\Theta}}{\partial z} \widetilde{\nabla} \omega \cdot \frac{\partial \Theta}{\partial z} \right] \tag{2.127}$$

The production term is denoted by P_k and it is based on the Boussinesq approximation. The turbulent length scale is denoted by l_k, given here by the blended RANS-LES function as follows:

$$l_k = l_{RANS} - f_d \; max \left(l_{RANS} - l_{LES}, 0 \right) \tag{2.128}$$

where f_d is a blending function. The background RANS model length scale, l_{RANS}, is given as

$$l_{RANS} = \frac{k^{1/2}}{\beta_k \omega} \tag{2.129}$$

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

$$l_{LES} = C_{DES} \Delta \tag{2.130}$$

The turbulent viscosity is defined as

$$\mu_t = l_\nu \rho k^{1/2} \tag{2.131}$$

where the turbulence length scale is l_ν (given in Table 2.2). 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 $\kappa = 0.41$. The appropriate length scales that distinguish each model are given in Table 2.2

It should be noted that while the common approach to estimate the grid cell scale as the largest dimension of the grid cell edges, $\Delta = max(\delta_\xi, \Delta_\eta, \Delta_\zeta)$ for the DES calculations, in the present work, the Δ for the X-LES model is calculated according to Abe [33]. Furthermore, the scale Δ that is used for the DDES and X-DDES models is taken from Shur *et al* study [34].

Model type	f_d	l_ν	Δ
RANS	0	$\frac{k^{1/2}}{\omega}$	-
X-LES	1	$min\left(\frac{k^{1/2}}{\omega}, C_1\Delta\right)$	$\sqrt{\frac{\delta_\xi\delta_\eta\delta_\zeta}{min(\delta_\xi,\delta_\eta,\delta_\zeta)}}$
DDES	$f_d = 1 - tanh\left[(8r_d)^3\right]$	$\frac{k^{1/2}}{\omega}$	$min\left[max\left(c_w d, c_w\Delta_m, \Delta_{mn}\right)\Delta_m\right]$
X-DDES	$f_d = 1 - tanh\left[(8r_d)^3\right]$	$min\left(\frac{k^{1/2}}{\omega}, C_1\Delta\right)$	$min\left[max\left(c_w d, c_w\Delta_m, \Delta_{mn}\right)\Delta_m\right]$

$r_d = \frac{\nu+\nu_t}{\left[\kappa^2 d^2 max\left(\sqrt{\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j}}, 10^{-10}\right)\right]}$, $\Delta_m = max\left(\delta_\xi, \delta_\eta, \delta_\zeta\right)$, $c_w = 0.15$, $\kappa = 0.41$,

d=wall distance, Δ_{mn} = grid step size in the wall normal direction

Table 2.2: Turbulence length scales

2.9.6 Reynolds Stress Models

Three Reynolds stress models have been recently introduced into the code. Future versions of the manual shall contain a detailed description of the models. Note that the implementation is still under development. Hence, use with care.

Chapter 3

Computational Mesh Topology

3.1 Introduction

The grid generation process in itself presents one of the main obstacles in performing numerical simulations. Using a structured mesh, as implemented in the EZNSS code, there are two different approaches for mesh generation. The first is the patched grid approach and the second is the over-set grid approach, known as Chimera [1].

In the patched grid approach, the physical domain is divided into zones that touch each other and the computational mesh is generated for each zone separately. Two types of patched grids are supported by the code, the first, a point to point patched grid (with no overlap, but limited flexibility); and the second, with a small overlap between neighboring zones to allow the smooth transfer of information between zones. This approach allows to generate a computational mesh for complex geometries. However, in a case of a geometry change, the computational mesh associated with the component that was changed and the neighboring meshes must be adjusted or regenerated. In the case of minor changes to the geometry, adjustments to the original mesh provide a satisfactory solution, whereas regeneration is required for major geometry changes.

3.2 Chimera

In the Chimera grid approach [1, 35], a separate computational mesh is generated for each component, such as the fuselage or the wing of an aircraft, separately. An outer mesh is generated so that it fully includes the meshes of all the components. The domain decomposition in this approach is simplified and the number of zones that are required is substantially lower. However, the generation of grids surrounding each component is performed independently, and points of a certain mesh may be located within the solid boundaries of the aircraft. Regions located within solid boundaries are called "holes." This is treated by excluding points that are in the holes from the solution process, and using interpolations to update the edges of the holes. The calculations to determine the holes and hole boundaries require a considerable computational effort that, in a moving body simulations, has to be added to the overall computational cost.

3.2.1 Hole Cutting

The Chimera grid topology provides large flexibility in generating a computational mesh about about complex geometries. However, a typical fighter aircraft geometry, especially when it is loaded with stores presents a great challenge for grid generation. Another challenging geometry may that of extremely small geometries where the ratio between grid cells and geometry size is small. Therefore, the Chimera procedure, expecially the hole cutting procedure has been extended to allow a greater flexibility level. The Chimera procedure tolerances have been added to the input file so that they can be appropriately set for various geometry sizes.

3.2.1.1 Multi Zone Bodies

The ability to compose a certain component from several grid zones was introduced to construct a computational mesh for highly complex geometries. This capability requires a different hole cutting methodology and a different scheme for handling the related interpolation boundaries. The hole cutting method is implemented by defining a "box" about the "body" and by examininig each grid point of all the zones that

are not a part of the "body." A point that is found within the "box" is considered a hole.

3.2.1.2 Variable Thickness and Selective Hole Cutting

Two other important features that extend the Chimera scheme felxibility are the variable thickness and selective hole cutting capabilities. These two features are rather simple. They are both composed of a matrix where an entry corresponds to a certain zone with respect to another zone. The entries in the matrices define whether hole cutting is carried on in each case and the thickness matrices define the hole sizes. In addition, the selective hole cutting feature provides the means to increase the Chimera scheme efficiency 6 degrees of freedom motion simulations. It allows to turn off hole cutting (there is a flag for turning off interpolation stencil calculations) for zones where the holes do not change with the motion.

3.2.2 Virtual Body Hole Cutting

It is well known that Chimera grids need to have sufficient overlap. In addition, Chimera grids allow better performance if the regions of data transfer between zones exhibit low gradients. This can be controlled by setting the size of the holes that each geometry component cuts in neighboring components (see previous section). Still, hole edges may still reside too close to the walls of other components. Increasing the hole sizes may result in overlapping holes or in loss of sufficient overlap. Rogers *et al* [36] have proposed an automatic procedure that creates a cutting surface (termed there a "phantom surface") that provides sufficient overlap in cetrain regions and large holes elsewhere. The EZNSS code has a similar feature that is called the "virtual body," a body that is defined for the purpose of hole cutting only. Its primary use is for three-dimensional complex geometries, *e.g.*, the inlet of a fighter aircraft but it can be used for other cases as well (as can be seen from the following example). Figure 3.1 illustrates the use of virtual bodies. The figure contains a description of a Chimera computational mesh about a multi-element airfoil. The virtual bodies, whose boundaries are marked by a black line, allow to generate larger holes in certain

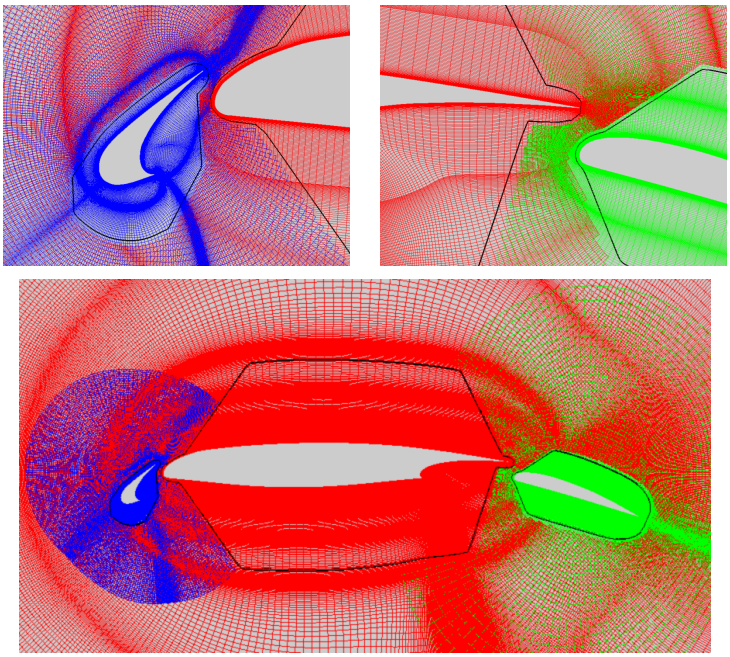

Figure 3.1: Chimera computational mesh (with virtual body hole cutting) about a multi-element airfoil

regions (*e.g.*, boundary layers) without reducing the overlap in other regions (*e.g.*, the gaps).

3.2.3 Fail Safe Mechanism for Interpolations Searches

The interpolation stencil calculation algorithm is avery efficient one as it allows to find a point in a certain grid within a number of step that does not exceed the largest dimension of the three dimensions of the zone. However, in certain cases the search fails. The failure is usually the results of poor grid or high body imperfections (that in certain cases cannot be avoided). To remedy this problem, a cell by cell search is conducted for the small number of cells for which the efficient algorithm fails.

3.3 Patched Grids

Taking advantage of the Chimera grid topology capabilities, patched grid may be implemented such that an arbitrary overlap may exist between neighboring meshes. Therefore, to a certain extent, such patched grids may adequately treat sizeable geometry changes. Further, the special implementation in the EZNSS code provides the means to solve various complex problems is more than one way. When point to point patched grids are used their boundaries can be mixed. That means that some boundaries may be point to point where the other boundaries may have overlap with Chimera grids.

3.4 Single Mesh Topology

To minimize the input parameters that a user must set before running a simulation, the types of separate computational meshes is limited to four main types. The first is a regular type mesh (Cartesian), that is primarily used for describing the walls of a wind tunnel, the second is a C-H grid topology, that is used primarily for wings and pylons, the third is a C-C grid topology that is also used for wings and pylons, and the fourth is a C-O or an O-O grid topology for outer meshes or for slender bodies such as a fuselage. A C-O type mesh, with certain grid connectivity, can be also used for wings.

To further simplify the input process, the code automatically detects the grid topology and assigns appropriate flags that are later used for boundary conditions. These flags may be over-ridden by the user. For C-O or O-O grid topologies, the code identifies whether the mesh is periodic and sets the parameter $JPER$ accordingly ($JPER = 1$ for a periodic element). It also identifies the existence of a singular axis and appropriately sets the IAX parameter ($IAX = 1$ for a forward pointing singular axis, $IAX = 2$ for a backward pointing singular axis, and $IAX = 12$ for two singular axes). For C-H and C-C grid topologies the code identifies the trailing edge and sets $ISTAR$ for the lower and $IEND$ for the upper and both tips (if exist) $JTIPL$ and $JTIPR$. It also sets the $IWING$ flag based on four types. It treats each side

separately to form a 2 digit $IWING$. The types are Open, which is marked by 1, H, which is marked by 2, C, marked by 3, and a special one for hyperbolic grids that is marked by 4. The last type is a wing mesh up to some point and a body mesh from that point on. The C-O type mesh for wings is identified through the $JPER$ flag and the $ICUT$ flag that is also identified automatically. The automatic detection is performed for each zone separately. Another wing topology is the wing that is composed of two zones, one for the upper surface and the other for the lower surface, each having an H topology. This is achieved by setting the $I2ZWING$ flag of one zone to be the negative of the other zone number. The code verifies the connectivity and sets the remaining boundary conditions accordingly.

3.4.1 Examples

This section contains five examples of grid topologies that are supported by the EZNSS code. There are numerous variations of the following examples and these are brought here to illustrate the idea of the automatic topology detection that is supported by the EZNSS code. Note, one must follow the grid generation rules for the automatic detection. This means that the tolerances in the grid generator should be appropriately set. Moreover, it is recommended to verify that certain surfaces have a zero tolerance (when using the Gridgen package, these surfaces are merged into one). In case that the tolerances in the EZNSS code do not allow for the automatic detection, the user may manually enter the appropriate parameters that define the geometry. Care must be taken here because the detection (or lack thereof) by the EZNSS code may point to problems in the grid.

The first example is of a C-O type topology about a slender body (see Figure 3.2). Since the mesh is periodic in the circumferential direction, the periodicity parameter $JPER$ is set to $JPER = 1$. The body has to singular axes and therefore IAX is set to $IAX = 12$. In addition, $IWING$ is set to $IWING = 0$.

The second example is of a C-H topology(see Figure 3.3). The wing in this case extends from the root to the tip (as opposed to a tip to tip wing). The mesh extends in an H shape toward the tips. In this case the $IWING$ parameter is set to $IWING =$

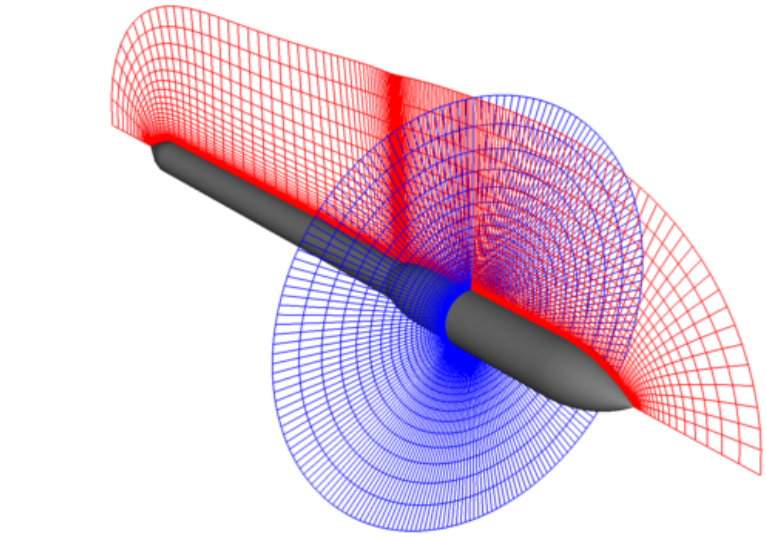

Figure 3.2: C-O grid topology

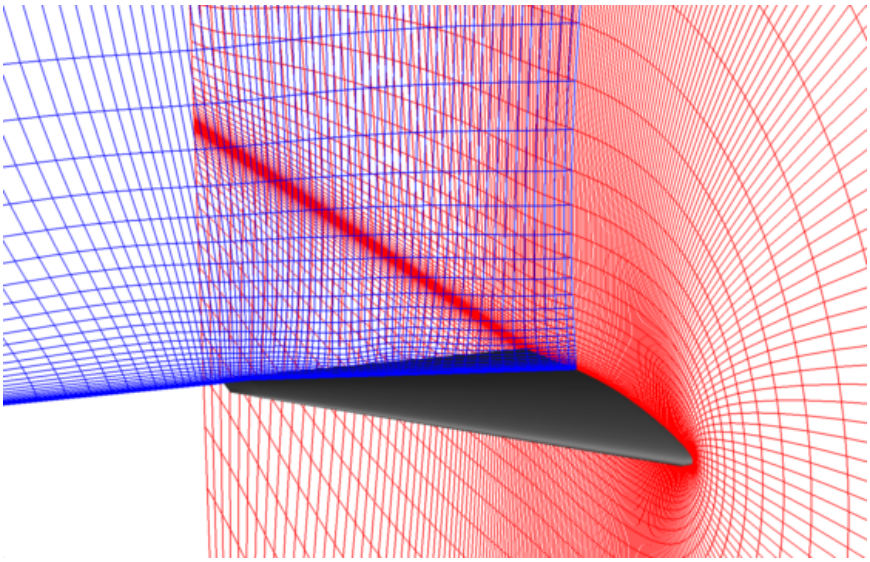

Figure 3.3: C-H grid topology

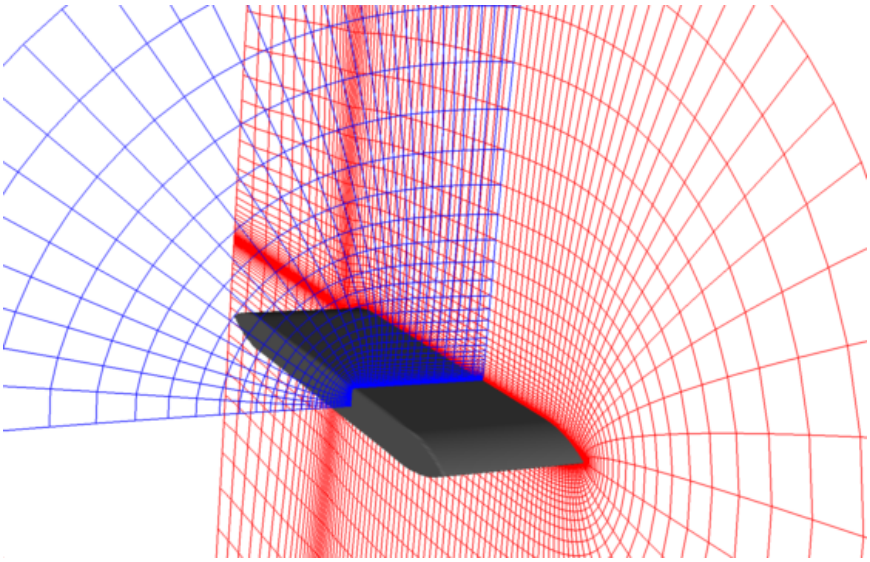

Figure 3.4: C-C grid topology

12. The IAX parameter and the $JPER$ parameter are set to zero for this case.

The third example corresponds to a C-C type grid topology (see Figure 3.4). The $IWING$ parameter is set in this case to $IWING = 13$. In both wing cases above, the digit "1" stand for an open tip, characteristic to wing that emanates from the root. The digit "2" stands for the H type at the tip and the digit "3" stands for the C type at the tip. As in the case of the C-H wing grid above, the parameters IAX and $JPER$ are set to zero.

The fourth and last example corresponds to the C-O type grid topology when its used for wings (see Figure 3.5). The $JPER$ parameter is set in this case to $JPER = 1$ and the $ICUT$ parameter is set to $ICUT = 12$. Depending on the geometry, the $ICUT$ parameter can be also set to "1" or "2."

The fifth and last example is that of the two zone wing (see Figure 3.6). As mentioned above it is set using the variable $I2ZWING$. Once verified, the program automatically sets $ISTART$, $IEND$, $JTIPL$, and $JTIPR$.

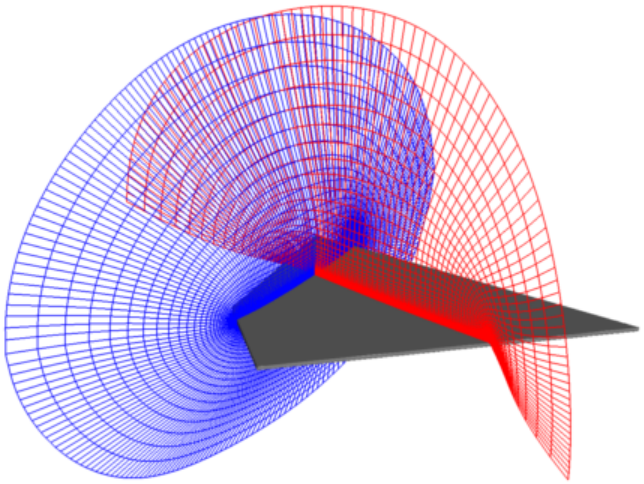

Figure 3.5: C-O grid topology for wings

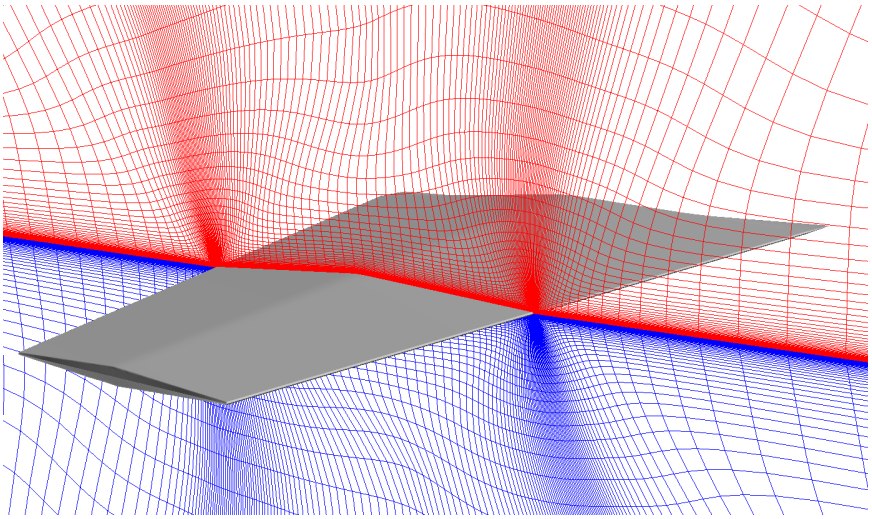

Figure 3.6: Two zone grid topology for wings

3.4.2 Grid Generation

Generation of the grid surrounding each component of the geometry is performed separately. The surface grid is created based on the surface definition using various tools. Grid points are distributed so that regions containing important flow features are sufficiently resolved. These include regions where shocks are expected, regions near intersections between components, and other areas where large flow gradients are expected. Once the surface grid is defined the interior grid points are calculated using hyperbolic or elliptic methods. In the case of a hyperbolic grid generator, the grid is generated based on the surface grid by marching away from the surface, and the outer boundary is formed as a part of the process. In the case of an elliptic grid generator, the outer boundary has to be defined prior to the grid generation process.

3.5 Elliptic Collar Grids

3.5.1 Introduction

The Chimera approach provides an elegant solution to generating meshes surrounding complex geometries. However, difficulties arise around intersecting components, *e.g.*, fuselage and a wing, as they create overlapping holes. Parks *et al* [37] offered a solution by creating "collar grids" to provide appropriate interpolation stencils around the intersection region.

The collar grids are based on the C grid topology. It is a natural choice because this type of collar grids is used to provide the link between the C grid of the wing (or empennage component) and the fuselage grid. Consequently, two of the boundaries of the collar grid are surfaces of the intersecting geometries, one of the fuselage and the other of the wing. The remaining four boundaries are within the computational domains surrounding the fuselage and the wing. In contrast to the hyperbolically generated collar grids, one coordinate originates at the fuselage surface and runs outward along the wing surface. A second coordinate originates at the wing surface and runs along the fuselage surface.

The grid generation process is comprised of three steps. The first is the surfaces

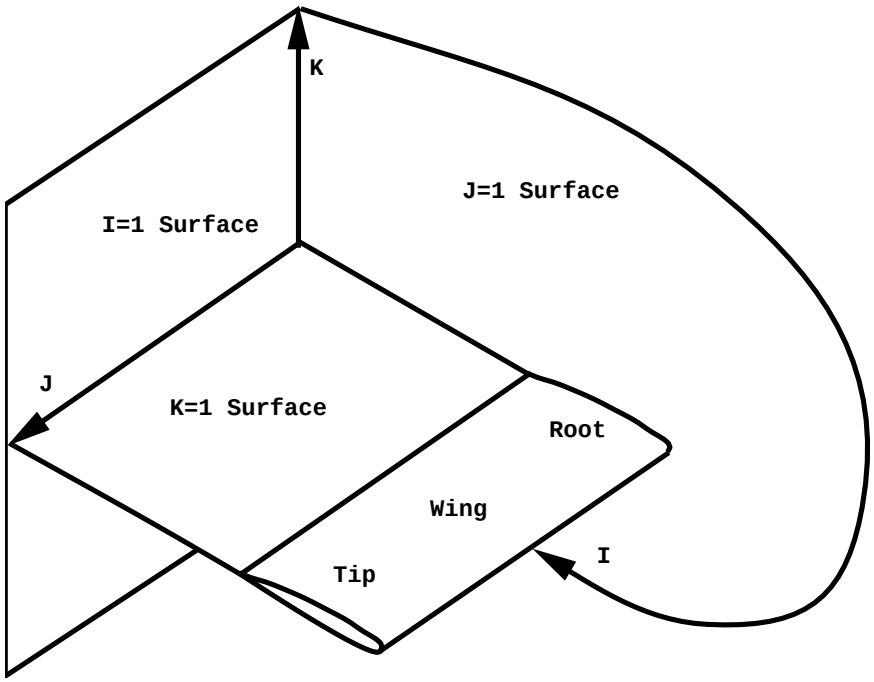

Figure 3.7: C grid topology

and boundaries generation, the second is an algebraic distribution, and the third is smoothing using an elliptic scheme.

3.5.2 Fuselage Surface

The surface tangent to the fuselage is generated by extending the wing mesh one grid point into the fuselage. The intersection points of the wing grid lines and the surface of the fuselage are calculated along the wing-fuselage intersection line and an additional line above it. These lines are defined as follows: all the lines in the J direction (see Figure 3.7), where $K = 1$, and I starts before the upper trailing edge point and ends after the lower trailing edge point. This process is repeated for a prescribe value of K that would ensure that the collar grid completely surrounds the holes on the surface of the fuselage. Straight lines are generated based on the pairs of points with the same I and J values, and a prescribed number of points are distributed along each line. At this stage, all of them are inside the fuselage. The

points are then projected onto the surface of the fuselage. The projection process is similar to finding the intersection between a line and a surface and is described below. By choosing an exponential distribution function, the points on the defined surface are clustered toward the wing-fuselage intersection.

3.5.3 Intersection Between a Line and a Discrete Surface

As was mentioned above, the fuselage surface is comprised of a series of grid points that are the intersections between grid lines of the wing mesh and the surface of the fuselage mesh. The intersection point can be easily found assuming both line and surface are know analytical functions.

Let $P_1 = \{x_1, y_1, z_1\}$ and $P_2 = \{x_2, y_2, z_2\}$ be the points that define a line and $\vec{V_1}$ and $\vec{V_2}$ be the position associated with P_1 and P_2 relative to zero, then the line passing through the points is given by

$$\vec{V_l} = t\vec{V_1} + (1-t)\vec{V_2} \tag{3.1}$$

where $-\infty < t < \infty$. Let $f(x, y, z) = 0$ be the definition of a surface, then the intersection between the line $\vec{V}$ and the surface f is given by $f(\vec{V}) = 0$. Depending on the function f, the intersection point can be found analytically or numerically.

In our case, the fuselage is given by discrete points and therefore the intersection point must be found numerically. An efficient method to locate the intersection point is based on a bilinear interpolation within a cell of the fuselage mesh. Let P_3, P_4, P_5, and P_6 be the point that define a surface cell and V_3, V_4, V_5, and V_6 be their respective positions relative to zero, then a point in the cell is given by

$$\vec{V_c} = s(r\vec{V_3} + (1-r)\vec{V_4}) + (1-s)(r\vec{V_5} + (1-r)\vec{V_6}) \tag{3.2}$$

where $0 < r, s < 1$.

If a line intersects a certain cell then $V_l = V_c$ and $0 < r, s < 1$. The parameters r, s, and t can be found by the solution of the 3×3 system given by equations 3.1 and 3.2. If $0 < s < 1$ the intersection point is between the points P_1 and P_2, and

the intersection point is found by a linear interpolation, otherwise the point is found through extrapolation. If r, or s are not between zero and one then the line does not intersect the particular cell. The values of r, and s are then used to move to the next cell and the system is solved until the intersection point is found.

3.5.4 Wing Surface

The surface tangent to the wing is generated following a similar process to that used for generating the fuselage surface. The points that define the wing-fuselage intersection are used together with points on the wing surface to generate the lines that are inside or outside the wing, depending on the wing's curvature. Points are exponentially distributed on these lines and then projected onto the wing surface. The surface is also extended in the $\pm I$ direction to cover the fuselage hole.

3.5.5 Volume Boundaries

The other four boundaries of the collar grid are generated based on the surface boundaries. The edges of two of the surfaces are normal to the fuselage and the wing surface, respectively, and the other two are set so they smoothly close the volume covered by the collar grid. It is achieved by using the same projection algorithms used for the surface boundary generation.

3.5.6 Grid Generation

Once the boundaries are defined, internal points are computed through interpolations based on the boundary values. The points are smoothed using an elliptic grid generator based on the TTM algorithm [38]. The smoothing procedure is applied to $I = constant$ slices to ensure a smooth orthogonal mesh. Using the TTM algorithm guarantees that in each I slice the coordinates are orthogonal and smooth. Control functions are used to ensure a point distribution similar to that of the boundaries. The whole process requires negligible amount of computer time since only a few iterations are needed for convergence.

The resulting collar grid is smooth and it maintains orthogonality everywhere it is defined even in the region close to the intersection line. In addition, its points are clustered toward the wing and the fuselage surfaces in a way that allows high resolution of the intersection region.

3.5.7 Force and Moment Calculations

Collar grids maybe also used to calculate the force and moments of the surface grid cells that are cut by the hole cutting procedure. A suite of routines allows to project collar grid surfaces on to the appropriate body components and to set a special hole array in the opposite manner. That way, a simple use of the force calculation routines allow to add the contribution of the surfaces of the holes to the total forces and moments.

3.6 Hyperbolic Collar Grids

In certain cases, it is beneficial to generate collar grids hyperbolically. In such cases, the surface mesh is generated either by marching along the fuselage and wing surface or, it is generated by the elliptic collar grid generator. A hyperbolic grid can have either a C type topology or an O type topology. The latter is easy to generate and handle but sometimes an O type mesh does not provide an adequate solution. On the other hand, the C type topology for a hyperbolic collar grid dictates a somewhat different grid topology at the fuselage side. This topology is marked by setting the $IWING$ parameter to "4" at the appropriate end of the wing. Figure 3.8 shows a C type hyperbolic collar grid. The $IWING$ parameter of this mesh is set to $IWING = 41$.

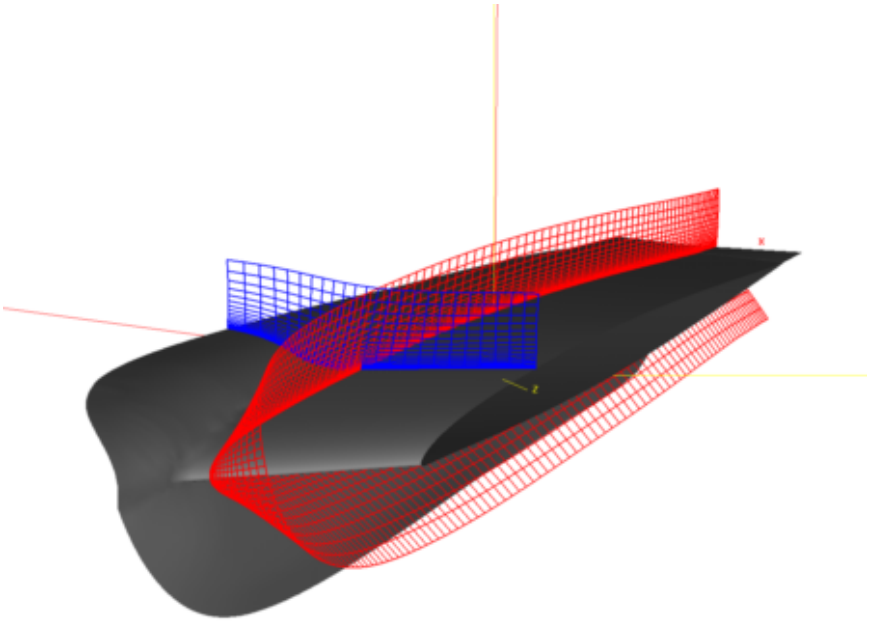

Figure 3.8: C type hyperbolic collar grid

Chapter 4

Boundary Conditions

4.1 Introduction

Version 2.5 of the EZNSS code contains a wide variety of boundary conditions. The boundary conditions are automatically set by the code, based on the identification of the topology. The user may over-ride any of the boundary conditions using the input file.

4.2 Wall Boundary Conditions

The boundary conditions applied in the inviscid, Euler equations simulations are set to obtain a slip condition at all walls. All surfaces are assumed to be adiabatic and the wall boundary conditions consist of an impermeable wall, with a zeroth-order zero normal pressure gradient, and an adiabatic wall boundary condition (usually applied at $\zeta = 1$).

The boundary conditions applied in both laminar-flow and turbulent-flow calculations are identical and are chosen to simulate viscous flows. The wall boundary conditions, applied at $\zeta = 1$, enforce zero slip and adiabatic wall conditions. The contravariant velocities U, V, W are all set to zero, and a zeroth-order zero normal pressure gradient condition is applied.

The above conditions correspond to a non-accelerating geometry. When acceler-

ation is apllied, to the whole geometry or parts of it, an additional term is added to the pressure on the body, since the normal pressure gradient is not longer zero. In such cases the pressure gradient is calculated as follows.

$$\frac{\partial p}{\partial n} = \rho \vec{a} \cdot \hat{n} \tag{4.1}$$

where ρ is the fluid density, $\hat{n}$ is the normal to the geometry, and $\vec{a}$ is the acceleration.

4.3 Inflow, Outflow, and Extrapolations

The inflow boundary condition, applied at $\zeta = \zeta_{max}$, enforces free-stream conditions, while exit boundary conditions utilize a simple zeroth-order zero-gradient extrapolation condition. In general, for all extrapolations, a zeroth-order scheme is used. This includes the zero normal pressure gradient condition at walls.

4.3.1 Characteristic-like Inflow Outflow

To improve the inflow or boundary conditions, the characteristic relations that are due-to Turkel have been added. For supersonic inflow, the flow quantities at the inflow boundary (implemented in the I direction only) are set based on the current values of the corresponding boundary. For subsonic incoming flows, the density and velocity components are taken as the current values and the pressure is extrapolated from the neighboring cell (zeroth order extrapolation). For supersonic outflow, all variables are extrapolated using a zeroth order extrapolation while for subsonic flow the outflow pressure is prescribed by the user and the density is based on the formula:

$$\rho_1 = \rho_2 + \left(p_{exit} - p_2\right)/a_2^2\right) \tag{4.2}$$

where the subscript 1 denotes the boundary and the subscript 2 denotes the neighboring cell (a being the speed of sound).

The same scheme is also implemented for the $K = K_{max}$. In that case the procedure is as follows. For each point, it is determined wether the flow is an inflow

or outflow. Then it is determined whether it is supersonic or subsonic. And finally, the appropriate boundary condition is applied (by setting the appropriate $KOUT$ to 100).

4.4 Periodic Boundary Conditions

When a periodic boundary condition is required it is applied at $\eta = \eta_{max}$, allowing the flow to become asymmetric if warranted by the flow physics. Application of the periodic boundary condition results in a system of periodic equations for the respective factor. Computationally, the solution of the periodic set is very expensive but it is necessary in order to capture asymmetric flow solutions as well as symmetric solutions.

4.5 Cut Conditions in I Direction

The $ICUT$ variable is used to set cut conditions in either I_{min}, I_{max}, or both. The cut condition is employed using a simple interpolation between both sides of the cut (taking the value from $K = 2$ for the interpolation.

4.6 Axis Conditions

Axis conditions can be employed at I_{min} or I_{max}, or at K_{min}. The conditions in the I direction are usually used for tips of slender bodies where the conditions in the K direction are used for the center of tubes. The flow values at the axis are determined by interpolation.

4.7 Symmetry Conditions

Symmetry conditions may be emplyoed in the I or J directions at each end or at both ends. Symmetry is employed about the $Y = 0$ plane only and the grid must

have a section on that plane to achieve the appropriate symmetry. For example, if symmetry is employed at J_{min}, the $J = 2$ section must be placed on the $Y = 0$ plane.

4.8 Free Stream Conditions

Free stream conditions may be employed on any boundary surface. In the current version, nothing is set and therefore the free stream conditions remain the same. The free stream flags may be used to turn off zonal boundary conditions on the appropriate faces. Future versions may include the capability to change the free stream conditions during restart.

4.9 Inlet Conditions

Inlet conditions are available in the I direction where the surface is primary aligned with the X direction. The inlet conditions are implemented through charcteristic relations.

4.10 Zonal Boundary Conditions

Zonal boundary conditions are handled in an automatic manner. The EZNSS code includes a suite of routines to calculate inter grid communications. These routines are invoked every time the configuration experiences a geometry change. All inner boundaries (holes edges), and outer boundaries (usually at $\zeta = \zeta_{max}$) are updated using a tri-linear interpolation.

Chapter 5

Six Degrees of Freedom Simulation

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

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

5.1 Translational Equation of Motion

The translational equation of motion is obtained by applying Newton's law of motion. Since it is assumed that the mass of an aircraft or its stores remain constant, the law

takes the form:

$$\sum \vec{F} = m\frac{d\vec{V}}{dt} \tag{5.1}$$

where $\sum \vec{F}$ is the sum of forces (aerodynamic and other) acting on the body, m is the mass of the store, and $\vec{V}$ is the velocity vector. The translational velocity of a certain store at a time step $n+1$ is calculated using forward differences arriving at:

$$\vec{V}^{n+1} = \vec{V}^{n} + \frac{\Delta t}{m}\left(\frac{3}{2}\sum \vec{F}^{n} - \frac{1}{2}\sum \vec{F}^{n-1}\right) \tag{5.2}$$

The motion increment is then evaluated using

$$\Delta\vec{X} = \Delta t\frac{\left(\vec{V}^{n+1} + \vec{V}^{n}\right)}{2} \tag{5.3}$$

The motion increment is applied at the center of gravity of the corresponding body. In practice, this means that every grid point is translated by $\Delta\vec{X}$.

5.2 Rotational Equation of Motion

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

$$\sum \vec{M} = \mathbf{I}\frac{d\vec{\Omega}}{dt} + \vec{\Omega} \times \vec{H} \tag{5.4}$$

where $\sum \vec{M}$ is the sum of moments, $\mathbf{I}$ is the moment of inertia matrix, $\vec{\Omega}$ is the angular velocity vector, and $\vec{H}$ is the angular momentum vector. The aerodynamic moments,

evaluated in Cartesian coordinates, are transformed into body coordinates using:

$$\begin{bmatrix} M_1 \\ M_2 \\ \\ M_3 \\ \\ \end{bmatrix} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \begin{pmatrix} \cos\psi\sin\theta\sin\phi \\ -\sin\psi\cos\phi \end{pmatrix} & \begin{pmatrix} \sin\psi\sin\theta\sin\phi \\ +\cos\psi\cos\phi \end{pmatrix} & \cos\theta\sin\phi \\ \begin{pmatrix} \cos\psi\sin\theta\cos\phi \\ +\sin\psi\sin\phi \end{pmatrix} & \begin{pmatrix} \sin\psi\sin\theta\cos\phi \\ -\cos\psi\sin\phi \end{pmatrix} & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ \\ M_z \\ \\ \end{bmatrix} \quad (5.5)$$

In addition to the above mentioned assumptions, it is further assumed that a typical store has at least one symmetry plane. In its local coordinates, out of the three off-diagonal terms, I_{xz} is allowed to be non-zero. The other two terms I_{xy} and I_{yz} must be zero. In light of this simplification, the angular acceleration may be calculated by:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{M_1-(\dot{r}+pq)I_{xz}-qr(I_z-I_y)}{I_x} \\ \frac{M_2-\left(r^2-p^2\right)I_{xz}-pr(I_x-I_z)}{I_y} \\ \frac{M_3-(\dot{p}-qr)I_{xz}-pq(I_y-I_x)}{I_z} \end{bmatrix} \quad (5.6)$$

where p, q, and r are the angular velocity components in body coordinates. The angular velocity of a certain store at a time step $n+1$ is also calculated using forward differences arriving at:

$$\begin{bmatrix} p^{n+1} \\ q^{n+1} \\ r^{n+1} \end{bmatrix} = \begin{bmatrix} p^n \\ q^n \\ r^n \end{bmatrix} + \Delta t \left\{ \frac{3}{2} \begin{bmatrix} \dot{p}^n \\ \dot{q}^n \\ \dot{r}^n \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \dot{p}^{n-1} \\ \dot{q}^{n-1} \\ \dot{r}^{n-1} \end{bmatrix} \right\} \quad (5.7)$$

The angular velocities in Cartesian coordinates ($\dot{\psi}$, $\dot{\theta}$, and $\dot{\phi}$) may be calculated using the angular velocities in body coordinates (p, q, and r) by applying the following transformation:

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta} \\ q\cos\phi - r\sin\theta \\ p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \end{bmatrix} \quad (5.8)$$

5.2.1 Quaternions

Euler angles present the well-know pitch singularity. When the angle θ approaches values of $\theta = \pm 90^\circ$ attitude errots become noticeable. To resolve this difficulty, quaternions have been introduced into the 6 degrees of freedom simulation. The quaternion consists of four parameters: a scalar (q_0), and a vector (q_1,q_2,q_3). An extra constraint is required, $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. This constraint can be easily enforced through a normalization. Quaternion update is conducted through four first order differential equations as follows:

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{5.9}$$

Users may choose whether to use quaternions or Euler angles. When quaternions are chosen the Euler angles are used just for presentation of the simulation results.

5.3 Store Location Update

Following the solution of the translational equation of motion and the rotational equation of motion, the position and attitude of the store may be updated using the relations:

$$\begin{aligned} x_{cg}^{n+1} &= x_{cg}^{n} + \Delta x \\ y_{cg}^{n+1} &= y_{cg}^{n} + \Delta y \\ z_{cg}^{n+1} &= z_{cg}^{n} + \Delta z \\ \psi^{n+1} &= \psi^{n} + \Delta t\ \dot{\psi} \\ \theta^{n+1} &= \theta^{n} + \Delta t\ \dot{\theta} \\ \phi^{n+1} &= \phi^{n} + \Delta t\ \dot{\phi} \end{aligned} \tag{5.10}$$

where $\dot{\psi}$, $\dot{\theta}$, $\dot{\phi}$, are evaluated using

$$\begin{aligned} \dot{\psi} &= \frac{\dot{\psi}^{n+1} + \dot{\psi}^{n}}{2} \\ \dot{\theta} &= \frac{\dot{\theta}^{n+1} + \dot{\theta}^{n}}{2} \\ \dot{\phi} &= \frac{\dot{\phi}^{n+1} + \dot{\phi}^{n}}{2} \end{aligned} \tag{5.11}$$

The new grid location of a certain store is obtained by applying the above relations to each grid point. The application of the rotation is conducted about the center of gravity of the store.

Chapter 6

Static and Dynamic Aeroelasticity

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

$$\{u\} = \begin{bmatrix} \Phi_R & \Phi_E \end{bmatrix} \begin{Bmatrix} \xi_R \\ \xi_E \end{Bmatrix} \tag{6.1}$$

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

$$\left([K] - \omega_i^2 [M]\right) \{\Phi_i\} = \{0\} \tag{6.2}$$

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

The EZNSS code utilizes a modal structural model for interfacing the flow solver

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

6.1 Static Aeroelasticity

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

$$\{u_E\} = [\phi_E]\{\xi_E\} \tag{6.3}$$

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

$$[K_E]\{\xi_E\} = \{F_E\} \tag{6.4}$$

where $[K_E]$ is the generalized stiffness matrix associated with $[\phi_E]$, and $\{F_E\}$ is the associated generalized aerodynamic force vector. Orthogonality of the rigid-body and elastic modes with respect to the structural mass and stiffness matrices implies that $[K_E]$ is diagonal and that inertia relief effects in the right hand side of Eq. (6.4) are taken care of automatically. The rigid-body counterpart of Eq. (6.4) is used below in the maneuver trim process.

The generalized forces in Eq. (6.4) are obtained by

$$\{F_E\} = [\phi_A]^T (\{F_A\} - \{F_C\}) \tag{6.5}$$

where $\{F_A\}$ is the aerodynamic force vector at the aerodynamic surface grid points, $\{F_C\}$ is the aerodynamic force vector associated with the reference aerodynamic shape, and $[\phi_A]$ is the elastic modes matrix, expressed at the aerodynamic inter-

face grid points as described below. It is assumed that the reference loads vector $\{F_C\}$ has been defined in a previous aerodynamic analysis. This is the case when the reference aerodynamic shape is the "cruise shape", namely the shape designed for best aerodynamic performance at nominal cruise conditions. The reference aerodynamic shape can also be defined as the "jig shape", namely the unstressed shape, for which $\{F_C\} = 0$.

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

6.2 Equations of Motion for Uncoupled Aeroelastic Motion

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

$$[M_{ii}]\left\{\ddot{\xi}\right\} + \left[2\zeta M_{ii}\sqrt{\Omega_i}\right]\left\{\dot{\xi}\right\} + [M_{ii}\Omega_i]\{\xi\} = \{F(t)\} \tag{6.6}$$

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

6.2.1 Numerical Scheme

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

This results in the following set of equations:

$$\Delta\ddot{\xi}_i + 2\zeta\sqrt{\Omega_i}\Delta\dot{\xi}_i + \Omega_i\Delta\xi_i = \frac{\Delta F_i}{M_{ii}} \tag{6.7}$$

one equation for each mode.

6.2.2 Solution Method

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

$$\begin{aligned} \Delta\dot{\xi}_i &= \Delta\eta_i \\ \Delta\dot{\eta}_i &= \frac{\Delta F_i}{M_{ii}} - 2\zeta\sqrt{\Omega_i}\Delta\eta_i - \Omega_i\Delta\xi_i \end{aligned} \tag{6.8}$$

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

6.3 Mesh Deformations using the TFI Approach

The TFI approach utilized in the EZNSS code is an arc-length TFI. In this approach, the grid coordinates are parameterized by the length of the coordinates in three directions. In each direction, the parameterization results in an array whose values extend between 0.0, at the beginning of the line and 1.0 at the end of the line. The arrays for the I direction, the J direction and the K direction are marked by P^{ξ}, P^{η}, and P^{ζ}, respectively.

A volume grid in constrained by 6 faces. Each of the faces is constrained by 4 edges and each edge is constrained by to vertices. To deform a volume grid, the edge deformations are set, then the face deformations and finally, the face deformations are used as the input to the three-dimensional TFI procedure.

6.3.1 Face Deformations

The typical computational mesh that is used in the CFD computations is a body-fitted, curvilinear mesh transformed from the physical space into a Cartesian computational space. This transformation is provided as a part of the input to the CFD program (the mesh itself is already given in the Cartesian computational space). The transformation brings the body surface onto one computational plane, referred to here, as the $K = 1$ face. The next face is found at the opposite end of the computational space and is called the $K = Kdim$ face. The other faces are the $I = 1$ face, the $I = Idim$ face, the $J = 1$ face, and the $J = Jdim$ face. Depending on the mesh topology, the body surface may fill up the whole $K = 1$ face or just a part of it. For example, in the case of the C-H type topology that is used for wings, the body surface extends only between the left and right wing tips and from the leading edge and trailing edge. Figure 6.1 shows a schematic of the computational space and the body surface. The deformations of the body surface grid points are obtained using the modal approach. The rest of the $K = 1$ face are calculated based on the surface grid points by imposing smooth grid lines throughout the surface.

Once the $K = 1$ face grid points deformations are set the rest of the face are deformed as follows. The $K = Kdim$ face deformations are set to be equal to the $K = 1$ face. Then deformations at the edges connecting between the $K = 1$ face and the $K = Kdim$ face are calculated using a one-dimensional TFI. In the K direction the one-dimensional TFI for the ΔX deformations takes the form:

$$\Delta X_{i,j,k} = \left(1 - P^{\zeta}_{i,j,k}\right) \Delta X_{ij1} + P^{\zeta}_{i,j,k} \Delta X_{i,j,Kdim} \tag{6.9}$$

Next, the deformations of the other faces are calculated using a two-dimensional TFI. The formula for the ΔX deformations on the $I = Idim$ face takes the form:

$$\begin{aligned} \Delta X_{1,j,k} \;=\; & A_{1,j,k}\Delta X_{1,j,1} + B_{1,j,k}\Delta X_{1,j,Kdim} \\ & C_{1,j,k}\Delta X_{1,1,k} + D_{1,j,k}\Delta X_{1,Jdim,k} \\ & -A_{1,j,k}C_{1,j,k}\Delta X_{1,1,1} - B_{1,j,k}C_{1,j,k}\Delta X_{1,1,Kdim} \\ & -A_{1,j,k}D_{1,j,k}\Delta X_{1,Jdim,1} - B_{1,j,k}C_{1,j,k}\Delta X_{1,Jdim,Kdim} \end{aligned} \tag{6.10}$$

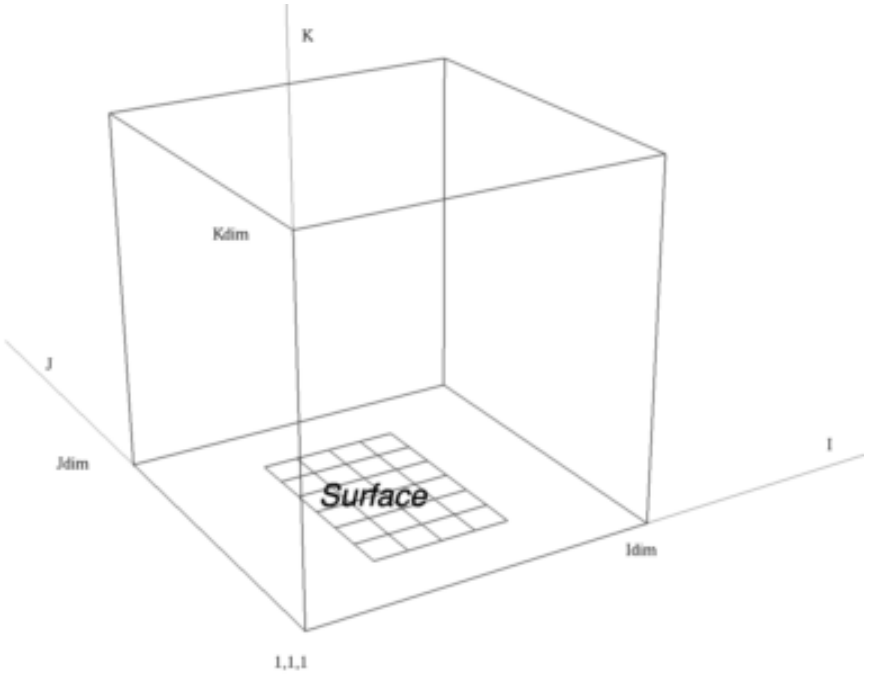

Figure 6.1: Computational space

The blending functions, A, B, C, and D are given by:

$$\begin{aligned} A_{i,j,k} &= 1 - \eta_{i,j,k} \\ B_{i,j,k} &= \eta_{i,j,k} \\ C_{i,j,k} &= 1 - \xi_{i,j,k} \\ D_{i,j,k} &= \xi_{i,j,k} \end{aligned} \tag{6.11}$$

and

$$\begin{aligned} \xi &= \frac{P^{\eta}_{1,j,1} + P^{\zeta}_{1,1,k}\left(P^{\eta}_{1,j,Kdim} - P^{\eta}_{1,j,1}\right)}{P_{1,j,k}} \\ \eta &= \frac{P^{\zeta}_{1,1,k} + P^{\eta}_{1,j,1}\left(P^{\eta}_{1,Jdim,k} - P^{\eta}_{1,1,k}\right)}{P_{1,j,k}} \\ P_{1,j,k} &= 1 - \left(P^{\eta}_{1,j,Kdim} - P^{\eta}_{1,j,1}\right)\left(P^{\zeta}_{1,Jdim,k} - P^{\zeta}_{1,1,k}\right) \end{aligned} \tag{6.12}$$

6.3.2 Volume Deformations

The face deformations are used as the input to the three-dimensional TFI whose formula is given by:

$$\Delta X_{i,j,k} = V1 + V2 + V3 - V12 - V13 - V23 + V123 \tag{6.13}$$

where

$$
\begin{aligned}
V1 &= \left(1-P^{\xi}_{i,j,k}\right)\Delta X_{1,j,k} + P^{\xi}_{i,j,k}\Delta X_{Idim,j,k} \\
V2 &= \left(1-P^{\eta}_{i,j,k}\right)\Delta X_{i,1,k} + P^{\eta}_{i,j,k}\Delta X_{i,Jdim,k} \\
V3 &= \left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{i,j,1} + P^{\zeta}_{i,j,k}\Delta X_{i,j,Kdim} \\
V12 &= \left(1-P^{\xi}_{i,j,k}\right)\left(1-P^{\eta}_{i,j,k}\right)\Delta X_{1,1,k} + \left(1-P^{\xi}_{i,j,k}\right)P^{\eta}_{i,j,k}\Delta X_{1,Jdim,k} \\
&+ P^{\xi}_{i,j,k}\left(1-P^{\eta}_{i,j,k}\right)\Delta X_{Idim,1,k} + P^{\xi}_{i,j,k}P^{\eta}_{i,j,k}\Delta X_{Idim,Jdim,k} \\
V13 &= \left(1-P^{\xi}_{i,j,k}\right)\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{1,j,1} + \left(1-P^{\xi}_{i,j,k}\right)P^{\zeta}_{i,j,k}\Delta X_{1,j,Kdim} \\
&+ P^{\xi}_{i,j,k}\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{Idim,j,1} + P^{\xi}_{i,j,k}P^{\zeta}_{i,j,k}\Delta X_{Idim,j,Kdim} \\
V23 &= \left(1-P^{\eta}_{i,j,k}\right)\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{i,1,1} + \left(1-P^{\eta}_{i,j,k}\right)P^{\zeta}_{i,j,k}\Delta X_{i,1,Kdim} \\
&+ P^{\eta}_{i,j,k}\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{i,Jdim,1} + P^{\eta}_{i,j,k}P^{\zeta}_{i,j,k}\Delta X_{i,Jdim,Kdim} \\
V123 &= \left(1-P^{\xi}_{i,j,k}\right)\left(1-P^{\eta}_{i,j,k}\right)\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{1,1,1} \\
&+ \left(1-P^{\xi}_{i,j,k}\right)\left(1-P^{\eta}_{i,j,k}\right)P^{\zeta}_{i,j,k}\Delta X_{1,1,Kdim} \\
&+ \left(1-P^{\xi}_{i,j,k}\right)P^{\eta}_{i,j,k}\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{1,Jdim,1} \\
&+ P^{\xi}_{i,j,k}\left(1-P^{\eta}_{i,j,k}\right)\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{Idim,1,1} \\
&+ \left(1-P^{\xi}_{i,j,k}\right)P^{\eta}_{i,j,k}P^{\zeta}_{i,j,k}\Delta X_{1,Jdim,Kdim} \\
&+ P^{\xi}_{i,j,k}\left(1-P^{\eta}_{i,j,k}\right)P^{\zeta}_{i,j,k}\Delta X_{Idim,1,Kdim} \\
&+ P^{\xi}_{i,j,k}P^{\eta}_{i,j,k}\left(1-P^{\zeta}_{i,j,k}\right)\Delta X_{Idim,Jdim,1} \\
&+ P^{\xi}_{i,j,k}P^{\eta}_{i,j,k}P^{\zeta}_{i,j,k}\Delta X_{Idim,Jdim,Kdim}
\end{aligned}
\tag{6.14}
$$

Following the volume mesh deformations calculation, the mesh is updated using a simple addition using:

$$
\begin{aligned}
X_{i,j,k} &= X_{i,j,k} + \Delta X_{i,j,k} \\
Y_{i,j,k} &= X_{i,j,k} + \Delta Y_{i,j,k} \\
Z_{i,j,k} &= X_{i,j,k} + \Delta Z_{i,j,k}
\end{aligned}
\tag{6.15}
$$

Chapter 7

Parallelization

7.1 Introduction

Parallel computer architectures are an essential tool in simulating the flow about complex aircraft configurations, especially when the flow is unsteady or when the geometry changes. The ever growing computational meshes and the accompanying vast amount of numerical operations require the use of parallel processing. Current applications utilize shared and distributed memory architectures along with the appropriate application program interface (API). The OpenMP API [39] was introduced as an industry standard to support shared memory programs with simple directives. Simple directives facilitate the loop level parallelism. The message passing interface (MPI) [40] is an example of the support of distributed memory programming. Another example is the parallel virtual machine (PVM). [41] In contrast to the shared memory architecture, distributed memory requires to hand-write pieces of code to facilitate the parallel processing.

The OpenMP programming paradigm provides the means to easily introduce parallelism when developing new applications, or even when parallelizing an existing code. OpenMP, however, is limited to the loop level and most compilers do not allow directive nesting and are thus limited to a single parallelism level. As a result, scalability is limited and the efficiency rapidly degrades when using a large number of processors. Moreover, even compilers that do allow the use of nested directives do

not automatically account for all the critical issues that arise when parallelizing an existing code.

Scalability is mostly achieve by using application specific code and the MPI libraries. Besides being difficult, the use of MPI is error-prone. Furthermore, using MPI usually requires to utilize domain decomposition methods. It is quite common that the physical space is arbitrarily split to accommodate the parallelization. Computational blocks that result from the decomposition procedure communicate with each other using MPI to transfer data to and from the boundaries of the blocks. When using explicit algorithms, the decomposition does not hinder the convergence nor does it affect the development of the flow. In contrast, domain decomposition may slow down the convergence of implicit algorithms. Furthermore, when simulating unsteady flows, the decomposition may alter the solution behavior and accuracy.

The recent emergence of shared memory parallel architectures, such as the SGI Altix Linux based servers, provide the means for significant scalability. However, as mentioned above, when using single-level parallelism (with OpenMP directives), the parallelization efficiency dramatically decreases when the number of processors increases. In recent years, the growing interest in multi-level parallelism (MLP) resulted in various strategies. From using system level calls (such as "fork" and "join"), to hybrid approaches using MPI and OpenMP, to multiple levels of parallelism using OpenMP only. [42] Reference [42] also includes a review of previous multi-level parallelism efforts. The current paper describes a new multi-level parallelism approach that allows to use the OpenMP directives and to divide the loads between groups of processors in such a way that the parallelization efficiency drop may be greatly reduced. Moreover, the current approach provides the means to convert a single level parallelism code to a multi-level parallelism code with minor changes to the original code.

7.2 Single-Level Parallelism

Using single level OpenMP parallelism, one could either parallelize the code in the zonal level or the loop level. Zonal level parallelism is limited since the largest number

of processors that may be used is the number of zones, namely one processor handles one zone. Such a parallelization scheme is efficient only when the sizes of the zones are similar. This rarely occurs. Moreover, quite often, the number of zones may be small, especially when simulating simple geometries. For example, the simulation of the flow about a wing or a simple slender body may require only one zone. And even when the number of zones is large, a balanced load may be achieved only after several zones are grouped together. Each group of zones is then handled by a certain single processor.

Consequently, a scalable parallelization may be achieved only on the loop level. Of course, one would like to parallelize the outer most loop or loops of the program, loops that contain the bulk load of the solver. Each section of the right hand side, namely, the inviscid section, the smoothing section (if applicable), the viscous section, and the turbulence modeling section may be separately parallelized quite easily. Parallelization of the left hand side, namely the inversion of the left hand side matrix, is facilitated by the alternate direction nature of the approximate factorization (it is in fact an ADI scheme). Following the factorization, the inversion of the matrix is replaced by a series of line inversions that are independent from each other. Therefore, the parallelization of the left hand side is also easily carried out. All of the above is conducted by a relatively small number of OpenMP directives.

The following example, taken directly from the EZNSS code, illustrates the ease of parallelization using OpenMP directives. As explained above, the outer most loop is parallelized to reduce the overhead and to avoid was is term as "granularity." The variables that are defined by the "PRIVATE" clause are multiplied during run time, based on the number of processors that available for the specific task. The efficiency of such parallelization depend on two factors. The first is the length of the outer most loop with respect to the number of available processors. If the loop is short and the number of processors does not evenly divide into the loop length, the efficiency is expected to be low. The second factor is the amount of work that is conducted in the inner most loop. A small amount of work, compared to the whole program, would result in low efficiency. High efficiency may be achieved only when both factors favorably affect the parallelization.

```
C$OMP PARALLEL DO PRIVATE(I,J,K,WA,A,B,C,WA0,WA1,WA2,WA3,WA4)
C$OMP&            SHARED(MDIM,IDIM,JDIM,KDIM,IB,IE,IVIS)
                DO K=KB,KE
                   DO J=JB,JE
                      DO I=IB,IE
                         ....
                         Some work
                         ....
                      END DO
                   END DO
                END DO
C$OMP END PARALLEL DO
```

Consequently, as mentioned above, the efficiency of the loop level parallelism drops when using a large number of processors. The reason is two-fold: first, certain processors have to access the memory that resides in another node that may be physically further away; and second, the remainder of a loop may become significant. For example, when a loop of the size of 50 is parallelized and 16 processors are used, 14 of the processors may wait for the completion of the last 2.

7.3 Multi-Level Parallelism

The ideal solution would be to use zonal parallelization along with loop level parallelization. Such an ideal solution would utilize OpenMP directives. The choice of OpenMP is due to its aforementioned simplicity. However, not all compilers support the nesting of parallel constructs in OpenMP. Moreover, even the compilers that do allow to have nested directives, do not always account for the appropriate duplication of works arrays (through the "PRIVATE" directive). The recent version of the Intel Fortran compiler allows the nesting of OpenMP directives and it is the one that is used in the current work. [43]

The first part of the multi-level parallelization entails the grouping of the zones into groups as prescribed by the user. This is the outer parallelization level. Splitting

into groups that would have an approximately balanced computational load may be achieved by using a rather simple algorithm. The second part has already existed in the code, namely the loop level parallelism. All that is left to do is to add a few more directives to set the nested parallelism, to prescribe the number of groups, and to prescribe the number of threads for each group (the total number of processors being the number of groups times the number of threads).

As mentioned above, not all compilers fully account for the need to duplicate work arrays and temporary variables. Since the number of total processors and the amount of required memory space are known, one could allocated the appropriate amount of memory and to set the memory for each group or thread as needed.

The following example illustrates the introduction of multi level parallelism into the EZNSS code. The original single level code had a simple loop for solving the flow in all the zones as follows:

```
! Zonal loop
            DO N = 1, NZ
               ....
               SOLVER
               ....
            END DO
! End zonal loop
```

The flow solver part was parallelized exactly as described in the previous section.

As mentioned above, to achieve a balanced multi level parallelism, the zones have to be grouped where each of the groups has a comparable computational load. In the development process, it was found that the computational time per grid point grows as the zone becomes larger. Therefore a weighing strategy was adopted as a part of the grouping algorithm. The number of zones per each group is stored in an appropriate array and another array stores the list of zones that belong to each groups. The following code shows how the arrays are utilized to determine the appropriate zone for each outer level. The SOLVER part remains parallelized as described above as it comprises the inner parallelization level. The rest of the changes are OpenMP

calls and directives that are used to enable nested parallelism, to set the number of threads for each level, and to actually parallelize the outer loop. The number of total threads is of course MLPOUTER * MLPINNER.

```
! Start of multi level parallelism section
           CALL OMP_SET_NESTED(.TRUE.)
! Outer parallelization level
           CALL OMP_SET_NUM_THREADS(MLPOUTER)
!$OMP PARALLEL DO PRIVATE(NMLP,IMEM,N)
           DO NMLP = 1, MLPOUTER
              CALL OMP_SET_NUM_THREADS(MLPINNER)
              DO IMEM = 1, IGRPSIZE(NMLP)
                 N = IGRPMEMS(NMLP,IMEM)
                    ....
                    SOLVER
                    ....
              END DO
! End zonal loop
           END DO
C$OMP END PARALLEL DO
!End group
```

Ideally, the above code should have been sufficient to accommodate the MLP scheme. However, the Intel compiler, does not multiply work arrays twice. Fortunately, this deficiency does not present a major obstacle. Since the user has to prescribe the number of inner and outer threads, the total number of processors is known and therefore the exact size of the work arrays is also known. These are allocated and appropriately transferred to each of the groups. They are later multiplied by the OpenMP directives of the inner parallelization level. Thus the SOLVER part remains unchanged.

Chapter 8

Summary

This version of the manual is the first one that contains a user manual and a partial test case description. This version of the code contains many of the long waited features such as store separation from a pivot release point, a line Gauss-Seidel marching scheme, Spalart-Allmaras and $k-\omega$-SST turbulence model, and an embedded spline capability. As the code is still being developed, with a long list of feature requests, new versions with new features shall be available in the near future.

Appendix A

Jacobian Matrices

A.1 Inviscid Flux Vectors Jacobian Matrices

$$\hat{A}, \hat{B}, \hat{C} = \begin{bmatrix} k_t & k_x & k_y & k_z & 0 \\ k_x\phi^2 - u\theta & k_t + \theta - k_x\gamma_2 u & k_y u - \gamma_1 k_x v & k_z u - \gamma_1 k_x w & k_x\gamma_1 \\ k_y\phi^2 - v\theta & k_x v - k_y\gamma_1 u & k_t + \theta - k_y\gamma_2 v & k_z v - \gamma_1 k_y w & k_y\gamma_1 \\ k_z\phi^2 - w\theta & k_x w - k_z\gamma_1 u & k_y w - k_z\gamma_1 v & k_t + \theta - k_z\gamma_2 w & k_z\gamma_1 \\ \theta(2\phi^2 - \frac{\gamma e}{\rho}) & k_x\beta - \gamma_1 u\theta & k_y\beta - \gamma_1 v\theta & k_z\beta - \gamma_1 w\theta & k_t + \gamma\theta \end{bmatrix} \quad \text{(A.1)}$$

where

$$\begin{aligned} \phi^2 &= 0.5(\gamma - 1)(u^2 + v^2 + w^2) \\ \theta &= k_x u + k_y v + k_z w \\ \gamma_1 &= \gamma - 1 \\ \gamma_2 &= \gamma - 2 \\ \beta &= \frac{\gamma e}{\rho} - \phi^2 \end{aligned}$$

with $k = \xi$ to obtain $\hat{A}$, $k = \eta$ to obtain $\hat{B}$, and $k = \zeta$ to obtain $\hat{C}$.

A.2 Jacobian Matrix of the Viscous Flux Vector

$$\hat{A}_v^{\xi}, \hat{B}_v^{\eta}, \hat{C}_v^{\zeta} = \frac{1}{J} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ m_{21} & \alpha_1 \delta_k \left(\rho^{-1}\right) & \alpha_2 \delta_k \left(\rho^{-1}\right) & \alpha_3 \delta_k \left(\rho^{-1}\right) & 0 \\ m_{31} & \alpha_2 \delta_k \left(\rho^{-1}\right) & \alpha_4 \delta_k \left(\rho^{-1}\right) & \alpha_5 \delta_k \left(\rho^{-1}\right) & 0 \\ m_{41} & \alpha_3 \delta_k \left(\rho^{-1}\right) & \alpha_5 \delta_k \left(\rho^{-1}\right) & \alpha_6 \delta_k \left(\rho^{-1}\right) & 0 \\ m_{51} & m_{52} & m_{53} & m_{54} & \alpha_0 \delta_k \left(\rho^{-1}\right) \end{bmatrix} J \quad \text{(A.2)}$$

where

$$
\begin{aligned}
m_{21} &= \alpha_1\delta_k\left(-u/\rho\right)+\alpha_2\delta_k\left(-v/\rho\right)+\alpha_3\delta_k\left(-w/\rho\right)\\
m_{31} &= \alpha_2\delta_k\left(-u/\rho\right)+\alpha_4\delta_k\left(-v/\rho\right)+\alpha_5\delta_k\left(-w/\rho\right)\\
m_{41} &= \alpha_3\delta_k\left(-u/\rho\right)+\alpha_5\delta_k\left(-v/\rho\right)+\alpha_6\delta_k\left(-w/\rho\right)\\
m_{51} &= \alpha_1\delta_k\left(-u^2/\rho\right)+\alpha_2\delta_k\left(-2uv/\rho\right)+\alpha_3\delta_k\left(-2uw/\rho\right)\\
&+ \alpha_4\delta_k\left(-v^2/\rho\right)+\alpha_6\delta_k\left(-w^2/\rho\right)+\alpha_3\delta_k\left(-2vw/\rho\right)\\
&+ \alpha_0\delta_k\left(-e/\rho^2\right)+\alpha_0\delta_k\left[\left(u^2+v^2+w^2\right)/\rho\right]\\
m_{52} &= -m_{21}-\alpha_0\delta_k\left(u/\rho\right)\\
m_{53} &= -m_{31}-\alpha_0\delta_k\left(v/\rho\right)\\
m_{54} &= -m_{41}-\alpha_0\delta_k\left(w/\rho\right)\\
\alpha_0 &= \frac{\gamma\kappa}{c_p}\left(k_x^2+k_y^2+k_z^2\right)\\
\alpha_1 &= \mu\left(\frac{4k_x^2}{3}+k_y^2+k_z^2\right)\\
\alpha_2 &= \left(\frac{\mu k_x k_y}{3}\right)\\
\alpha_3 &= \left(\frac{\mu k_x k_z}{3}\right)\\
\alpha_4 &= \mu\left(k_x^2+\frac{4k_y^2}{3}+k_z^2\right)\\
\alpha_5 &= \left(\frac{\mu k_y k_z}{3}\right)\\
\alpha_6 &= \mu\left(k_x^2+k_y^2+\frac{4k_z^2}{3}\right)
\end{aligned}
$$

with $k=\xi$ to obtain $\hat{A}_v^{\xi}$, $k=\eta$ to obtain $\hat{B}_v^{\eta}$, and $k=\zeta$ to obtain $\hat{C}_v^{\zeta}$.

Note that for the dimensionless form, the term α_0 becomes:

$$\alpha_0 = \frac{\gamma\overline{\kappa}}{\overline{c}_p Pr_\infty}\left(k_x^2+k_y^2+k_z^2\right)$$

Appendix B

Jacobian Matrices Eigensystems

B.1 Eigenvectors

$$T_k = \begin{bmatrix} \tilde{k}_x & \tilde{k}_y & \tilde{k}_z & \alpha & \alpha \\ \tilde{k}_x u & \tilde{k}_y u - \tilde{k}_z \rho & \tilde{k}_z u + \tilde{k}_y \rho & \alpha(u + \tilde{k}_x c) & \alpha(u - \tilde{k}_x c) \\ \tilde{k}_x v + \tilde{k}_z \rho & \tilde{k}_y v & \tilde{k}_z v - \tilde{k}_x \rho & \alpha(v + \tilde{k}_y c) & \alpha(v - \tilde{k}_y c) \\ \tilde{k}_x w - \tilde{k}_y \rho & \tilde{k}_y w + \tilde{k}_x \rho & \tilde{k}_z w & \alpha(w + \tilde{k}_z c) & \alpha(w - \tilde{k}_z c) \\ \tilde{k}_x \phi_1 + \beta_1 & \tilde{k}_y \phi_1 + \beta_2 & \tilde{k}_z \phi_1 + \beta_3 & \alpha(\phi_2 + c\tilde{\theta}) & \alpha(\phi_2 - c\tilde{\theta}) \end{bmatrix} \tag{B.1}$$

where

$$
\begin{aligned}
\tilde{k}_x &= \frac{k_x}{(k_x^2+k_y^2+k_z^2)^{1/2}} \\
\tilde{k}_y &= \frac{k_y}{(k_x^2+k_y^2+k_z^2)^{1/2}} \\
\tilde{k}_z &= \frac{k_z}{(k_x^2+k_y^2+k_z^2)^{1/2}} \\
\tilde{\theta} &= \tilde{k}_x u + \tilde{k}_y v + \tilde{k}_z w \\
\alpha &= \frac{\rho}{\sqrt{2c}} \\
\phi_1 &= \frac{\phi^2}{\gamma-1} \\
\phi_2 &= \frac{\phi^2+c^2}{\gamma-1} \\
\phi^2 &= 0.5(\gamma-1)(u^2+v^2+w^2) \\
\beta_1 &= \rho(\tilde{k}_z v - \tilde{k}_y w) \\
\beta_2 &= \rho(\tilde{k}_x w - \tilde{k}_z u) \\
\beta_3 &= \rho(\tilde{k}_y u - \tilde{k}_x v)
\end{aligned}
$$

and $k=\xi$ to obtain T_ξ, $k=\eta$ to obtain T_η, and $k=\zeta$ to obtain T_ζ.

B.2 Eigenvalues

$$
\begin{aligned}
\hat{\Lambda}_\xi &= D\left[U,U,U,U+c(\xi_x^2+\xi_y^2+\xi_z^2)^{1/2},U-c(\xi_x^2+\xi_y^2+\xi_z^2)^{1/2}\right] \\
\hat{\Lambda}_\eta &= D\left[V,V,V,V+c(\eta_x^2+\eta_y^2+\eta_z^2)^{1/2},V-c(\eta_x^2+\eta_y^2+\eta_z^2)^{1/2}\right] \\
\hat{\Lambda}_\zeta &= D\left[W,W,W,W+c(\zeta_x^2+\zeta_y^2+\zeta_z^2)^{1/2},W-c(\zeta_x^2+\zeta_y^2+\zeta_z^2)^{1/2}\right] \qquad \text{(B.2)}
\end{aligned}
$$

Appendix C

Input Files

C.1 Input File Names and Types

The main input file name can named arbitrarily ($eznss.i.defaults$ in the distribution). The rest of the input files are set inside the main input file. All have example templates as a part of the distribution. They are: $eznssa.i$, the array input file name, $cgg.i$, the collar grid generation input file, $6dof.i$, the 6DOF input file, $eznssvb.i$, the virtual body iput file, $ecgu.i$, the elliptic collar grid update input file, $spline.i$, the spline input file, and $flap.i$, the flap input file.

C.2 Main Input File *eznss.i.defaults*

The main input file is a classical Fortran NAMELIST file. It contains all global input for the code. Is is composed of groups based on subject. In what follows, all the groups and the input parameters are described.

VERINP - Versioning Input

Variable Name	Type	Def.	Description
IVFLAG	Int	0	IVFLAG=0 - Normal run. IVFLAG=1 - Write version information and stop.

FLOINP - Flow Conditions

Variable Name	Type	Def.	Description
ALT	Real	0.0	Altitude [meters].
ALP	Real	0.0	Angle of attack [degrees].
BET	Real	0.0	Side slip angle [degrees].
FSMACH	Real	0.84	Free stream Mach number.
REY	Real	10^6	Reynolds number; The Reynolds number entered here should be evaluated based on the real reference length that was used for normalization of the computational mesh, the free stream density, the free stream velocity, and the free stream viscosity. REY =1.0 - Dimensional simulation is invoked and the Reynolds number per meter is calculated and printed in the *eznss.out* file.
PR	Real	0.7	Prandtl number (not used by version 2.6.0 and up).
GAMMA	Real	1.4	Initial specific heats ratio (γ).
IGAMMAF	Int	0	Variable γ flag. IGAMMAF = 0 - Constant γ. IGAMMAF = 1 - Variable γ.
IPERFLOW	Int	0	Periodic boundary conditions. IPERFLOW =1 - Periodic boundary condition in the I coordinate direction is enforced. IPERFLOW =2 - Periodic boundary condition in the J coordinate direction is enforced. IPERFLOW =12 - Periodic boundary conditions in both the I and J coordinate directions are enforced.
ILOWMACH	Int	0	Low Mach number preconditioning flag; ILOWMACH =0 - No preconditioning. ILOWMACH =1 - Low Mach preconditioning (works properly with AUSM fluxes and MAPS Jacobians).

SMOINP - Smoothing Input

Variable Name	Type	Def.	Description
EPSE	Real	0.03	Explicit smoothing coefficient; Used only for the Beam and Warming and F3D schemes; The default value is good for subsonic flows. For transonic and supersonic flows EPSE should be set to 0.06 to 0.1.
EPSI	Real	0.03	Implicit smoothing coefficient; EPSI is set as equal to EPSE inside the program so there is no need to set it anymore.
SMOOL	Real	1.0	Smoothing coefficient reduction factor; If everything works SMOOL should be kept as 1.0 but it can be set to any value between 0.0 and 1.0.

FNSINP - File Names Input

Variable Name	Type	Def.	Description
AFNAME	String	”	Array input file name.
CGFNAME	String	”	Collar grid generation file name.
SDFNAME	String	”	6 DOF file name.
VBFNAME	String	”	Virtual body file name.
ECGUFNAME	String	”	Elliptic collar grid update file name.
SPLFNAME	String	”	Spline input file name.
FLPFNAME	String	”	Flap deflection input file name.
RTRFNAME	String	”	Rotor input file name.

TIMINP - Temporal and Spatial Accuracy

Variable Name	Type	Def.	Description
GDTI	Real	-1.0	Initial time step. GDTI < 0 - Initial CFL number. GDTI > 0 - Initial time step for first order time accurate simulation (no dual time stepping).
GDTF	Real	-1.0	Final time step. GDTF < 0 - Final CFL number; The CFL number grows from GDTI to GDTF within ISLOWS steps (see below how and where ISLOWS is defined) GDTF > 0 - Final time step for first order time accurate simulation (no dual time stepping).
FSA	Real	0.5	Flux split spatial accuracy. FSA < 0.0 - Beam and Warming algorithm is utilized (in this case the magnitude of FSA is immaterial). FSA ≥ 0.0 - (preferably at or around 0.5) and all IFSF? (see below) input arguments are zero, the F3D algorithm is utilized. FSA ≥ 0.0 and any of the IFSF? input arguments is greater than 0, flux splitting is utilized in that specific direction based on the values of IFSF? (see below). FSA $= 0.0$ - first order spatial accuracy. FSA $= 0.5$ - second order spatial accuracy (Steger Warming splitting allows blending).
DDT	Real	0.0	Time step for dual-time step time marching.
H	Real	1.0	Implicit temporal accuracy (do not change!).

METINP - Methods Input

Variable Name	Type	Def.	Description
IB2	Int	1	IB2 = 1 - Regular time marching. IB2 = 2 - B2 type time marching (available only for single time stepping and for NRK = 1).
IMET	Int	1	IMET = 0 - Explicit time marching. IMET = 1 - Implicit ADI time marching. IMET = 2 - Implicit LGS time marching.
IFSFX	Int	3	Flux splitting flag; ξ coordinate direction. IFSFX = 1 - Steger Warming FVS. IFSFX = 2 - HLLC FDS. IFSFX = 3 - AUSM+-up. IFSFX = 4 - MAPS.
IFSFY	Int	3	Flux splitting flag; η coordinate direction. IFSFY = 1 - Steger Warming FVS. IFSFY = 2 - HLLC FDS. IFSFY = 3 - AUSM+-up. IFSFY = 4 - MAPS.
IFSFZ	Int	3	Flux splitting flag; ζ coordinate direction. IFSFZ = 1 - Steger Warming FVS. IFSFZ = 2 - HLLC FDS. IFSFZ = 3 - AUSM+-up. IFSFZ = 4 - MAPS.
ILHS	Int	1	LHS Jacobians. ILHS = 1 - Steger Warming. ILHS = 4 - MAPS.
ILIMITER	Int	1	ILIMITER = 0 - Limiter is disabled. ILIMITER = 1 - Limiter is enabled. ILIMITER = 3 - Limiter is enabled with additional pressure based limiting.
NRK	Int	1	Number of Runge-Kutta (RK) steps (implicit). NRK = 1 - No RK steps. NRK = 3 - 3^{rd} order RK. NRK = 5 - 5^{th} order RK.
IDODDADI	Int	0	Diagonal dominance ADI flag. IDODDADI = 0 - ADI. IDODDADI $\geq$ 1 - Number of DDADI sub-iterations.
NSUBITER	Int	1	Number of sub-iterations in dual time stepping (should be set to 1 except for dual time).

RESINP - Restart Input

Variable Name	Type	Def.	Description
ISTART	Int	0	ISTART = 0 - Starting a new solution from scratch. ISTART = 1 - Restarting a solution (reading all fort.5?? solution files, except for Chimera [fort.59?] files). ISTART = 100 - Restarting a solution while reading also Chimera files (reading all fort.5?? restart files, including fort.59? files). ISTART = 200 - Starting a new solution from scratch but reading Chimera files (fort.59? files).
NSTEPS	Int	1	Number of steps.
ISLOWS	Int	100	If GDTI = GDTF then ISLOWS signifies the number of slow start steps where the wall boundary conditions are gradually introduced into the solution. If ABS(GDTI) < ABS(GDTF) then the CFL number is increased from ABS(GDTI) to ABS(GDTF) within ISLOWS steps.
IRUN	Int	1	IRUN = 1 - Normal run. IRUN = 0 - Conduct hole cutting, interpolation stencil search, and exit; write fort.79? with hole interpolation information, fort.69? (hole files that can be used for restart) and fort.699 for interpolations (also can be used for restart). IRUN = -1 - Conduct hole cutting and exit; write fort.79? with hole information only. IRUN = -2 - Conduct hole cutting and exit; write fort.99? with hole information including hole origin. IRUN = 3 - Dry 6 DOF run; no hole cutting and no flow solution. IRUN = 5 - BC detection run.

OUTINP - Output Frequency Input

Variable Name	Type	Def.	Description
ISTEPL2N	Int	20	L2NORM printout frequency; set to one in debug mode.
ISTEPOUT	Int	20	Output restart files frequency.
ISTEPSUR	Int	-1	Surface output frequency. ISTEPSUR = -1 - No surface files are printed out. ISTEPSUR > 0 - Surface files are printed out every ISTEPSUR steps; the files are named gsurf{step_number}.udp and qsurf{step_number}.udp for the grid and q files, respectively.
ISTEPSOL	Int	-1	Saved solution output frequency. ISTEPSOL = -1 - No solution files are printed out. ISTEPSOL > 0 - Solution files are printed out every ISTEPSOL steps; the files are named gsol{step_number}.udp and qsol{step_number}.udp for the grid and q files, respectively.

FMRINP - Forces and Moments Input

Variable Name	Type	Def.	Description
REFLEN	Real	1.0	Real reference length [meters] (used only to calculate the real time in a normalized 6 DOF run or a normalized elastic run).
GSREF	Real	1.0	Global reference area [$meters^2$] or [dimensionless].
GCREF	Real	1.0	Global reference length [$meters$] or [dimensionless].
GBREF	Real	1.0	Global reference span [$meters$] or [dimensionless].
X0	Real	0.0	Global moment reference point [$meters$] or [dimensionless].
Y0	Real	0.0	Global moment reference point [$meters$] or [dimensionless].
Z0	Real	0.0	Global moment reference point [$meters$] or [dimensionless].
BPFAC	Real	0.0	The back pressure of a cylindrical body is evaluated as BPFAC $\times$ AREA $\times$ PINF [back pressure factor $\times$ area (calculated automatically) $\times$ free stream pressure] and is subtracted from the axial force.

VISINP - Viscosity and Turbulence Input

Variable Name	Type	Def.	Description
IVISG	Int	0	Global viscosity flag. IVISG = 0 - Inviscid flow assumption. IVISG = 1 - Viscous flow. IVISG = 2 - Thin layer approximation.
ITURG	Int	0	Global turbulence flag. ITURG = 0 - Laminar flow assumption. ITURG = 1 - Modified Baldwin-Lomax model. ITURG = 2 - Degani-Schiff model. ~~ITURG = 3 - Goldberg (R_t) model (obsolete, not supported).~~ ITURG = 4 - Spalart-Allmaras model (LGS/DDADI). ~~ITURG = 5 - k-ω-TNT model (ADI, obsolete, cannot be used with Runge Kutta).~~ ITURG = 6 - k-ω-TNT model (LGS/DDADI). ITURG = 7 - k-ω-SST model (LGS/DDADI). ITURG = 11 - RSM-JH model (LGS). ITURG = 12 - RSM SSG/LRR-ω model (LGS/DDADI). ITURG = 13 - RSM MCL model (LGS). ITURG = 14 - RSM GLVY model (LGS/DDADI).
ITRANS	Int	0	ITRANS = 1 toggles transition model γ-Re_{θ_t}-SST (works for ITURG = 7).
RTINIT	Real	3.0	Initial undamped eddy viscosity; used for Goldberg and Spalart-Allmaras.
TUINT	Real	0.001	Turbulence intensity for k-ω models.
VMUETINF	Real	0.01	Free stream μ_t for k-ω models.
GDT_TURBI	Real	GDTI	Initial time step/CFL number for turbulence.
GDT_TURBF	Real	GDTF	Final time step/CFL number for turbulence.

VISINP - Viscosity and Turbulence Input (continued)

Variable Name	Type	Def.	Description
IDES	Int	0	Detached eddy simulation flag. IDES = 0 - RANS. IDES = 1 - XLES. IDES = 2 - DDES. IDES = 3 - X-DDES.
ITORDD	Int	1	Turbulence model spatial accuracy (relevant only for ITURG = 4,6,7,11,12,13,14). ITORDD = 1 - First order (and thin layer for ITURG = 4,6). ITORDD = 2 - Second order.
ITMIMP	Int	1	Turbulence model matrix solver (relevant only for ITURG = 4,6,7,12,14). ITMIMP = 1 - DDADI inversion. ITMIMP = 2 - LGS inversion.

ELAINP - Elastic Input

Variable Name	Type	Def.	Description
IELAST	Int	0	IELAST=0 - Rigid; IELAST=1 - Elastic. If exists, read file *fort.800*, else, spline and write to file *fort.801*. The program looks for the file *genforces.in* which contains the initial/old generalized forces. File *genforces.in* contains one real number per each mode, in sequential lines.; IELAST=2 - Same as IELAST = 1 but with rotated modes (for large deformations); IELAST<0 - Read/spline modes, write deformed mesh, and exit. The program looks for the file *xi.in* which contains the prescribed modal displacements. File *xi.in* has the same format as *genforces.in*. The deformed mesh is written to file *fort.804*.
IDEFMET	Int	1	IDEFMET=1 - Simple shearing; IDEFMET=2 - Boundary Element Method (BEM)
ITER_ELAST_-STAGE	Int	0	Number of flow iterations between consecutive elastic deformations. ITER_ELAST_STAGE = 1 - Dynamic aeroelastic simulation; ITER_ELAST_STAGE > 1 - Static aeroelastic analysis; ITER_ELAST_STAGE = 0 - No elastic deformations.
ELAST_FACT	Real	1.0	Factor multiplying all the eigenvalues. Use with care, at your own risk.
DAMPING	Array	0.0	Structural damping factor for dynamic aeroelasticity (ζ, per mode).

SUBINP - Sub-domain and Flap Input

Variable Name	Type	Def.	Description
NSUBD	Int	0	Number of sub-domains.
NFLAP	Int	0	Number of flaps.

SDFINP - 6 DOF Input

Variable Name	Type	Def.	Description
GFACX	Real	0.0	Load factor.
GFACY	Real	0.0	Load factor.
GFACZ	Real	1.0	Load factor.
NSDFSTAR	Int	0	Starting point for 6dof simulation.
IQUATERNION	Int	1	Quaternion flag. IQUATERNION = 0 - Euler angles. IQUATERNION = 1 - Quaternions.

MLPINP - Multi-Level Parallelism Input

Variable Name	Type	Def.	Description
MLPOUTER	Int	1	Number of groups for multi level parallelism.
MLINNER	Int	1	Number of threads per group.

DEBINP - Debug Input

Variable Name	Type	Def.	Description
DEBUGF	Logical	.FALSE.	Debug flag.
IDEBUGL	Int	0	Debug level.

CONVINP - Convergence Control Input

Variable Name	Type	Def.	Description
RESDROP	Real	3.0	Residual drop cutoff for dual-time stepping. If $RESDROP < 0$, the cutoff is imposed on both the mean flow and turbulence model equation sets. If $RESDROP > 0$, the cutoff is imposed on the mean flow equation set only.
ISIDC	Int	-1	Sum of zone weights for weighted convergence control; Sub-iterations are continued until the cumulative weights of converged zones is above ISIDC (or the maximum number of sub-iterations, NSUBITER, has been reached); If ISIDC is the default then it is set to the total number of zones within the program.

CHIMINP - Chimera Input

Variable Name	Type	Def.	Description
ADOTBEPS	Real	0.02	Hole cutting tolerance.
ADOTBEPS1	Real	0.01	Hole cutting tolerance for caps.
ADOTBEPS2	Real	0.01	Hole cutting tolerance for caps.
TOLNODE	Real	0.505	Node tolerance for regular search.
TOLCBC	Real	0.505	Node tolerance for cell by cell search.
GRMSTOL	Real	2.0	RMS tolerance for cell by cell search.
GFRMSTOL	Real	4.0	RMS tolerance for forced points (FORCING).
K1FMAX	Int	11	K level for forced points.
IS502N2N	Int	0	Node to node flag for fort.502. IS502N2N = 0 - fort.502 is a regular patched grids file. IS502N2N = 1 - fort.502 is a node to node patched grids file.
IDOUBLEF	Int	0	Chimera single/double fringe flag. IDOUBLEF = 0 - Single fringe. IDOUBLEF = 1 - Double fringe.
DELTA_MAX_-DOCELL	Real	5.0	Delta max for DOCELL.
MAXITER_DO-CELL	Int	6	Maximum iterations for DOCELL.

MOTIONINP - Motion Input

Variable Name	Type	Def.	Description
IDONGRID	Int	0	Prescribed motion flag. IDONGRID = 0 - No prescribed motion. IDONGRID = 1 - Prescribed motion; The subroutine ngrid (user defined, see ngrid.f) is invoked with every step.

WDISINP - Wall Distance Solver Input (not used)

Variable Name	Type	Def.	Description
EPSPWD	Real	100.0	Not used.
ITERSWD	Real	10000	Not used.
DTWD	Real	10^{-3}	Not used.

SPLINP - Spline Input

Variable Name	Reps.	Def.	Description
ITRANSALL	Int	0	ITRANSALL=0 - Transformations apply to aerodynamics model only; ITRANSALL=1 - Transformations apply to both aerodynamics as well as structural model.
IS_FSP	Int	0	Force spline. Not used.
IS_TAS	Int	0	IS_TAS=1 - Compute spline transformation matrix TAS. Write it into file *TAS.DAT* ; IS_TAS=0 - Read spline transformation matrix from file *TAS.DAT*
NSSURF	Int	1	Number of structural surfaces used in spline.
ISFLAG	Int	1	ISFLAG=1-All structural grids reside in one file *grid_s_100.dat* and modes file is *modes_s_100.dat*. Files *spl_1xx.dat* define the grid numbers used in each structural surface spline, where xx stands for the surface number. ISFLAG=0 - Structural grids reside in separate files, one per structural surface, named *grid_s_1xx.dat* and modes files are *modes_s_1xx.dat*.
N_MODES	Int	1	Number of structural modes to spline
N_RBMODES	Int	0	Number of rigid body modes in the modes input file that will be removed and not used in the aeroelastic analysis
MFORM	Int	1	Format of modes file. MFORM=1 - Standard NASTRAN punch file. ~~MFORM=2 - ASTROS database ICE file (obsolete).~~ ~~MFORM=3 - ASTROS DMI (obsolete).~~ ~~MFORM=4 - Flap mode (obsolete).~~ MFORM=5 - ZAERO free format.
SCALE	Int	1.0	Factor transferring the length units of the structural model to *meters*. The coordinates of the structural nodes, and the modes, are *divided* by SCALE to transform them to meters.
SCALEM	Int	1.0	Factor transferring the mass units of the structural model to *kg*. The generalized mass is *divided* by (SCALEM*SCALE*SCALE).

SPLINP - Spline Input (continued)

Variable Name	Reps.	Def.	Description
SCALEA	Int	1.0	Factor transferring the length units of the aerodynamic model to *meters*. The coordinates of the aerodynamic grids are *divided* by SCALEA to transform them to meters.
PLOT_FACTOR	Int	1.0	Factor multiplying the modes. For display only.

GSTINP - Gust Input

Variable Name	Reps.	Def.	Description
IGUST	Int	0	IGUST = 0 - No gust input IGUST = 1 - Wagner function IGUST = 2 - Kussner function (sharp-edge gust) IGUST = 3 - One-minus-cosine gust input. $w_g = w_{g_0} \cdot (1 - \cos(2 \cdot \pi \cdot x/GGL)$. Note that the max value this input function takes is $2 \cdot w_{g_0}$ IGUST = 4 - One-cycle sine gust input IGUST = 5 - Prescribed gust input. The excitation at each time step is read from file *wgnf.dat* and multiplied by w_{g_0}. IGUST = 6 - Sinusoidal heave (not gust, all the grid points heave together)
ALP_GUST	Real	0.0	Gust equivalent angle of attack, in degrees. The gust input velocity is $M \cdot \sin w_{g_0}$
ISTEP0G	Int	0	For gust restart run only - Iteration number from which the current gust run is started (number of iterations in the previous gust run)
GGL	Real	0.0	Gust gradient length

TRMINP - Trim Input

Variable Name	Type	Def.	Description
ITER_TRIM_-STAGE	Int	0	Number of elastic iterations after which a trim correction is performed
CL_REQ	Real	0.0	Required trim lift coefficient in a symmetric 2DOF trim analysis
CMY_REQ	Real	0.0	Required trim pitching moment coefficient (about C.G.) in a symmetric 2DOF trim analysis

EXCINP - Prescribed Modal/FLAP Excitation Input

Variable Name	Type	Def.	Description
IEXC	Int	0	IEXC = 0 - No prescribed excitation IEXC = 1 - Modal ramp IEXC = 2 - Modal step IEXC = 3 - Modal sinusoidal IEXC = 4 - Not implemented IEXC = 5 - Modal prescribed excitation. The excitation at each time step is read from file *wgnf.dat* and multiplied by EXC_XIMAX(N). IEXC = 11 - Flap ramp IEXC = 12 - Flap step IEXC = 13 - Flap sinusoidal IEXC = 14 - Not implemented IEXC = 15 - Flap prescribed excitation. The excitation at each time step is read from file *wgnf.dat* and multiplied by XIFLAP(N).
EXC_FREQ	Array	0.0	Excitation frequency for sinusoidal modal/flap excitation, for each mode/flap (for modal excitation repeats N_MODES-N_RBMODES times)
EXC_XIMAX	Array	0.0	Excitation modal amplitude, for each mode (repeats N_MODES-N_RBMODES times). In case of flap excitation EXC_XIMAX is not used. The excitation amplitude is set by parameter XIFLAP in file flap.i
VTIME0	Real	0.0	End time of the analysis from which the current analysis is restarted
VTIME1	Real	0.0	For ramp-type excitation, time of ramp up start
VTIME2	Real	0.0	For ramp-type excitation, time of ramp up end
VTIME3	Real	0.0	For ramp-type excitation, time of ramp down start
VTIME4	Real	0.0	For ramp-type excitation, time of ramp down end

DSCRTFRCINP - Prescribed Discrete Force Input

Variable Name	Type	Def.	Description
NDISCRETEFORCE	Int	0	Number of discrete forces Each force is prescribed through a file named *discreteforcefile_[100 + force number].in* (see Appendix D.11)
DFTIME0	Real	0.0	Time of starting point for discrete force action
I_DF_SUB_DOMAIN	array	-1	Discrete force associated sub-domain
F_DISCRETE_X	array	0.0	X coordinate of force (must be within the bounds as appear in the file *body_extents.dat*)
F_DISCRETE_Y	array	0.0	Y coordinate of force (must be within the bounds as appear in the file *body_extents.dat*)
F_DISCRETE_Z	array	0.0	Z coordinate of force (must be within the bounds as appear in the file *body_extents.dat*)

C.3 Array Input File *eznssa.i*

The following section includes a brief description of the array input file. The EZNSS Glyph – a Tcl/Tk based script – that runs using the Gridgen package, allows to set up the array input file and to generate a Database.

This input file is used for individual zone or sub-domain input. It also serves as a means to override the automatic topology detection as well as to change the default boundary conditions.

IVIS - Zonal viscosity flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - Inviscid 1 - Viscous 1 - Thin layer approximation

IVISBC - Facial viscosity flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Face id (1-6) 1 - ξ_{min} 2 - ξ_{max} 3 - η_{min} 4 - η_{max} 5 - ζ_{min} 6 - ζ_{max}	0/1	0 - Inviscid 1 - Viscous

ITUR - Zonal turbulence flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0-7	N/A	0 - Laminar 1 - Baldwin-Lomax 2 - Degani-Schiff 3 - Goldberg R_t 4 - Spalart-Allmaras 5 - $k-\omega$-TNT (ADI) 6 - $k-\omega$-TNT (LGS) 7 - $k-\omega$-SST

IWING - Zonal wing flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Wing flag	N/A	See Chapter 3 for a full description of possible flags

I2ZWING - Two-zone, H-type topology wing

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Zone id	N/A	Matching zone number

ICUT - Collapsed edge in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No cut boundary conditions for ξ 1 - Cut boundary condition for ξ_{min} 2 - Cut boundary condition for ξ_{max} 12 - Cut boundary condition for ξ_{min} and ξ_{max}

IAX - Axis edge in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No axis boundary conditions for ξ 1 - Axis boundary condition for ξ_{min} 2 - Axis boundary condition for ξ_{max} 12 - Axis boundary condition for ξ_{min} and ξ_{max}

KAX - Axis edge in the ζ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - No axis boundary conditions for ζ 1 - Axis boundary condition for ζ_{min}

JPER - Periodicity in the η coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - Non-periodic 1 - Periodic

ISTAR - Wall starting ξ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	ISTAR	N/A	Wall boundary conditions are applied from this point onward

IEND - Wall ending ξ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	IEND	N/A	Wall boundary conditions are applied up to this point

JTIPL - Wall starting η coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JTIPL	N/A	Wall boundary conditions are applied from this point onward

JTIPR - Wall ending η coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JTIPR	N/A	Wall boundary conditions are applied up to this point

ISTARFM - Starting ξ coordinate for force and moment calculations

1^{st}	2^{nd}	3^{rd}	Description
Zone id	ISTARFM	N/A	Forces and moments are calculated from this point onward

IENDFM - Ending ξ coordinate for force and moment calculations

1^{st}	2^{nd}	3^{rd}	Description
Zone id	IENDFM	N/A	Forces and moments are calculated up to this point

JTIPLFM - Starting η coordinate for force and moment calculations

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JTIPLFM	N/A	Forces and moments are calculated from this point onward

JTIPRFM - Ending η coordinate for force and moment calculations

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JTIPRFM	N/A	Forces and moments are calculated up to this point

KSTAR - Wall starting ζ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	KSTAR	N/A	Wall boundary conditions are applied from this point onward

KEND - Wall ending ζ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	KEND	N/A	Wall boundary conditions are applied up to this point

IOUT - In/out boundary conditions in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No in/out boundary conditions for ξ 1 - Zeroth order extrapolation for ξ_{min} 2 - Zeroth order extrapolation for ξ_{max} 12 - Zeroth order extrapolation for ξ_{min} and ξ_{max}

JOUT - In/out boundary conditions in the η coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No in/out boundary conditions for η 1 - Zeroth order extrapolation for η_{min} 2 - Zeroth order extrapolation for η_{max} 12 - Zeroth order extrapolation for η_{min} and η_{max} 100 - Turkel type boundary conditions for η_{min} and η_{max}

KOUT - In/out boundary conditions in the ζ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No in/out boundary conditions for ζ 1 - Zeroth order extrapolation for ζ_{min} 2 - Zeroth order extrapolation for ζ_{max} 12 - Zeroth order extrapolation for ζ_{min} and ζ_{max} 100 - Turkel type boundary conditions for ζ_{min} and ζ_{max}

IFSBC - Free stream boundary conditions in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No free stream boundary conditions for ξ 1 - Free stream boundary conditions for ξ_{min} 2 - Free stream boundary conditions for ξ_{max} 12 - Free stream boundary conditions for ξ_{min} and ξ_{max}

JFSBC - Free stream boundary conditions in the η coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No free stream boundary conditions for η 1 - Free stream boundary conditions for η_{min} 2 - Free stream boundary conditions for η_{max} 12 - Free stream boundary conditions for η_{min} and η_{max}

KFSBC - Free stream boundary conditions in the ζ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No free stream boundary conditions for ζ 1 - Free stream boundary conditions for ζ_{min} 2 - Free stream boundary conditions for ζ_{max} 12 - Free stream boundary conditions for ζ_{min} and ζ_{max}

IWALL - Wall boundary conditions in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No wall boundary conditions for ξ 1 - Wall boundary conditions for ξ_{min} 2 - Wall boundary conditions for ξ_{max} 12 - Wall boundary conditions for ξ_{min} and ξ_{max}

JWALL - Wall boundary conditions in the η coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No wall boundary conditions for η 1 - Wall boundary conditions for η_{min} 2 - Wall boundary conditions for η_{max} 12 - Wall boundary conditions for η_{min} and η_{max}

KWALL - Wall boundary conditions in the ζ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No wall boundary conditions for ζ 1 - Wall boundary conditions for ζ_{min} 2 - Wall boundary conditions for ζ_{max} 12 - Wall boundary conditions for ζ_{min} and ζ_{max}

ISYM - Symmetry boundary conditions in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No symmetry boundary conditions for ξ 1 - Symmetry boundary conditions for ξ_{min} 2 - Symmetry boundary conditions for ξ_{max} 12 - Symmetry boundary conditions for ξ_{min} and ξ_{max}

JSYM - Symmetry boundary conditions in the η coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No symmetry boundary conditions for η 1 - Symmetry boundary conditions for η_{min} 2 - Symmetry boundary conditions for η_{max} 12 - Symmetry boundary conditions for η_{min} and η_{max}

IJYSYM - Special hole cutting; Invoked ONLY when **ISYM**=0 & **JSYM**=0

1^{st}	2^{nd}	3^{rd}	Description
Zone id	-1/0/1	N/A	0 - No action -1 - All grid points whose Y coordinate is less than zero are blanked (and their neighbors) 1 - All grid points whose Y coordinate is greater than zero are blanked (and their neighbors)

ITHICK1 - Hole cutting lower ξ limit

1^{st}	2^{nd}	3^{rd}	Description
Zone id (hole cutter)	Zone id (subject)	I level	Lower limit for hole cutting; every grid point in the second zone that is found below the I level in the first zone is marked as a hole

ITHICK2 - Hole cutting upper ξ limit

1^{st}	2^{nd}	3^{rd}	Description
Zone id (hole cutter)	Zone id (subject)	I level	Upper limit for hole cutting; every grid point in the second zone that is found above the I level in the first zone is marked as a hole

JTHICK1 - Hole cutting lower η limit

1^{st}	2^{nd}	3^{rd}	Description
Zone id (hole cutter)	Zone id (subject)	J level	Lower limit for hole cutting; every grid point in the second zone that is found below the J level in the first zone is marked as a hole

JTHICK2 - Hole cutting upper η limit

1^{st}	2^{nd}	3^{rd}	Description
Zone id (hole cutter)	Zone id (subject)	J level	Upper limit for hole cutting; every grid point in the second zone that is found above the J level in the first zone is marked as a hole

KTHICK1 - Hole cutting lower ζ limit

1^{st}	2^{nd}	3^{rd}	Description
Zone id (hole cutter)	Zone id (subject)	K level	Lower limit for hole cutting; every grid point in the second zone that is found below the K level in the first zone is marked as a hole

KTHICK2 - Hole cutting upper ζ limit

1^{st}	2^{nd}	3^{rd}	Description
Zone id (hole cutter)	Zone id (subject)	K level	Upper limit for hole cutting; every grid point in the second zone that is found above the K level in the first zone is marked as a hole

KBODY - K level for hole cutting body definition

1^{st}	2^{nd}	3^{rd}	Description
Zone id	K level	N/A	K level that defines the extensions of the body definition for hole cutting

IJET - Jet flag in the ξ direction

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - No jet in the ξ direction 1 - Jet boundary conditions in the ξ direction

JSJET - Starting η coordinate of the jet

1^{st}	2^{nd}	3^{rd}	Description
Zone id	J level	N/A	Jet is implemented starting from this point

JEJET - Ending η coordinate of the jet

1^{st}	2^{nd}	3^{rd}	Description
Zone id	J level	N/A	Jet is implemented up to this point

KSJET - Starting ζ coordinate of the jet

1^{st}	2^{nd}	3^{rd}	Description
Zone id	K level	N/A	Jet is implemented starting from this point

KEJET - Ending ζ coordinate of the jet

1^{st}	2^{nd}	3^{rd}	Description
Zone id	K level	N/A	Jet is implemented up to this point

~~**PJET** - Pressure of the jet~~ - **obsolete (see Appendix D.10 for jet inputs)**

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Pressure	N/A	Pressure of the jet

~~**TJET** - Temperature of the jet~~ - **obsolete (see Appendix D.10 for jet inputs)**

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Temperature	N/A	Temperature of the jet

~~**FJET** - Mass flow rate of the jet~~ - **obsolete (see Appendix D.10 for jet inputs)**

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Pressure	N/A	Mass flow rate of the jet

RTJET - Turbulence intensity factor of the jet

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Jet turbulence intensity factor	N/A	Used for S-A, $k-\omega$, and RSM models

KCAVITY - Cavity boundary conditions in the ζ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - No cavity boundary conditions for ζ 1 - Cavity boundary conditions for ζ_{min}

ISCAVITY - Cavity starting ξ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	ISCAVITY	N/A	Wall boundary conditions are applied from $\xi = 1$ up to $\xi = ISCAVITY$

IECAVITY - Cavity ending ξ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	IECAVITY	N/A	Wall boundary conditions are applied from $\xi = IECAVITY$ up to $\xi = IDIM$

JSCAVITY - Cavity starting η coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JSCAVITY	N/A	Wall boundary conditions are applied from $\eta = 1$ up to $\eta = JSCAVITY$

JECAVITY - Cavity ending η coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JECAVITY	N/A	Wall boundary conditions are applied from $\eta = JECAVITY$ up to $\eta = JDIM$

ISUBDPZ - Primary zone for this zone

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Zone id primary zone	N/A	Currently not utilized

ISUBD - Sub-domain number for this zone

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Sub-domain id	N/A	This zone is a part of this sub-domain

SUBDSREF - Reference area for sub-domain

1^{st}	2^{nd}	3^{rd}	Description
Sub-domain id	Reference area	N/A	Reference area for this sub-domain

SUBDCREF - Reference length for sub-domain

1^{st}	2^{nd}	3^{rd}	Description
Sub-domain id	Reference length	N/A	Reference length for this sub-domain

SUBDX0 - Reference point for moment calculations for sub-domain

1^{st}	2^{nd}	3^{rd}	Description
Sub-domain id	X coordinate	N/A	X coordinate for sub-domain reference point

SUBDY0 - Reference point for moment calculations for sub-domain

1^{st}	2^{nd}	3^{rd}	Description
Sub-domain id	Y coordinate	N/A	Y coordinate for sub-domain reference point

SUBDZ0 - Reference point for moment calculations for sub-domain

1^{st}	2^{nd}	3^{rd}	Description
Sub-domain id	Z coordinate	N/A	Z coordinate for sub-domain reference point

BXFAC - Box factor for body hole cutting

1^{st}	2^{nd}	3^{rd}	Description
Zone id	$BXFAC$	N/A	X coordinate direction factor; Bounding box of the body (for hole cutting purposes only) is multiplied by this factor

BYFAC - Box factor for body hole cutting

1^{st}	2^{nd}	3^{rd}	Description
Zone id	$BYFAC$	N/A	Y coordinate direction factor; Bounding box of the body (for hole cutting purposes only) is multiplied by this factor

BZFAC - Box factor for body hole cutting

1^{st}	2^{nd}	3^{rd}	Description
Zone id	$BZFAC$	N/A	Z coordinate direction factor; Bounding box of the body (for hole cutting purposes only) is multiplied by this factor

IBODY - Assign zone to BODY

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Body id	N/A	Used to assign the body number to the zone; Zones that belong to the same body do not cut holes in each other and cut holes in all other meshes in a different manner

IDOHOLES - Hole cutting flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id (subject)	Zone id (hole cutter)	0/1 100/101 200/201	0 - No holes are cut in this subject by this cutter 1 - Holes are cut in this subject by this cutter 100 - If diagonal is set to 100 then no holes are cut through this zone 101 - If diagonal is set to 101 then this zone does not cut holes in any zone 200 - If diagonal is set to 200 then everything is set to 0 201 - If diagonal is set to 101 then everything is set to 1

IDOFBCS - Face zonal boundary conditions flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id (recipient)	Zone id (donor)	0/1	0 - Recipient cannot receive boundary values from donor 1 - Recipient can receive boundary values from donor; If diagonal is set to 0/1 then all donors receive the same value

ITURK1 - Inflow/Outflow Turkel boundary conditions for ξ_{min}

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/-1	N/A	0 - Nothing 1 - Outflow -1 - Inflow

ITURK2 - Inflow/Outflow Turkel boundary conditions for ξ_{max}

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/-1	N/A	0 - Nothing 1 - Outflow -1 - Inflow

IRIEMANN - Inflow/Outflow Riemann boundary conditions in the ξ direction

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - Nothing 1 - Riemann boundary conditions for ξ_{min} 2 - Riemann boundary conditions for ξ_{max} 2 - Riemann boundary conditions for ξ_{min} and ξ_{max}

JRIEMANN - Inflow/Outflow Riemann boundary conditions in the η direction

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - Nothing 1 - Riemann boundary conditions for η_{min} 2 - Riemann boundary conditions for η_{max} 2 - Riemann boundary conditions for η_{min} and η_{max}

KRIEMANN - Inflow/Outflow Riemann boundary conditions in the ζ direction

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - Nothing 1 - Riemann boundary conditions for ζ_{min} 2 - Riemann boundary conditions for ζ_{max} 2 - Riemann boundary conditions for ζ_{min} and ζ_{max}

IINLET - Inlet boundary condition flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2	N/A	0 - Nothing 1 - Inlet boundary condition for ξ_{min} 2 - Inlet boundary condition for ξ_{max} 101 - Riemann based inlet boundary condition for ξ_{min} 102 - Riemann based inlet boundary condition for ξ_{max}

PEXIT - Pressure value for inlet

1^{st}	2^{nd}	3^{rd}	Description
Zone id	$PEXIT$	N/A	Value of $\frac{p_e}{p_\infty}$. In use by $IINLET$, $ITURK1$, $ITURK2$, and $MASSFLOW$

MASSFLOW - Mass flow rate for inlet (see Appendix D.9)

1^{st}	2^{nd}	3^{rd}	Description
Zone id	$MASSFLOW$	N/A	Mass flow rate

IZOM - Convergence control weighting

1^{st}	2^{nd}	3^{rd}	Description
Zone id	$IZOM$	N/A	Weight of this zone for dual-time convergence control purposes

IRUNFLOW - Flow solution flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - No flow solution for this zone ("frozen flow solution") 1 - Normal flow solution

IREDOHOLES - Redoing holes flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - Do not conduct holes cutting except for the first step 1 - Conduct hole cutting after every mesh change

IREDOZBCS - Redoing zonal boundary conditions flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - Do not search for Chimera interpolation stencils except for the first step 1 - Repeat the search with every mesh change

IDOBOXHOLES - Box hole cutting flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id (subject)	Box id (hole cutter)	0/1	0 - Do not cut holes 1 - Cut holes

IDOANTIHOLES - Anti-hole flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id (subject)	Box id (hole negator)	0/1	0 - Do not negate holes 1 - Negate holes

C.4 Collar Grid Generation Input File *cgg.i*

This section contains the input for collar grid generation using EZNSS. The code can generate a C-type mesh for a C-C type wing or C-H type wing that penetrates a body or an O-type mesh for a C-O type wing that penetrates a body. For the program to invoke the elliptic grid generation, the user must supply a $cgg.i$ file (whose input parameters are described below) and must delete the file $fort.503$.

ISUB1 - Left edge connectivity (ξ_{min} or η_{min})

1^{st}	2^{nd}	Description
Zone id (wing)	Zone id (body)	Generate a collar grid for that specific wing body intersection at ξ_{min} or η_{min} (depending on $JPER$ of the wing)

ISUB2 - Right edge connectivity (ξ_{max} or η_{max})

1^{st}	2^{nd}	Description
Zone id (wing)	Zone id (body)	Generate a collar grid for that specific wing body intersection at ξ_{max} or η_{max} (depending on $JPER$ of the wing)

ICDADD - Added wake points for C type collar

1^{st}	2^{nd}	Description
Zone id (collar)	$ICDADD$	Number of wake points added to the trailing edge intersection point. Ignored when O type collar is generated

JCD - Collar dimensions in the ξ or η coordinate direction

1^{st}	2^{nd}	Description
Zone id (collar)	JCD	Dimension of the collar grid. For O type grid, dimension in the ξ direction, for C type grid, dimension of η direction

KCD - Collar dimensions in the ζ coordinate direction

1^{st}	2^{nd}	Description
Zone id (collar)	KCD	Dimension of the collar grid in the ζ direction

JDTOP - Wing grid line for collar grid edge

1^{st}	2^{nd}	Description
Zone id (collar)	$JDTOP$	For O type mesh, ξ coordinate for collar grid edge, for C type mesh, η coordinate for the edge (based on the wing grid)

KDTOP - Wing grid line for collar grid edge

1^{st}	2^{nd}	Description
Zone id (collar)	$KDTOP$	ζ coordinate for collar grid edge, for C type mesh, (based on the wing grid)

ITEGS - Number of iterations for the elliptic solver

1^{st}	2^{nd}	Description
Zone id (collar)	$ITEGS$	Number of iterations for the elliptic grid solver

OMEGS - First relaxation parameter

1^{st}	2^{nd}	Description
Zone id (collar)	1.0-2.0	Over-relaxation parameter. For coarse meshes use 1.5, for more dense meshes may go closer to 2.0

RELEGS - Second relaxation parameter

1^{st}	2^{nd}	Description
Zone id (collar)	> 0.0	Relaxation parameter. Behaves as inverse time step. Typical values should be around 0.01-0.001. Use higher values for large stretching

C.5 Six Degrees of Freedom Motion Simulation Input File *6dof.i*

The 6 DOF input file that is described below is necessary for 6 DOF motion simulation. Its existence sets the right flags for the simulation. In addition, it is necessary to set the appropriate sub-domain information in the *eznssa.i* file . Note that the number of sub domains is set in the main input file by setting $NSUBD$ to the number of sub domains.

INERTIAM - Sub-domain moment of inertia matrix

1^{st}	2^{nd}	3^{rd}	4^{th}	Description
Row	Column	Sub-domain id	I_{xx} I_{yy} I_{yz} I_{xy} I_{xz} I_{yz}	Enter the inertia moment matrix components of the sub-domains one by one

IDO6DOF - 6 DOF activation flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - Not activated 1 -Activated

TIME6DOF - 6 DOF activation flag by time

1^{st}	2^{nd}	Description
Sub-domain id	TIME6DOF	6 DOF simulation for this sub-domain starts after the time is greater than TIME6DOF

FMASS - Sub-domain mass

1^{st}	2^{nd}	Description
Sub-domain id	Mass	Mass of sub-domain in Newtons

RCREF - Sub-domain reference length

1^{st}	2^{nd}	Description
Sub-domain id	RCREF	Reference length for dimensionless coefficients

RSREF - Sub-domain reference area

1^{st}	2^{nd}	Description
Sub-domain id	RSREF	Reference area for dimensionless coefficients

FEJECTX - Sub-domain ejection force

1^{st}	2^{nd}	Description
Sub-domain id	FEJECTX	Ejection force component (X coordinate direction) for the sub-domain

FEJECTY - Sub-domain ejection force

1^{st}	2^{nd}	Description
Sub-domain id	FEJECTY	Ejection force component (Y coordinate direction) for the sub-domain

FEJECTZ - Sub-domain ejection force

1^{st}	2^{nd}	Description
Sub-domain id	FEJECTZ	Ejection force component (Z coordinate direction) for the sub-domain

FMEJECTX - Sub-domain ejection moment

1^{st}	2^{nd}	Description
Sub-domain id	FMEJECTX	Ejection moment component (X coordinate direction) for the sub-domain

FMEJECTY - Sub-domain ejection moment

1^{st}	2^{nd}	Description
Sub-domain id	FMEJECTY	Ejection moment component (Y coordinate direction) for the sub-domain

FMEJECTZ - Sub-domain ejection moment

1^{st}	2^{nd}	Description
Sub-domain id	FMEJECTZ	Ejection moment component (Z coordinate direction) for the sub-domain

EJECTSTR - Sub-domain ejection stroke

1^{st}	2^{nd}	Description
Sub-domain id	EJECTSTR	Ejection force acts for that length

POLYFORCE_START - Starting point for polynomial forces

1^{st}	2^{nd}	Description
Sub-domain id	POLYFORCE_-START [seconds]	Starting point for prescribed forces and moments through a polynomial. The polynomials are entered through the file *polyforce.in*

POLYFORCE_END - Ending point for polynomial forces

1^{st}	2^{nd}	Description
Sub-domain id	POLYFORCE_-END [seconds]	Ending point for prescribed forces and moments through a polynomial

FORCEFILE_START - Starting point for forces through files

1^{st}	2^{nd}	Description
Sub-domain id	FORCEFILE_-START [seconds]	Starting point for prescribed forces and moments through a series of files. The files are named *forcefile_[100 + sub-domain id].in*

FORCEFILE_END - Ending point for forces through files

1^{st}	2^{nd}	Description
Sub-domain id	FORCEFILE_-END [seconds]	Ending point for prescribed forces and moments through files

FRACFORCE_START - Starting point for fractional forces

1^{st}	2^{nd}	Description
Sub-domain id	FRACFORCE_-START [seconds]	Starting point for prescribed forces and moments through a fraction of the aerodynamics forces and moments. The values are entered through the file *fracforce.in*

FRACFORCE_END - Ending point for fractional forces

1^{st}	2^{nd}	Description
Sub-domain id	FRACFORCE_-END [seconds]	Ending point for prescribed forces and moments through a fraction of the aerodynamics forces and moments

P6DOF - Sub-domain roll rate in body fixed coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	P6DOF	Initial roll rate for this sub-domain

Q6DOF - Sub-domain pitch rate in body fixed coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	Q6DOF	Initial pitch rate for this sub-domain

R6DOF - Sub-domain yaw rate in body fixed coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	R6DOF	Initial yaw rate for this sub-domain

PHID - Sub-domain roll rate in Cartesian coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	PHID	Initial roll rate for this sub-domain

THETAD - Sub-domain pitch rate in Cartesian coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	THETAD	Initial pitch rate for this sub-domain

PSID - Sub-domain yaw rate in Cartesian coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	PSID	Initial yaw rate for this sub-domain

PHI - Sub-domain roll angle in Cartesian coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	PHI	Initial roll angle for this sub-domain

THETA - Sub-domain pitch angle in Cartesian coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	THETA	Initial pitch angle for this sub-domain

PSI - Sub-domain yaw angle in Cartesian coordinate system

1^{st}	2^{nd}	Description
Sub-domain id	PSI	Initial yaw angle for this sub-domain

U6DOF - Sub-domain X direction velocity

1^{st}	2^{nd}	Description
Sub-domain id	U6DOF	Initial velocity component for this sub-domain

V6DOF - Sub-domain Y direction velocity

1^{st}	2^{nd}	Description
Sub-domain id	V6DOF	Initial velocity component for this sub-domain

W6DOF - Sub-domain Z direction velocity

1^{st}	2^{nd}	Description
Sub-domain id	W6DOF	Initial velocity component for this sub-domain

XPOS - Sub-domain location X coordinate

1^{st}	2^{nd}	Description
Sub-domain id	XPOS	Initial location of center of gravity for this sub-domain

YPOS - Sub-domain location Y coordinate

1^{st}	2^{nd}	Description
Sub-domain id	YPOS	Initial location of center of gravity for this sub-domain

ZPOS - Sub-domain location Z coordinate

1^{st}	2^{nd}	Description
Sub-domain id	ZPOS	Initial location of center of gravity for this sub-domain

IXPOSL - X coordinate direction locking flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - Lock motion in X 1 - Perform motion as usual

IYPOSL - Y coordinate direction locking flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - Lock motion in Y 1 - Perform motion as usual

IZPOSL - Z coordinate direction locking flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - Lock motion in Z 1 - Perform motion as usual

IPHIL - Roll angle lock flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - Lock roll motion in body coordinates 1 - Perform motion as usual

ITHETAL - Pitch angle lock flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - Lock roll motion in body coordinates 1 - Perform motion as usual

IPSIL - Yaw angle lock flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - Lock roll motion in body coordinates 1 - Perform motion as usual

IPIVOT - Pivot flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - No pivot 1 - Pivot

PIVOTX - Sub-domain pivot location X coordinate

1^{st}	2^{nd}	Description
Sub-domain id	PIVOTX	X coordinate of pivot point position

PIVOTY - Sub-domain pivot location Y coordinate

1^{st}	2^{nd}	Description
Sub-domain id	PIVOTY	Y coordinate of pivot point position

PIVOTZ - Sub-domain pivot location Z coordinate

1^{st}	2^{nd}	Description
Sub-domain id	PIVOTZ	Z coordinate of pivot point position

IPIVOTANGLE - Pivot angle flag

1^{st}	2^{nd}	Description
Sub-domain id	1/2/3	1 - Motion is allowed in ϕ only 2 - Motion is allowed in θ only 3 - Motion is allowed in ψ only

PIVOTANGLE - Sub-domain pivot angle

1^{st}	2^{nd}	Description
Sub-domain id	PIVOTANGLE	Sub-domain's motion is restricted to the pivot location up to this angle

IDOMINDIST - Minimum distance calculation flag

1^{st}	2^{nd}	Description
Sub-domain id	0/1	0 - No calculation 1 - Calculate minimum distance

C.6 Virtual Body File Inputs *eznssvb.i*

The virtual body input file is used to set and control hole cutting using virtual bodies. All virtual bodies should be placed in the file $fort.507$ using the regular PLOT3D format. Without the existence of the input file all virtual body hole cutting is omitted. Furthermore, all required hole cutting should be set explicitly since all default flags are set to off.

IDOHOLESVB - Virtual body hole cutting flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id (subject)	Zone id (hole cutter)	0/1	0 - No holes are cut in this subject by this cutter 1 - Holes are cut in this subject by this cutter

KTHICKVB - Virtual body hole cutting lower ζ limit

1^{st}	2^{nd}	3^{rd}	Description
Zone id (hole cutter)	Zone id (subject)	K level	Lower limit for hole cutting; every grid point in the second zone that is found below the K level in the first zone is marked as a hole

KBODYVB - Virtual body K level for hole cutting body definition

1^{st}	2^{nd}	3^{rd}	Description
Zone id	K level	N/A	K level that defines the extensions of the virtual body definition for hole cutting

ISUBDVB - Virtual body sub-domain number for this zone

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Sub-domain id	N/A	This zone is a part of this sub-domain

IAXVB - Virtual body axis edge in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No axis boundary conditions for ξ 1 - Axis boundary condition for ξ_{min} 2 - Axis boundary condition for ξ_{max} 12 - Axis boundary condition for ξ_{min} and ξ_{max}

IWINGVB - Virtual body zonal wing flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	Wing flag	N/A	See Chapter 3 for a full description of possible flags

JPERVB - Virtual body periodicity in the η coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1	N/A	0 - Non-periodic 1 - Periodic

ISTARVB - Virtual body wall starting ξ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	ISTARVB	N/A	Wall boundary conditions are applied from this point onward

IENDVB - Virtual body wall ending ξ coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	IENDVB	N/A	Wall boundary conditions are applied up to this point

JTIPLVB - Virtual body wall starting η coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JTIPLVB	N/A	Wall boundary conditions are applied from this point onward

JTIPRVB - Virtual body wall ending η coordinate

1^{st}	2^{nd}	3^{rd}	Description
Zone id	JTIPRVB	N/A	Wall boundary conditions are applied up to this point

ICUTVB - Virtual body collapsed edge in the ξ coordinate direction flag

1^{st}	2^{nd}	3^{rd}	Description
Zone id	0/1/2/12	N/A	0 - No cut boundary conditions for ξ 1 - Cut boundary condition for ξ_{min} 2 - Cut boundary condition for ξ_{max} 12 - Cut boundary condition for ξ_{min} and ξ_{max}

C.7 Elliptic Collar Grid Update File Inputs *ecgu.i*

The elliptic collar grid update file is used for three purposes. The first, it updates the collar grid upon an elastic deformation. The second, it provides a $fort.893$ file to cover the gaps of overlapping holes. And last, it assists in the computation of forces exerted on the gaps.

IMINU - Projection zone for ξ_{min}

1^{st}	2^{nd}	description
Collar zone id	Projection zone id	The ξ_{min} coordinate of the collar zone is projected onto the projection zone as assigned here

JMINU - Projection zone for η_{min}

1^{st}	2^{nd}	description
Collar zone id	Projection zone id	The η_{min} coordinate of the collar zone is projected onto the projection zone as assigned here

JMAXU - Projection zone for η_{max}

1^{st}	2^{nd}	description
Collar zone id	Projection zone id	The η_{max} coordinate of the collar zone is projected onto the projection zone as assigned here

KMINU - Projection zone for ζ_{min}

1^{st}	2^{nd}	description
Collar zone id	Projection zone id	The ζ_{min} coordinate of the collar zone is projected onto the projection zone as assigned here

C.8 Spline Inputs in File *spline.i*

File *spline.i* includes the input data required for mapping the elastic modes from the structural grid in which they are provided to the aerodynamic grid. The required data includes: for each structural surface the type of spline (surface/beam); for each aerodynamic surface the corresponding structural surface for spline; displacements and rotations applied to the aerodynamic surface to align it with the structural surfaces (for spline purposes only), and plot factors for displaying the mapped modes. Each variable has two entries described below.

For cases of constrained deformations, in which subdomains are connected rigidly to a single point and are constrained to move rigidly with that point, the constraint points (one for each such sub-domain) are defined via their x,y, and z coordinates, and by the aerodynamic grid zone to which each sub-domain attaches.

IS_SS - Surface/Beam spline

1^{st}	2^{nd}	description
Structural surface id	1/0	1-Surface spline 0-Beam spline

ISPLINE_FROM - Which structural surface to spline from

1^{st}	2^{nd}	description
Aero surface id	0/id of a structural surface	

DELTAX - Transformation of aero coordinate system, Δx translation

1^{st}	2^{nd}	description
Aero surface id	translation Δx	in length units of the aero grid

DELTAY - Transformation of aero coordinate system, Δy translation

1^{st}	2^{nd}	description
Aero surface id	translation Δy	in length units of the aero grid

DELTAZ - Transformation of aero coordinate system, Δz translation

1^{st}	2^{nd}	description
Aero surface id	translation Δz	in length units of the aero grid

IROT1 - First rotation axis

1^{st}	2^{nd}	description
Aero surface id	First rotation axis	1-x axis; 2-y axis; 3-z axis

IROT2 - Second rotation axis

1^{st}	2^{nd}	description
Aero surface id	Second rotation axis	1-x axis; 2-y axis; 3-z axis

IROT3 - Third rotation axis

1^{st}	2^{nd}	description
Aero surface id	Third rotation axis	1-x axis; 2-y axis; 3-z axis

DELTAPHI - First rotation angle

1^{st}	2^{nd}	description
Aero surface id	First rotation angle	in degrees

Y0PHI - Center of rotation, y coordinate

1^{st}	2^{nd}	description
Aero surface id	Center of rotation, y coordinate	in length units of the aero grid

Z0PHI - Center of rotation, z coordinate

1^{st}	2^{nd}	description
Aero surface id	Center of rotation, z coordinate	in length units of the aero grid

DELTATHETA - Second rotation angle

1^{st}	2^{nd}	description
Aero surface id	Second rotation angle	in degrees

X0THETA - Center of rotation, x coordinate

1^{st}	2^{nd}	description
Aero surface id	Center of rotation, x coordinate	in length units of the aero grid

Z0THETA - Center of rotation, z coordinate

1^{st}	2^{nd}	description
Aero surface id	Center of rotation, z coordinate	in length units of the aero grid

DELTAPSI - Third rotation angle

1^{st}	2^{nd}	description
Aero surface id	Third rotation angle	in degrees

X0PSI - Center of rotation, x coordinate

1^{st}	2^{nd}	description
Aero surface id	Center of rotation, x coordinate	in length units of the aero grid

Y0PSI - Center of rotation, y coordinate

1^{st}	2^{nd}	description
Aero surface id	Center of rotation, y coordinate	in length units of the aero grid

PLOT_FACTORA - Plot factor

1^{st}	2^{nd}	description
Mode id	Factor multiplying the mode for display only	Real

SUBDXC - Constraint point, x coordinate

1^{st}	2^{nd}	description
Sub-domain id	Constraint point location, x coordinate	in length units of the aero grid

SUBDYC - Constraint point, y coordinate

1^{st}	2^{nd}	description
Sub-domain id	Constraint point location, y coordinate	in length units of the aero grid

SUBDZC - Constraint point, z coordinate

1^{st}	2^{nd}	description
Sub-domain id	Constraint point location, z coordinate	in length units of the aero grid

NSUBDC - Aerodynamic zone to which the sub-domain attaches

1^{st}	2^{nd}	description
Sub-domain id	Aerodynamic zone id	

C.9 Flap Inputs in File *flap.i*

File *flap.i* includes the input data required for defining leading and trailing edge control surfaces. The required data includes: Dimensions of a fictitious mesh on which flap deflections are defined; for each flap - which aerodynamic zone it belongs to, is the flap a leading/trailing edge flap, definition of the flap hinge (x,y coordinates of the flap hinge at the root side and at the tip side), flap deflection.

IFLAPDIM - I dimension of flap fictitious grid

1^{st}	2^{nd}	description
Flap id	IDIM	I dimension of a fictitious grid that is used to deflect the flap

JFLAPDIM - J dimension of flap fictitious grid

1^{st}	2^{nd}	description
Flap id	JDIM	J dimension of a fictitious grid that is used to deflect the flap

IFLAP - To which aerodynamic zone the flap belongs

1^{st}	2^{nd}	description
Flap id	Aerodynamic zone id	

IFLAPLT - Leading/trailing edge flap

1^{st}	2^{nd}	description
Flap id	1/-1	1-Trailing-edge flap -1-Leading-edge flap

XFLAPR - x coordinate of the flap hinge at the root side

1^{st}	2^{nd}	description
Flap id	x coordinate	in length units of the aero grid

YFLAPR - y coordinate of the flap hinge at the root side

1^{st}	2^{nd}	description
Flap id	y coordinate	in length units of the aero grid

XFLAPT - x coordinate of the flap hinge at the tip side

1^{st}	2^{nd}	description
Flap id	x coordinate	in length units of the aero grid

YFLAPT - y coordinate of the flap hinge at the tip side

1^{st}	2^{nd}	description
Flap id	y coordinate	in length units of the aero grid

XIFLAP - flap deflection

1^{st}	2^{nd}	description
Flap id	flap deflection	in degrees

C.10 Rotor Inputs in File *rotor.i*

File *rotor.i* includes the input data required for defining a rotor. A special grid should be generated for the rotor. The rotor disk is placed in between $\xi = IROTORDISK$ AND $\xi = IROTORDISK + 1$ and it spans up to $\zeta = KROTORDISK$. The mesh should then be generated accordingly, *i. e.*, coordinate lines in that range should be as straight as possible. The grid should be generated such that the coordinate ξ is pointing in the direction of the flow. A hole must be cut around the rotor area in all meshes except for the rotor mesh. This can be done using a virtual body or box holes.

The mesh can be a mesh without a hub (similar to $KAX = 1$), in which case $IROTOR$ is negative, or with a hub, in which case $IROTOR$ is positive. When the absolute value od $IROTOR$ is set to 3, the rotor local thrust coefficient, $\frac{dC_T}{d\bar{r}}$, local power coefficient, $\frac{dC_P}{d\bar{r}}$, and local radial force coefficient, $\frac{dC_R}{d\bar{r}}$, are entered thorough the file *rotor_blade.in*. The format of the file is as follows: the first line starts with the # sign followed by a space and the number of input entries; the second line starts with the # sign and any header (chosen by the user, may omit the header); the third line and onward contain the input in 4 columns, normalized local radius, local thrust coefficient, local power coefficient, and local radial force coefficient (as in the following example).

```
# 74
# x       dCT_dx       dCP_dx       dCR_dx
0.253       -0.0292      -0.0100      -0.0000
0.263       -0.0293      -0.0102      -0.0000
0.273       -0.0293      -0.0104      -0.0000
   .            .            .            .
   .            .            .            .
   .            .            .            .
0.967       0.2836        0.2158      0.0000
0.977       0.2790        0.2198      0.0000
0.987       0.2625        0.2184      0.0000
```

IROTOR - Rotor flag

1^{st}	2^{nd}	description
Aerodynamic zone id	-3, -2, -1, 0, 1, 2, 3	0 - No rotor ± 1 - Simple disk model (uniform pressure jump) ± 2 - Variable pressure jump (maximum at 3/4 disk radius ± 3 - User defined through input file (*rotor_blade.in*)

IROTORDISK - ξ coordinate location of the rotor disk

1^{st}	2^{nd}	description
Aerodynamic zone id	ξ coordinate	The rotor is placed between the coordinates ξ and $\xi + 1$

KROTORDISK - ζ coordinate location of the rotor disk edge

1^{st}	2^{nd}	description
Aerodynamic zone id	ζ coordinate	ζ coordinate of the disk edge

ROTORRADIUS - The radius of the rotor disk

1^{st}	2^{nd}	description
Aerodynamic zone id	Rotor disk radius	

ROTORRPM - The RPM of the rotor

1^{st}	2^{nd}	description
Aerodynamic zone id	RPM	

ROTORTHRUST - The thrust of the rotor

1^{st}	2^{nd}	description
Aerodynamic zone id	Rotor thrust	Total rotor thrust. Distributed via two linear functions, peeking at mid radius.

ROTORPOWER - The power of the rotor

1^{st}	2^{nd}	description
Aerodynamic zone id	Rotor power	

ROTORFR - Rotor disk radial force

1^{st}	2^{nd}	description
Aerodynamic zone id	Rotor disk radial force	

Appendix D

Data Files

D.1 Rationale

There are two main types of files, numbered files based on specific series and files that are identified by names. Numbered data files are numbered according to the following rationale. Single and double digit files contain global time history information. Triple digit files are grid, solution, and other input/output files. Quadruple digit files contain time history of the results of six degrees of freedom simulation and time history of forces and moments.

D.2 Input Data Files

Grid files are named $fort.501$ - $fort.505$. The file $fort.501$ contains Chimera grids, the file $fort.502$ contains patched grids, the file $fort.503$ contains elliptic collar grids, the file $fort.504$ contains hyperbolic collar grids, and the file $fort.505$ contains Cartesian grids. Another grid file, the virtual body grid file (named $fort.507$), is used to define bodies for hole cutting. These bodies are not involved in the actual flow calculations and their use is limited for the process of hole cutting only. Solution files are named $fort.511$ - $fort.515$ where $fort.511$ is the solution file for grid file $fort.501$ and so forth. Turbulent viscosity files are named $fort.521$ - $fort.525$, R_t files are named $fort.531$ - $fort.535$ and γ files are named $fort.541$ - $fort.545$. The files

$fort.551$ - $fort.555$ and $fort.561$ - $fort.565$ contain restart grid metrics for moving grid computations. The files $fort.571$ - $fort.575$ contain wall distances.

In addition to the above files, there are two input files for static and dynamic aeroelastic flow simulations. The file $fort.800$ contains the modes and eigenvalues, whereas the file $fort.808$ contains initial modes and general forces.

D.3 Output Data Files

All files from the $fort.500$ series have corresponding output files that are names $fort.600$ having exactly the last two digits. For example, the output γ file for an elliptic collar grid would be names $fort.643$. When holes are generated a new series of files is generated and is named $fort.791$, $fort.792$, and/or $fort.795$, depending on the the corresponding grid file. The output file for aeroelastic modes and generalized forces is named $fort.818$ (or $fort.817$ in case of a stop $MONITOR$ command; in that case the grid and solution files are named $fort.26$ and $fort.27$, respectively).

Convergence history is written into the files $fort.7$ (time vs. means square) and $fort.11$ (iteration number vs. root mean square) and forces and moments time history are written into the file $fort.17$. The format of the files $fort.7$ and $fort.11$ is the time (or iteration number for $fort.11$) and $L2NORM$ (or $\sqrt{L2NORM}$) for the file $fort.11$) for all zones. The file $fort.17$ contains the time and then C_X, C_Y, C_Z, C_MX, C_MY, C_MZ, C_L, and C_D. Note, forces and moments in the $fort.17$ file are sums of **all** forces and moments in **all** the zones.

The quadruple digit file series that correspond to sub-domains is numbered by sub-domains (last 1, 2, or 3 digits). The $fort.1000$ series contain trajectory time history, the $fort.2000$ series contain velocity time history, the $fort.3000$ series contain the acceleration time history. The $fort.8000$ series contains forces and moments in Cartesian coordinates while the $fort.9000$ series contains forces and moments in body coordinates.

In addition, the $fort.7000$ file series contains time history for each zone, in a Cartesian coordinate system. The $fort.804$ file series contains modal deflections in forms of surface grid files (this happens only when the input parameter $ITER\_ELAT\_STAGE$

parameter is negative; the program stops after that). The $fort.5000$ series contains surface collar grids for the possible generation of hyperbolic collar grids.

Convergence of sub-iterations in dual-time-stepping mode is reported in the $fort.6000$ file series. Each line of the file contains 8 columns. The first column is the time, the second column is the number of sub-iteration that has been conducted, the next pair of columns contain the residual drop of the mean flow equations with respect to the maximum residual and the residual in the first sub-iteration, respectively. The next pair contains the same information for the turbulence model equation(s). The last pair is kept for historical reasons.

D.4 Files with Specific Designation

There are four diagnostic files named *eznss.out*, *eznss.err*, *eznss.wrn*, and *eznss.dgn*. The six degrees of freedom simulation may require a restart file and this one is *6dofrest.in* on input and *6dofrest.out* on output. The grid extents and the body extents ($\zeta = 1$) may be founds in the files *extents.dat*, *body_extents.dat*, *vb_extents.dat*, and *vb_body_extents.dat*. Jacobian and time step information is found in the file *jacobian_cfl.dat*. The information is updated every $ISTEPOUT$ steps or every step when the $DEBUG$ flag is set to $.TRUE.$ and $IDEBUGL$ is set to a value greater or equal to 1.

D.5 Aeroelasticity Input Output Files

D.5.1 Input Files

Input files for spline:

1. Splined modes:
   fort.800 - Modes splined to the aerodynamic grids. PLOT3D, unformatted, DP. If this file exists, the code reads the modes from it, and does not compute the spline. If the file does not exist the code computes the spline, using the following files:

2. Spline related parameters:
   spline.i. See description of entries in section C.8.

3. Structural grids:
   There are two methods of reading the structural grid and modes. Switching between them is done by flag ISFLAG in the main input file *eznss.i.defaults*.

   - ISFLAG = 0 - Reading grids from separate files for each structural components. In this case all grid points are included in the spline.
   - ISFLAG = 1 All grids are in a single file and all modal displacements are in a single file, and there are separate files defining which grids are used in spline of which surface.

   The structural grid files are:
   grid_s_100.dat (old fort.21) for ISFLAG = 1; or
   grid_s_101.dat, grid_s_102.dat, etc. (old fort.21 fort.22 etc.) for ISFLAG = 0.
   File format is NASTRAN GRID format.

4. Modes:
   mode_s_100.dat (old fort.61) for ISFLAG = 1; or
   mode_s_101.dat, mode_s_102.dat, etc. (old fort.61 fort.62 etc.) for ISFLAG = 0.
   File formats can be:

   - Nastran .pch file for MFORM = 1
   - Astros ICE for MFORM = 2. This option is not maintained.
   - Nastran DMI for MFORM = 3. This option is not maintained.
   - Flap mode for MFORM = 4
   - ZAERO free format for MFORM = 5

5. Nodes participating in spline:
   spl_101.dat, spl_102.dat, etc. (old fort.31 fort.32 etc.) for ISFLAG = 1 only.

6. *xi.in* - Prescribed modal displacements for IELAST= -1 case.

7. Flap related parameters:
   flap.i. See description of entries in section C.9.

Other input files for AE analyses:

1. *genforces.in* - Initial generalized forces, modal displacements, and modal velocities (old fort.808). ASCII, list. An entry (in a separate line) for each mode.

D.5.2 Output Files

Output files from the spline computation:

1. *mode_a_101.dat, mode_a_102.dat, etc.* - mode 1,2, etc. mapped to the aerodynamic surface grid and plotted on top of the surface grid. For plotting purposes. ASCII, PLOT3D format.

2. *mode_d_101.dat, mode_d_102.dat, etc.* - mode 1,2, etc. mapped to the aerodynamic surface grid, displacements only. For plotting purposes. ASCII, PLOT3D format.

3. *fort.801* - Modes splined to the aerodynamic grids. PLOT3D, unformatted, DP. Generated by the spline routine. Can be moved into file *fort.800* to be read by the code for further analyses (to avoid recalculating the spline).

4. *struct_100.dat* - Structural nodes from file *grid_s_100.dat* in PLOT3D, ASCII, format. Used to check the structural input.

5. *TAS.DAT* - Transformation matrix. Once computed, the transformation matrix can be used in further analyses of the same model (avoiding recalculating it), by setting the ITAS = 0 flag in the main input file.

Other aeroelastic output files:

1. *xi.hist* - History of generalized displacements. One column per mode.

2. *genforces.hist* - History of generalized forces. One column per mode.

3. *flap_a_101.dat, flap_a_102.dat, etc.* - For cases of deflected flaps only - flap 1,2, etc. mapped to the aerodynamic surface grid and plotted on top of the surface grid. For plotting purposes. ASCII, PLOT3D format.

4. *flap_d_101.dat, flap_d_102.dat, etc.* - For cases of deflected flaps only - flap 1,2, etc. mapped to the aerodynamic surface grid, displacements only. For plotting purposes. ASCII, PLOT3D format.

D.6 Six Degrees of Freedom Motion Simulation Input Files

The 6 dof suite provides the capability to prescribe forces and moments through input files. There are three types of capabilities and associated files:

1. Forces and moments are 5^{th} polynomial functions of times. The file is named *polyforce.in* (see Figure D.1).

2. Forces and moments are explicitly prescribed through a series of input files. The files are named *forcefile_[100 + sub-domain id].in* (see Figure D.2).

3. Forces and moments are set as a fraction of the aerodynamic forces. The files is named *fracforce.in* (see Figure D.3).

```
# Polynomial input file format:
# A line starting with the symbol '#' signifies a comment
# You may have as many comments as you'd like
# You must have six lines of input per sub domain,
# each having 6 real number coefficients
# For example:
# 1st sub-domain
# FX coefficients
# a0 a1 a2 a3 a4 a5
0.0 0.0 0.0 0.0 0.0 0.0
# FY coefficients
# a0 a1 a2 a3 a4 a5
0.0 0.0 0.0 0.0 0.0 0.0
# FZ coefficients
# a0 a1 a2 a3 a4 a5
0.0 0.0 0.0 0.0 0.0 0.0
# MX coefficients
# a0 a1 a2 a3 a4 a5
0.0 0.0 0.0 0.0 0.0 0.0
# MY coefficients
# a0 a1 a2 a3 a4 a5
0.0 0.0 0.0 0.0 0.0 0.0
# MZ coefficients
# a0 a1 a2 a3 a4 a5
0.0 0.0 0.0 0.0 0.0 0.0
```

Figure D.1: Polynomial input file format (*polyforce.in*)

```
# Force input file naming convention :
# forcefile_(100 + sub_domain_number).in
# Force input file format :
# A line starting with the symbol '#' signifies a comment
# You may have as many comments as you'd like
# You must have at least 2 lines of input,
# each having 7 real number coefficients
# For example:
# time fx fy fz mx my mz
0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

Figure D.2: Force file input file format (*forcefile_101.in*)

```
# Fractional force input file format:
# A line starting with the symbol '#' signifies a comment
# You may have as many comments as you'd like
# You must have one line of input per sub domain,
# each having 6 real number coefficients
# For example:
# 1st sub-domain
# Fx Fy Fz Mx My Mz
0.0 0.0 0.0 0.0 0.0 0.0
```

Figure D.3: Fractional input file format (*fracforce.in*)

D.7 Non-Standard Atmosphere Input File

EZNSS calculates the thermodynamic flow conditions based on the entered altitude and standard atmosphere tables. Non standard atmosphere conditions may be entered through the file *non_standard_atmosphere.dat* file. The input parameters are as follows (note that no defaults are allowed for $RTINF$ and $RPINF$ and that if C_{p_∞} is set to any negative number it will be calculated based on Equation 2.31 with the parameters as described in Equations 2.32 and 2.33.):

NSAINP - Non Standard Atmosphere Input

Variable Name	Type	Def.	Description
RPINF	Real	None allowed	Pressure
RTINF	Real	None allowed	Temperature
R_GAS	Real	287.0	Specific gas constant
VMUE_C1	Real	1.458E-6	Viscosity Sutherland law coefficient
VMUE_C2	Real	110.3	Viscosity Sutherland law coefficient
HCKAP_C1	Real	2.495E-3	Heat conduction law coefficient
HCKAP_C2	Real	194.0	Heat conduction law coefficient
CPINF	Real	-1.0	Free stream specific heat (C_{p_∞})

D.8 Specific Heat Polynomial Coefficients Input File

The polynomial coefficients that are used to evaluate the specific heats are based on the assumption that the fluid is air. By utilizing the non-standard atmosphere input file and the file named *cp_user_input.dat*, one can enter the coefficients for another perfect gas other than air. The description of the input variables is as follows:

CPINP - C_p Polynomial Input

Variable Name	Type	Def.	Description
A1A	Real	3.08792717E+00	Coefficients
A2A	Real	1.24597184E-03	Coefficients
A3A	Real	- 4.23718945E-07	Coefficients
A4A	Real	6.74774789E-11	Coefficients
A5A	Real	- 3.97076972E-15	Coefficients
A1B	Real	3.56839620E+00	Coefficients
A2B	Real	- 6.78729429E-04	Coefficients
A3B	Real	1.55371476E-06	Coefficients
A4B	Real	- 3.29937060E-12	Coefficients
A5B	Real	- 4.66395387E-13	Coefficients

Note that $A1A$, $A2A$, $A3A$, $A4A$, and $A5A$ pertain to the polynomial below a temperature of $T = 1000K$ while $A1B$, $A2B$, $A3B$, $A4B$, and $A5B$ to the polynomial above that temperature. The user is referred to Section 2.2.6 for the description of the polynomial and the manner in which real gas effects are implemented.

D.9 Inlet Mass Flow Rate Control Input File

The feature of mass flow rate control through an inlet is administered through the $BCIINLET$ flag, the $MASSFLOW$ flag, and the $PEXIT$ flag (see Appendix C.3 for details). The dedicated input file is named *massflow.in*. The value of $PEXIT$ is used as an initial guess and is iterated. The resulting $PEXIT$ is output to the file *pexit.out*. Upon restart this file should be moved to *pexit.in*. The file contains $PEXIT$ for all the zones (including ones without a prescribed $MASSFLOW$ rate).

MASSINP - Mass Flow Rate Input

Variable Name	Type	Def.	Description
ISTARTMASS	Integer	10000	Starting step for mass flow control iterations.
ISTEPMASS	Integer	1000	Frequency of mass flow corrections.
FACTMASSINC	Real	0.1	Mass flow correction factor: $\Delta P_{exit} = \Delta\bar{\rho u} \times (\rho u)_{\infty} \times FACTMASSINC$

D.10 Jet Input Files

Implementing a jet requires a dedicated zone for the jet. More specifically, the dedicated zone must be circular with a KAX boundary condition set to $KAX = 1$ (should be automatically identified but can be manually set by the array file, see Appendix C.3 for more details).The jet may be applied at ξ_{min} through a series of input files whose description follows. The part of ξ_{min} that is not covered through the jet may be set as a wall by using the $IWALL$ boundary condition simultaneously. The jet conditions are given as a function of r, the distance of each point from the jet center. The extents of the jet in terms of η and ζ are given by the input arrays $JSJET$, $JEJET$, $KSJET$, $KEJET$ (see Appendix C.3 for details).

The jet conditions are supplied through the files: *density.dat*, *press.dat*, *uVel.dat*, and *vVel.dat*. Each file contains the dimensional variables at the jet exit as a function of the distance from the jet center. The jet center is located at the axis of grid, *i.e*, $r\,(\zeta = 1) = 0$. The two velocity components are the axial (in the file *uVel.dat*) and radial (in the file *vVel.dat*) components. Both velocity components refer to the grid coordinates. In other words, the velocity component in the file *uVel.dat* is the velocity in the ξ direction. An example for the files structure is given in Figure D.4. In this example, the density at the jet exit is set to be constant, $\rho_{jet} = 1.225$, in the range $0.0 < r < 0.03$ and linearly changes between $\rho_{jet} = 1.225$ and $\rho_{jet} = 1.115$ in the range $0.03 < r < 0.06$. The header of the file contains the number of input lines in the file (not counting the header).

```
3
0.0 1.225
0.03 1.225
0.06 1.115
```

Figure D.4: An example of a typical input file (*density.dat*)

D.11 Discrete Force Input Files

Discrete forces acting on certain surface panels can be modeled using the discrete force feature. The location of the force is prescribed in the main input file. The actual force itself is prescribed a series of input files named: *discreteforcefile_[100 + force number].in*.

```
# Point force input file naming convention :
# pointforcefile_(100 + point_force_file_number).in
# Point force input file format :
# A line starting with the symbol '#' signifies a comment
# You may have as many comments as you'd like
# You must have at least 2 lines of input each having 2 real numbers.
# One for the time and second for the discrete force
# For example:
# time discrete_force
0.0 0.0
0.0 0.0
```

Figure D.5: Discrete force input file format (*discreteforcefile_[100 + force number].in*)

Each of the discrete forces may be associated to a sub-domain using the array $I\_DF\_SUB\_DOMAIN$ (see description of the main input file in Section C.2). If the discrete force is associated to a sub-domain it acts us an injection force (and moment when applicable). It preempts the ejection force moment and stroke that are prescribed in the *6dof.i* file

D.12 Sectional Force and Moment Output

Forces and moment calculation for each zone are reported in the file series *fort.7???* and each sub-domain forces and moments are reported in the file series *fort.8???* or *fort.9???* (in body coordinates). In addition, the user may supply a list of sections for which forces and moments are calculated. This is prescribed in the file *surface_force_list.dat*. The structure of the file is as follows:

```
1
1 1 1 1 1 1 1
```

Figure D.6: File format for (*surface_force_list.dat*)

The first line signifies the number of entries. Each entry contains seven integer values signifying the zone number of the section followed by ξ_{min}, ξ_{max}, η_{min}, η_{max}, ζ_{min}, ζ_{max}. The results are reported in a list of files, one per entry, with the naming convention: *surface_force_[100+entry number].dat* for the forces and moments and *surface_force_coeffs_[100+entry number].dat* for the coefficients. Each line contains the step number, the time, and six values for the forces and moments or the coefficients thereof.

Appendix E

Test Cases

E.1 Flow Solver

E.1.1 RAE 2822 Super Critical Airfoil

The RAE2822 supercritical airfoil is a well known test case. It is used for the validation of flow solvers with respect to transonic turbulent flows. It has been tested in the RAE wind tunnel in 11 different flow conditions at Mach numbers ranging from $M_\infty = 0.676$ to $M_\infty = 0.75$, and at several Reynolds numbers [44]. Out of the 11 experiments flow condition examined, case 9 has been chosen here as the benchmark case. The experimental flow conditions were set at Mach number of $M_\infty = 0.73$, angle of attack of $\alpha = 3.19°$, and a Reynolds number of $Re = 6.5 \times 10^6$. To compare the experimental data with a flow around an airfoil in free-flight conditions, corrections to the wind tunnel data are required. Various wind tunnel corrections have been suggested in the literature. The correction used in EUROVAL project [45] is adopted. These corrections correspond to the following flow conditions: Mach number of $M_\infty = 0.734$, angle of attack of $\alpha = 2.54°$, and a Reynolds number of $Re = 6.5 \times 10^6$.Note that in the experiments, transiton has been tripped near the leading edge of the airfoil at x/c = 0.03 on both upper and lower surfaces of the airfoil.

Figure E.1 shows a close up of a C type mesh that was generated for the purpose

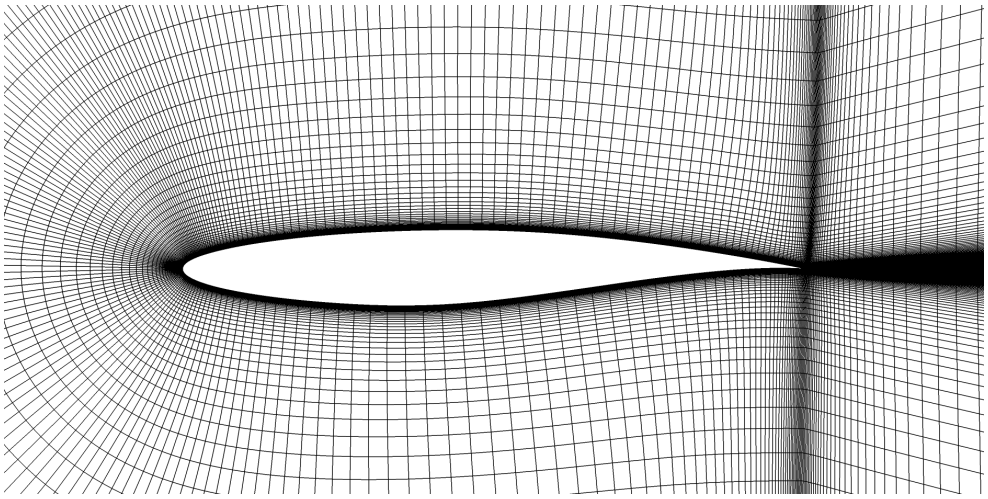

Figure E.1: RAE 2822 computational mesh

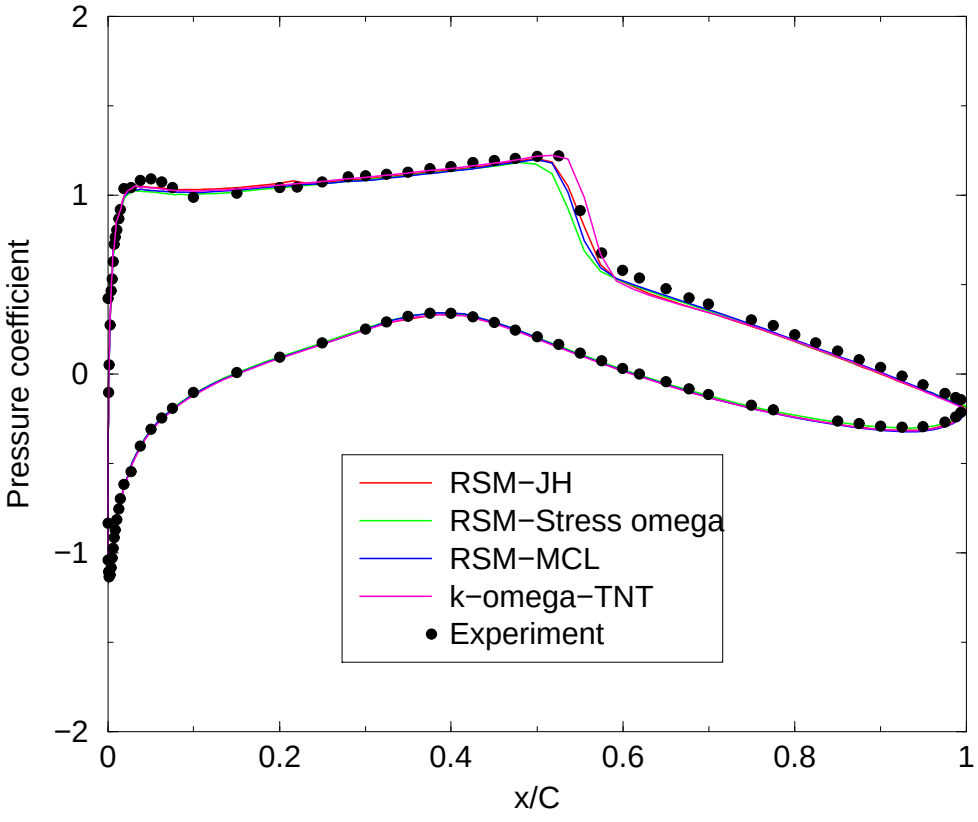

Figure E.2: Pressure coefficient

of flow simulations. Various turbulence models have been examined and the results presented herein include results from 3 Reynolds stress models and the $k-\omega$-TNT RANS model. Riemann type boundary conditions are set at all boundaries, except for the wall and the cut. Figure E.2 shows a comparison of the computed pressure coefficient with the experimental one. All models exhibit excellent agreement with the experiment.

The following figures (Figures E.3-E.6) contain excerpts from the input files that are used for this simulation. The figures describe the input entries that are related to the flow conditions, time step, method, restart information, and turbulence model inputs. Note that one of the figures is repeated, once for the $k-\omega$-TNT model and once for the RSM-MCL model.

```
$FLOINP
 ALT = 0.0 ! Default is 0.0 !
 ALP = 2.54 ! Default is 0.0 !
 BET = 0.0 ! Default is 0.0 !
 FSMACH = 0.734 ! Default is 0.84 !
 REY = 6500000.0 ! Default is 1000000 !
 PR = 0.92 ! Default is 0.7 !
 GAMMA = 1.4 ! Default is 1.4 !
 IGAMMAF = 0 ! Default is 0 !
 IPERFLOW = 0 ! Default is 0 !
 ILOWMACH = 0 ! Default is 0 !
$END
```

Figure E.3: Flow conditions for the RAE 2822 case 9

```
$TIMINP
 GDTI = -1.0 ! Default is -1.0 !
 GDTF = -50.0 ! Default is -1.0 !
 FSA = 0.5 ! Default is 0.5 !
 DDT = 0.0 ! Default is 0.0 !
 H = 1.0 ! Default is 1.0 !
$END

$METINP
 IMET = 1 ! Default is 1 !
 IFSFX = 2 ! Default is 3 !
 IFSFY = 2 ! Default is 3 !
 IFSFZ = 2 ! Default is 3 !
 ILIMITER = 1 ! Default is 1 !
 NRK = 1 ! Default is 1 !
 IDODDADI = 0 ! Default is 0 !
 NSUBITER = 1 ! Default is 1 !
$END

$RESINP
 ISTART = 0 ! Default is 0 !
 NSTEPS = 5000 ! Default is  1 !
 ISLOWS = 50 ! Default is  30 !
 IRUN = 1 ! Default is 1 !
$END
```

Figure E.4: Time step, method, and restart info for the RAE 2822 case 9

```
$VISINP
 IVISG = 1 ! Default is 0 !
 ITURG = 6 ! Default is 0 !
 ITRANS = 0 ! Default is 0 !
 RTINIT = 1.0 ! Default is 1.0 !
 TUINT = 0.001 ! Default is 0.001 !
 VMUETINF = 0.01 ! Default is 0.01 !
 GDT_TURBI = 1.0 ! Default is GDTI !
 GDT_TURBF = 2200.0 ! Default is GDTF !
 IDES = 0 ! Default is 0 !
 ITORDD = 1 ! Default is 1!
 ITMIMP = 1 ! Default is 1!
$END
```

Figure E.5: Turbulence model input ($k-\omega$-TNT) for the RAE 2822 case 9

```
$VISINP
 IVISG = 1 ! Default is 0 !
 ITURG = 13 ! Default is 0 !
 ITRANS = 0 ! Default is 0 !
 RTINIT = 1.0 ! Default is 1.0 !
 TUINT = 0.001 ! Default is 0.001 !
 VMUETINF = 0.1 ! Default is 0.01 !
 GDT_TURBI = 0.1 ! Default is GDTI !
 GDT_TURBF = 40.0 ! Default is GDTF !
 IDES = 0 ! Default is 0 !
 ITORDD = 2 ! Default is 1!
 ITMIMP = 1 ! Default is 1!
$END
```

Figure E.6: Turbulence model input (RSM-MCL) for the RAE 2822 case 9

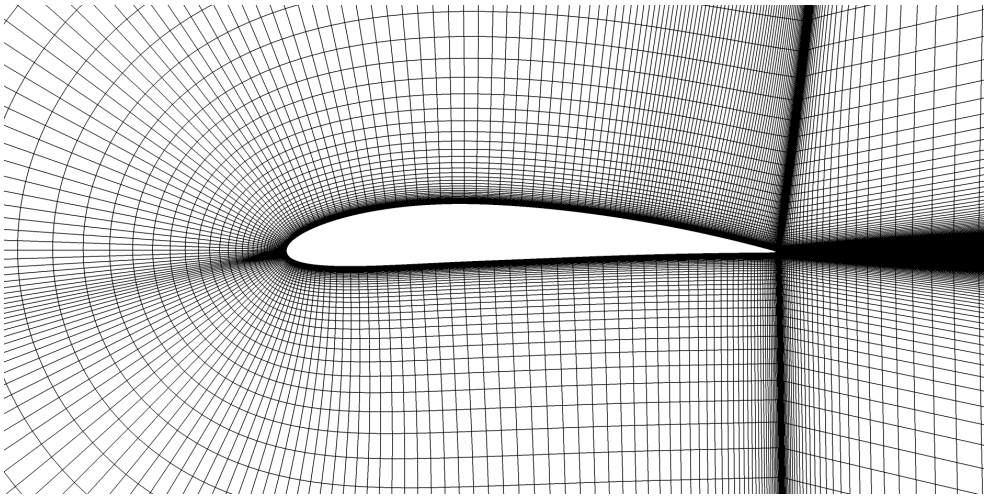

Figure E.7: NACA 4412 computational mesh

E.1.2 NACA 4412 Airfoil

The NACA 4412 test is a well known test case for high lift separated flows. The flow conditions in this case are an angle of attack of $\alpha = 13.87°$, a Reynolds number of $Re_\infty = 1.52 \times 10^6$, and a free stream Mach number of $M_\infty = 0.2$. At this incidence, a steady trailing-edge separation is present. Figure E.7 shows a close up of the computational mesh. This example has been run using two inviscid flux methods, the HLLC flux difference splitting (FDS) and the Steger-Warming flux vector splitting (FVS). A comparison of the calculated stream-wise velocity at the station $\frac{x}{C} = 0.953$ is given in Figure E.8. An interesting result is that Stege-Warming FVS provides a better comparison with the experiment than HLLC FDS. Figure E.9 describes the typical convergence for such a case. Note that proper separation and reversed flow are obtained only after the flow solution has fully converged. Figures E.10-E.15 contain excerpts from the input files that are used for this simulation. The only difference between the HLLC run and the Steger-Warming run is in the choice fo flux evaluation as may bee seen in Figures E.12 (HLLC) and E.13 (Steger-Warming).

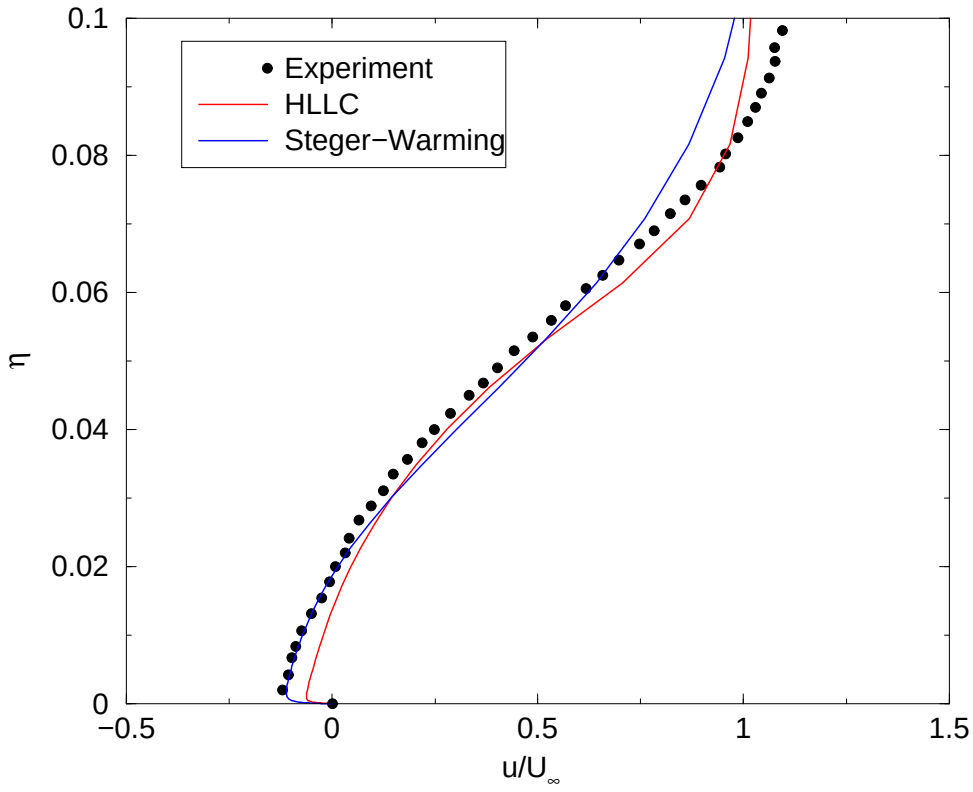

Figure E.8: Stream-wise velocity at the station $\frac{x}{C} = 0.953$

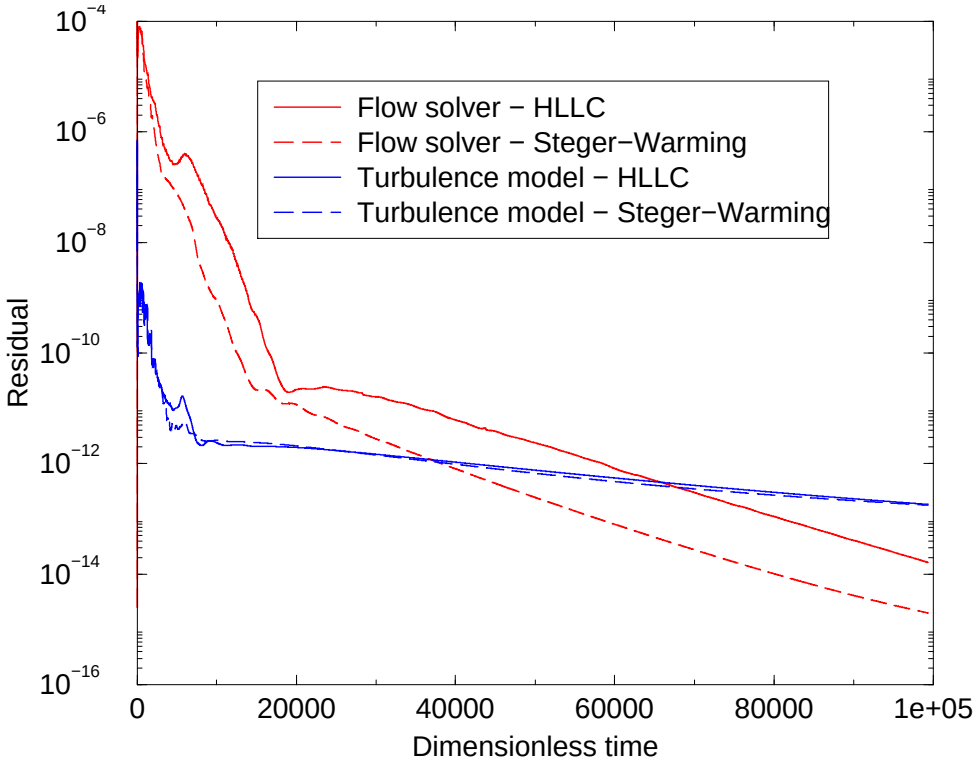

Figure E.9: Convergence time history

```
$FLOINP
 ALT = 0.0 ! Default is 0.0 !
 ALP = 13.87 ! Default is 0.0 !
 BET = 0.0 ! Default is 0.0 !
 FSMACH = 0.2 ! Default is 0.84 !
 REY = 1520000.0 ! Default is 1000000 !
 PR = 0.92 ! Default is 0.7 !
 GAMMA = 1.4 ! Default is 1.4 !
 IGAMMAF = 0 ! Default is 0 !
 IPERFLOW = 0 ! Default is 0 !
 ILOWMACH = 0 ! Default is 0 !
$END
```

Figure E.10: Flow conditions for the NACA 4412 case

```
$TIMINP
 GDTI = -1.0 ! Default is -1.0 !
 GDTF = -10.0 ! Default is -1.0 !
 FSA = 0.5 ! Default is 0.5 !
 DDT = 0.0 ! Default is 0.0 !
 H = 1.0 ! Default is 1.0 !
$END
```

Figure E.11: Time step and spatial accuracy for the NACA 4412 case

```
$METINP
 IMET = 1 ! Default is 1 !
 IFSFX = 2 ! Default is 3 !
 IFSFY = 2 ! Default is 3 !
 IFSFZ = 2 ! Default is 3 !
 ILIMITER = 1 ! Default is 1 !
 NRK = 1 ! Default is 1 !
 IDODDADI = 0 ! Default is 0 !
 NSUBITER = 1 ! Default is 1 !
$END
```

Figure E.12: Method for the NACA 4412 case (HLLC)

```
$METINP
 IMET = 1 ! Default is 1 !
 IFSFX = 1 ! Default is 3 !
 IFSFY = 1 ! Default is 3 !
 IFSFZ = 1 ! Default is 3 !
 ILIMITER = 1 ! Default is 1 !
 NRK = 1 ! Default is 1 !
 IDODDADI = 0 ! Default is 0 !
 NSUBITER = 1 ! Default is 1 !
$END
```

Figure E.13: Method for the NACA 4412 case (Steger-Warming

```
$RESINP
 ISTART = 0 ! Default is 0 !
 NSTEPS = 10000 ! Default is  1 !
 ISLOWS = 100 ! Default is  100 !
 IRUN = 1 ! Default is 1 !
$END
```

Figure E.14: Restart info for the NACA 4412 case

```
$VISINP
 IVISG = 1 ! Default is 0 !
 ITURG = 6 ! Default is 0 !
 ITRANS = 0 ! Default is 0 !
 RTINIT = 1.0 ! Default is 1.0 !
 TUINT = 0.001 ! Default is 0.001 !
 VMUETINF = 0.01 ! Default is 0.01 !
 GDT_TURBI = 1.0 ! Default is GDTI !
 GDT_TURBF = 2200.0 ! Default is GDTF !
 IDES = 0 ! Default is 0 !
 ITORDD = 1 ! Default is 1!
 ITMIMP = 1 ! Default is 1!
$END
```

Figure E.15: Turbulence model input for the NACA 4412 case

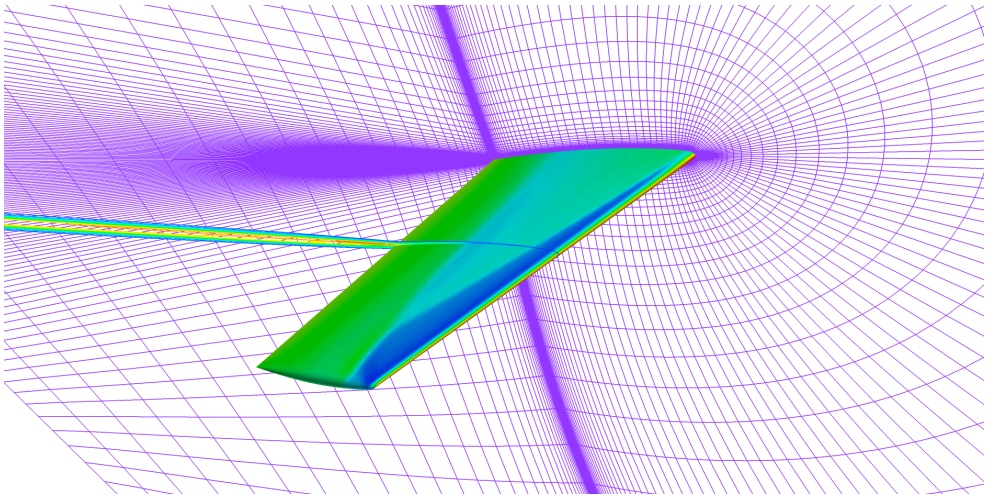

Figure E.16: Onera M6 computational mesh, color coded surface pressure, and turbulent viscosity contour lines

E.1.3 Onera M6 Transonic Wing

The Onera M6 wing is a well known test case for flows about wings in transonic flow conditions. The geometry of the Onera M6 wing is as follows: a root chord of $C_r = 0.8059\ m$ a semi-span of $\frac{b}{2} = 1.1963\ m$, the aspect ratio is $AR = 3.8$, the taper ratio is $\lambda = 0.56$, and the sweep angle of the quarter chord is $\Lambda_{25\%} = 26.7°$. The airfoil is relatively thick and it is symmetric.

The flow conditions were set to a free stream Mach number of $M_\infty = 0.84$ and an angle of attack of $\alpha = 3.06°$. Experimental data for this specific case is readily available. Numerical simulations were conducted assuming viscous turbulent flow with a Reynolds number of $Re = 11.72 \times 10^6$. Figure E.16 shows a slice of the computational mesh, color coded surface pressure (where the λ shock can be clearly seen) , and contour lines of the turbulent viscosity. This simulation was conducted using DDADI inversion (with two sub-iterations) and the $k-\omega$-TNT turbulence model (see Figures E.17 and E.18). A representative comparison of the calculated surface pressure coefficient with the experimental one, at a semi-span location of $\frac{y}{b/2} = 0.9$ can be found in Figures E.19.

```
$TIMINP
 GDTI = -5.0 ! Default is -1.0 !
 GDTF = -200.0 ! Default is -1.0 !
 FSA = 0.5 ! Default is 0.5 !
 DDT = 0.0 ! Default is 0.0 !
 H = 1.0 ! Default is 1.0 !
$END

$METINP
 IMET = 1 ! Default is 1 !
 IFSFX = 2 ! Default is 3 !
 IFSFY = 2 ! Default is 3 !
 IFSFZ = 2 ! Default is 3 !
 ILIMITER = 1 ! Default is 1 !
 NRK = 1 ! Default is 1 !
 IDODDADI = 2 ! Default is 0 !
 NSUBITER = 1 ! Default is 1 !
$END
```

Figure E.17: Time step and method for the Onera M6 case

```
$VISINP
 IVISG = 1 ! Default is 0 !
 ITURG = 6 ! Default is 0 !
 RTINIT = 0.01 ! Default is 0.01 !
 TUINT = 0.02 ! Default is 0.02 !
 VMUETINF = 0.1 ! Default is 0.1 !
!  GDT_TURBI = -2.0 !Default is GDTI !
!  GDT_TURBF = -20.0 !Default is GDTF !
 IDES = 0 ! Default is 0 !
 ITORDD = 1 ! Default is 1!
 ITMIMP = 1 ! Default is 1!
$END
```

Figure E.18: Turbulence model input ($k-\omega$-TNT) for the Onera M6 case

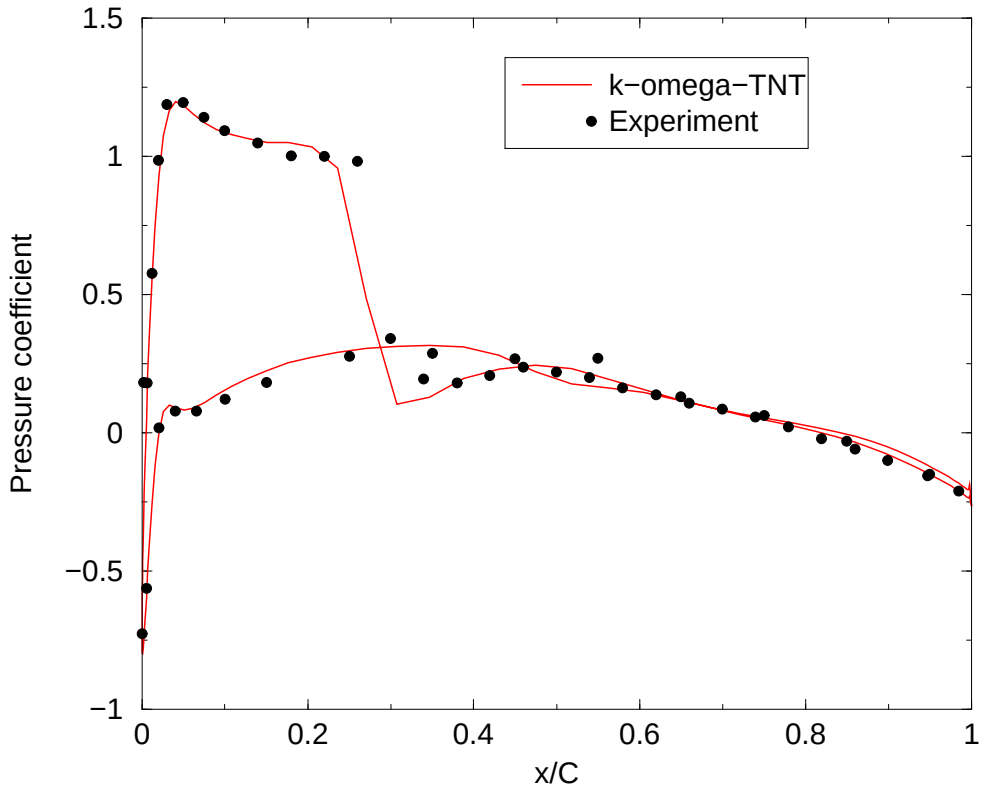

Figure E.19: Pressure coefficient at $\frac{y}{b/2} = 0.9$

E.2 Aeroelasticity Module

E.2.1 Basic Wing-Fuselage-Tail Test-Case - The PHDP

The basic test case is that of a generic transport model (a.k.a. PhD plane, or PHDP) that includes thre aerodynamic grid zones for the fuselage, wing and tail, plus collar zones. The structural grids are provided in three separate files, *grid_s_101.dat*, *grid_s_102.dat*, and *grid_s_103.dat*, for the fuselage, wing and tail respectively. Figure E.20 presents a sample grid file structure. The modes are provided in NASTRAN punch format, in files *mode_s_101.dat*, *mode_s_102.dat*, and *mode_s_103.dat*, for the fuselage, wing and tail respectively. Figure E.21 presents a sample modes-file structure. Each mode file has 15 modes, out of which two are rigid-body modes. The first five modes are splined to the aerodynamic surface grids (N_MODES=5), and out of them only the first three elastic modes are used in the aeroelastic analysis. The two rigid body modes (N_RBMODES = 2) are splined to the aero grids but are not used in the aeroelastic analysis.

Figures E.22 and E.23 present the elastic- and spline-related inputs, respectively, in the main input file *eznss.i.defaults*. In this run IELAST = -1, namely the code performs the spline and exists (no aerodynamic/aeroelastic analysis is performed). The rest of the parameters in the ELAINP namelist are therefore irrelevant for this analysis. Figure E.24 presents the first two entries in file *spline.i* for this test case, indicating that the first structural zone (fuselage) goes through beam spline (IS_SS=0), while the second and third structural zones (wing and tail) go through surface spline (IS_SS=1), and that aerodynamic zone number 1 gets its spline info from structural zone number 1 (ISPLINE_FROM = 1), and similarly 2 from 2 and 3 from 3. The rest of the entries in file *spline.i* get their default values.

Figure E.25 presents the wing surface mesh (grey, only every other i grid), and on top of it the structural nodes that are used in the wing mode spline (red, connected in line). The structural data is from file *struct_100.dat*. Plotting the structural spline nodes on top of the aerodynamic surfaces serves for validation of the selected structural spline nodes. Figure E.26 presents the first elastic mode, plotted from file *mode_a_103.dat* (blue), together with the undeformed surface grids (grey). Similar

```
GRID     7001           0.      0.      0.
GRID     7002           .666667 0.      0.
GRID     7003           1.33333 0.      0.
GRID     7004           2.      0.      0.
GRID     7005           2.66666 0.      0.
GRID     7006           3.33333 0.      0.
GRID     7007           4.      0.      0.
GRID     7008           4.66666 0.      0.
GRID     7009           5.33333 0.      0.
GRID     7010           6.      0.      0.
GRID     7011           6.66666 0.      0.
GRID     7012           7.33333 0.      0.
GRID     7013           8.25537 0.      0.
GRID     7014           8.66666 0.      0.
GRID     7015           9.33333 0.      0.
GRID     7016           10.0    0.      0.
GRID     7017           10.6666 0.      0.
GRID     7018           11.3333 0.      0.
GRID     7019           12.0000 0.      0.
GRID     7020           12.6666 0.      0.
GRID     7021           13.3333 0.      0.
GRID     7022           14.0000 0.      0.
GRID     7023           14.6666 0.      0.
GRID     7024           15.3333 0.      0.
GRID     7025           16.2000 0.      0.
GRID     7026           16.6666 0.      0.
GRID     7027           17.3333 0.      0.
GRID     7028           18.0000 0.      0.
GRID     7029           18.6666 0.      0.
GRID     7030           19.3333 0.      0.
GRID     7031           20.     0.      0.
```

Figure E.20: Sample grid file grid_s_101.dat

```
$TITLE   = NACA PLANE MODAL ANALYSIS                                          1
$SUBTITLE=                                                                    2
$LABEL   =                                                                    3
$EIGENVECTOR                                                                  4
$REAL OUTPUT                                                                  5
$SUBCASE ID =           1                                                     6
$EIGENVALUE =  0.0000000E+00  MODE =     1                                    7
      7001       G      6.654263E-19      0.000000E+00      1.389617E-02      8
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00      9
      7002       G      7.421637E-19      0.000000E+00      1.382501E-02     10
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     11
      7003       G      6.910631E-19      0.000000E+00      1.375385E-02     12
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     13
      7004       G      4.513296E-19      0.000000E+00      1.368269E-02     14
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     15
      7005       G     -6.634158E-19      0.000000E+00      1.361153E-02     16
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     17
      7006       G     -1.534254E-18      0.000000E+00      1.354037E-02     18
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     19
      7007       G     -3.291892E-18      0.000000E+00      1.346921E-02     20
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     21
      7008       G     -3.733346E-18      0.000000E+00      1.339805E-02     22
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     23
      7009       G     -2.847275E-18      0.000000E+00      1.332689E-02     24
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     25
      7010       G     -1.793405E-18      0.000000E+00      1.325573E-02     26
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     27
      7011       G     -6.701911E-19      0.000000E+00      1.318457E-02     28
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     29
      7012       G     -2.451984E-19      0.000000E+00      1.311340E-02     30
-CONT-                   0.000000E+00      1.067413E-04      0.000000E+00     31
```

Figure E.21: Sample modes file mode_s_101.dat

```
$ELAINP
 IELAST = -1 ! Default is 0 !
 IDEFMET = 1 ! Default is 1 !
 ITER_ELAST_STAGE = 0 ! Default is 0 !
 ELAST_FACT = 1.0 ! Default is 1.0 !
 DAMPING = 0.0 ! Default is 0.0 !
$END
```

Figure E.22: Elastic related inputs in the main input file *eznss.i.defaults*

```
$SPLINP
 IS_FSP = 0 ! Default is 0 !
 IS_TAS = 1 ! Default is 1 !
 NSSURF = 3 ! Default is 1 !
 ISFLAG = 0 ! Default is 1 !
 N_MODES = 5 ! Default is 1 !
 N_RBMODES = 2 ! Default is 0 !
 MFORM = 1 ! Default is 1 !
 SCALE = 1.0 ! Default is 1.0 !
 SCALEM = 1.0 ! Default is 1.0 !
 SCALEA = 1.0 ! Default is 1.0 !
 PLOT_FACTOR = 10.0 ! Default is 1.0 !
$END
```

Figure E.23: Spline related inputs in the main input file *eznss.i.defaults*

```
$IS_SS
1 0
2 1
3 1
$END
$ISPLINE_FROM
1 1
2 2
3 3
$END
```

Figure E.24: First entries in input file *spline.i*

plots can be plotted for each mapped mode.

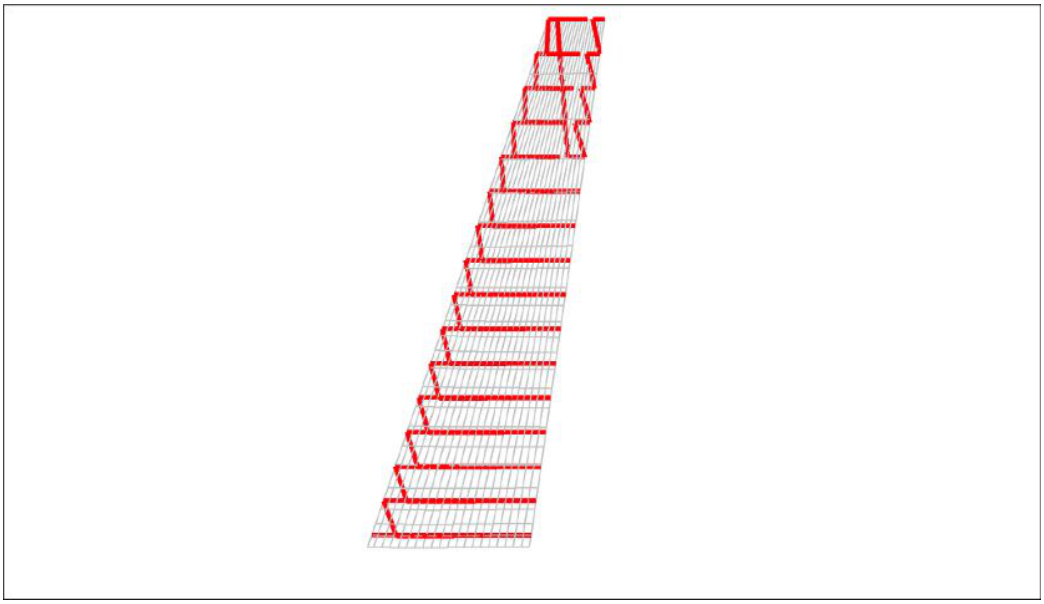

Figure E.25: Structural grid of the PHDP wing

E.2.2 Multi-block Wing

The multi-block (MB) wing is the wing of the PHDP that was broken into four grid zones (three grid zones for the wing plus a 'world-grid' that wraps around the wing grids and extends to the far field), in order to demonstrate the spline procedure using multiple aerodynamic grid zones. Figure E.27 shows the overlapping surface grids of the three wing zones. The structural grid locations are shown in figure E.28 in blue (with the aerodynamic grid in the background in light grey). Notice that there is only a single structural zone. The spline procedure maps the modes provided in the grids of this single structural zone into the grids of the three aerodynamic zones.

The elastic- and spline-related entries in file *eznss.i.defaults* are similar to those presented for the PHDP test case. Figure E.29 presents the first two entries in file *spline.i* for the MB wing test case, indicating that the first (and only) structural zone (wing) goes through surface spline (IS_SS=1). Aerodynamic zone number 1

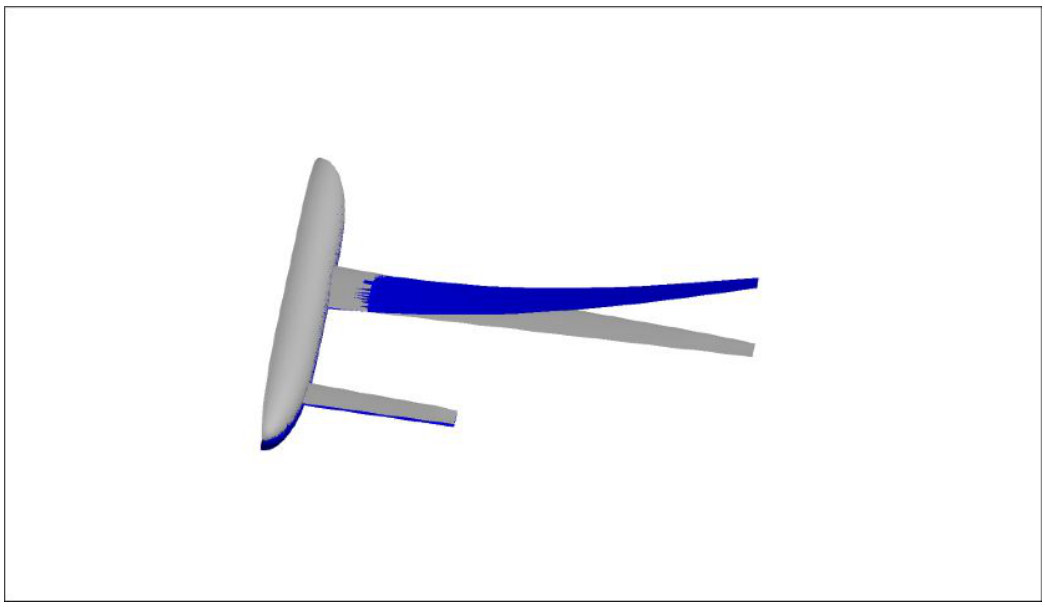

Figure E.26: First elastic mode shape mapped to the surface grid

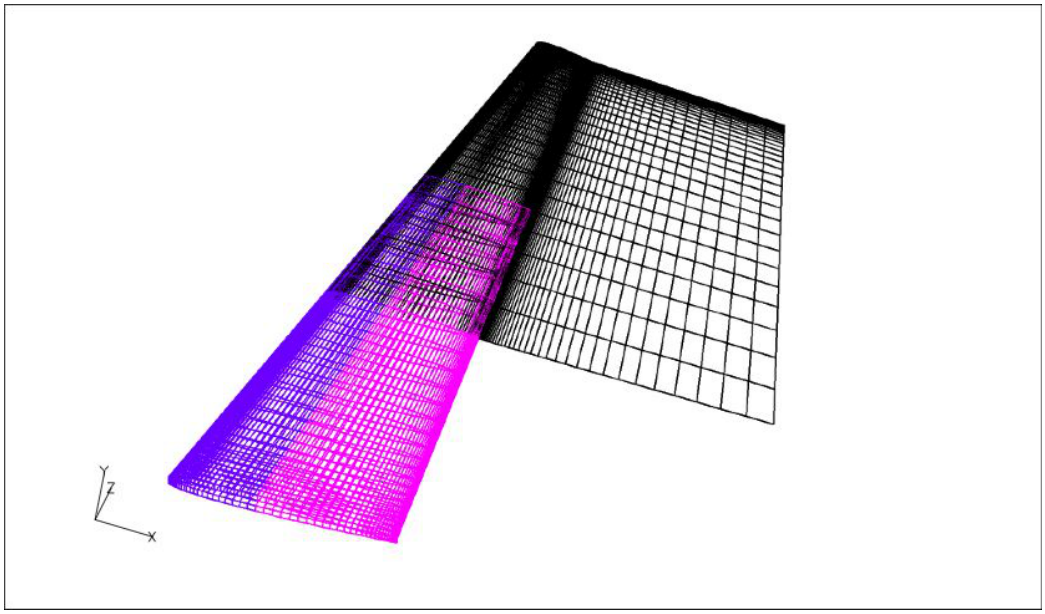

Figure E.27: Aerodynamic surface grid of the MB wing

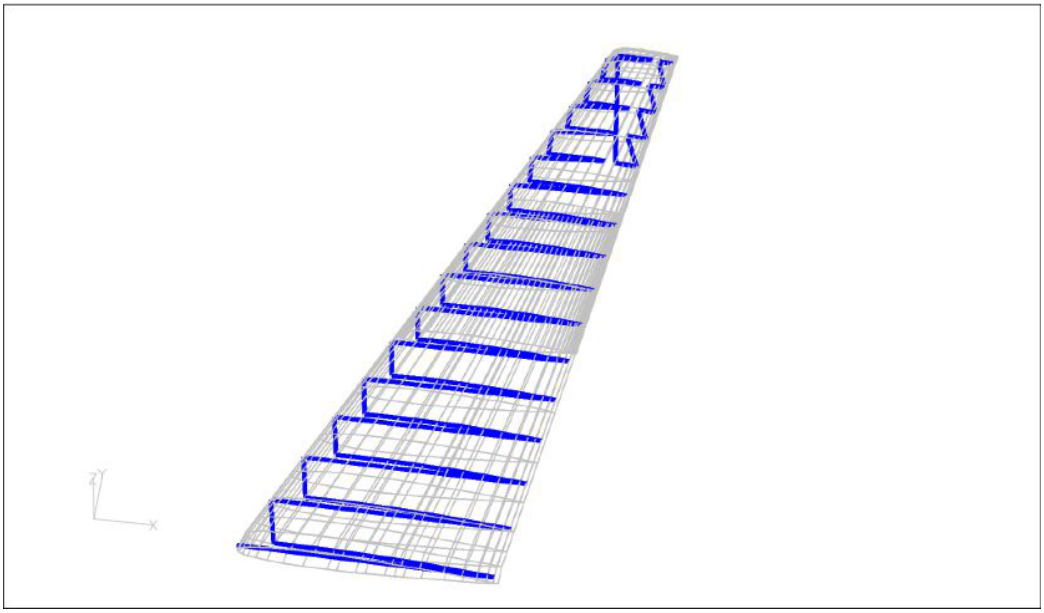

Figure E.28: Structural grid of the MB wing

(world zone) does not go through spline (ISPLINE_FROM = 0), while aerodynamic zones 2-3 (segments of the wing) get their spline info from the first structural zone (ISPLINE_FROM = 1). The rest of the entries in file *spline.i* get their default values. Figure E.30 presents the first elastic mode, plotted from file *mode_a_103.dat*, showing the smooth spline across the three zones that make up the wing.

```
$IS_SS
1 1
$END
$ISPLINE_FROM
1 0
2 1
3 1
4 1
$END
```

Figure E.29: First entries in input file *spline.i*

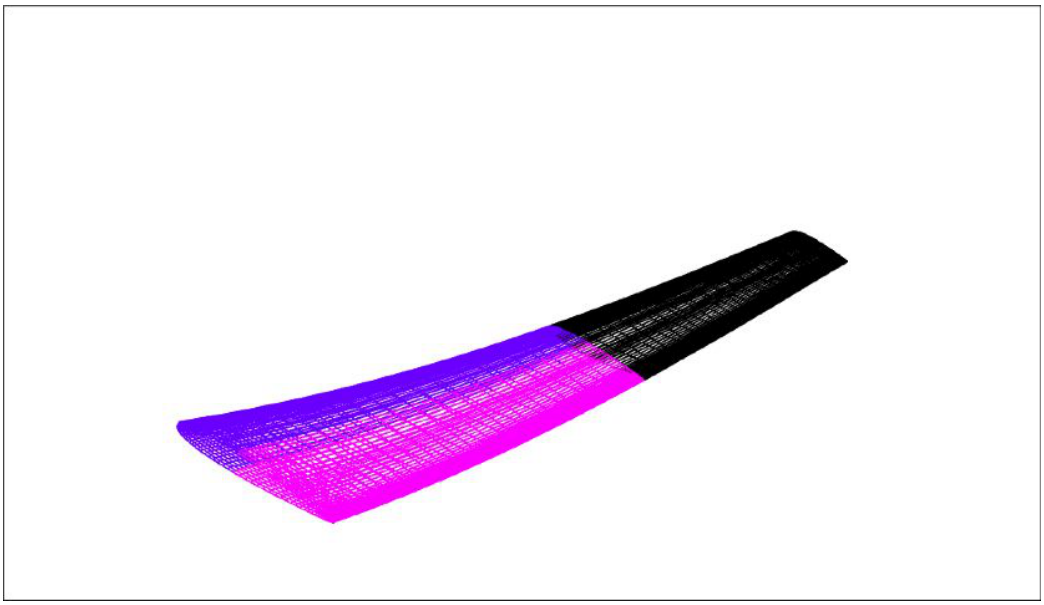

Figure E.30: First elastic mode shape mapped to the MB surface grids

E.2.3 Constrained Deformations - Wing-tip Missile

This test case demonstrates the application of 'constrained deformation' in which a pre-defined sub-domain is rigidly connected via a single point to a physical coordinate on an aerodynamic zone, and moves (translates and rotates) rigidly with it. This would be the case of a wing-tip missile that is connected with a 'bolt' to a point on the wing tip. The missile itself is rigid, does not have structural modal data, and is constrained to move rigidly with the point that it attaches to. The following test case is of the PHDP with added wing-tip missile, shown in grey figure E.31. The missile is intentionally located away from the wing tip. It attaches to the wing (aerodynamic zone 2) at point (10.695, 10.0, 0.0), which does not necessarily coincide with a grid point on the wing. The scheme searches for the closest grid point and attaches the missile sub-domain to it.

The missile aerodynamic zone (zone 4) is defined as a sub-domain in the arrays input file *eznssa.i*, and a single sub-domain is declared in the main input file *eznss.i.defaults* as shown in Figures E.32 and E.33. Figure E.34 shows the entries defining the attachment point (x,y,z coordinates and aerodynamic zone) in file *spline.i*.

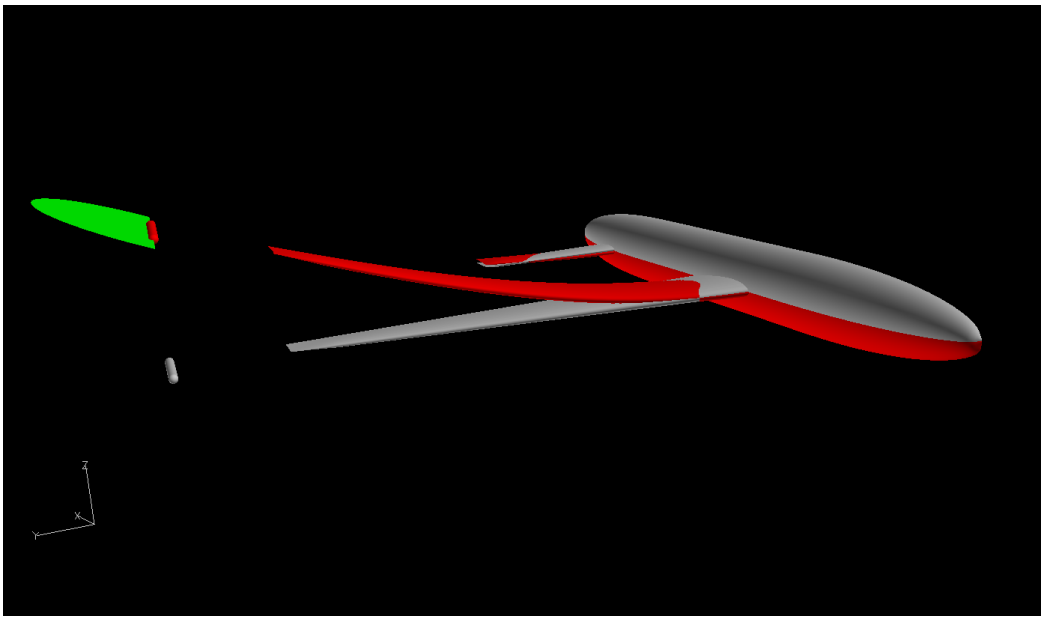

Figure E.31: Wing-tip Missile Test-case - Before and After Elastic Deformatiom

The other non-default entries in this file are as in the basic PHDP test case (see section E.2.1, figure E.24). The elastic and spline-related entries in file *eznss.i.defaults* are also as in the basic PHDP case (Figures E.22 and E.23).

```
$SUBINP
 NSUBD = 1 ! Default is 0 !
$END
```

Figure E.32: Defining the wing-tip missile sub-domain - main input file

E.2.4 Flaps

In this test case a single trailing-edge flap is defined for the PHDP wing, and deflected 5° down. The number of flaps is declared in the main input file *eznss.i.defaults*, in the sub-domain namelist, as shown in Figure E.35. The rest of the flap inputs are in file *flap.i*, shown in Figure E.36. The flap virtual grid is of size 31 × 11, defined by IFLAPDIM and JFLAPDIM. These are default values that performed well for

```
$ISUBD
4 1
$END
```

Figure E.33: Defining the wing-tip missile sub-domain - array input file

```
$SUBDXC
1 10.695
$END
$SUBDYC
1 10.0
$END
$SUBDZC
1 0.0
$END
$NSUBDC
1 2
$END
```

Figure E.34: Defining the attachment point in input file *spline.i*

some test cases. Be careful changing these numbers as too large values may result in singularities in the transformation matrix. IFLAP declares that flap number one resides in grid zone number 2 (the wing). IFLAPLT defines flap number one to be a trailing edge flap. The flap hinge runs from $x = 9.9, y = 3.0$ (defined by variables XFLAPR, YFLAPR) to $x = 10.3, y = 5.0$ (defined by variables XFLAPT, YFLAPT). Finally, the flap deflection of 5° down is defined by variable XIFLAP. Figure E.37 shows the surface mesh after flap deflection, showing the smooth mesh and transition between the flap region and the rest of the wing.

```
$SUBINP
 NSUBD = 0 ! Default is 0 !
 NFLAP = 1 ! Default is 0 !
$END
```

Figure E.35: Defining the number of flaps in the main input file

```
$IFLAPDIM
1 31
$END
$JFLAPDIM
1 11
$END
$IFLAP
1 2
$END
$IFLAPLT
1 1
$END
$XFLAPR
1 9.9
$END
$YFLAPR
1 3.0
$END
$XFLAPT
1 10.3
$END
$YFLAPT
1 5.0
$END
$XIFLAP
1 5.0
$END
```

Figure E.36: Flap input file

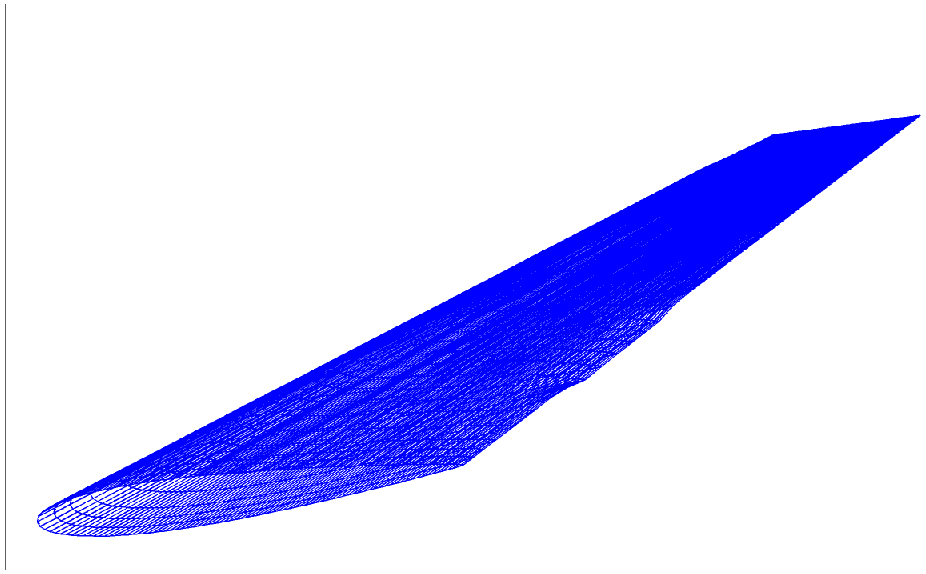

Figure E.37: Surface mesh of PHDP wing with deflected trailing-edge flap

E.2.5 Prescribed Sinusoidal Flap Motion

In this test case a single trailing-edge flap is defined for a High Altitude Long Endurance (HALE) wing model. An aeroelastic simulation simulates the dynamic response of the highly elastic wing to prescribed flap excitation of 1Hz frequency and 1° amplitude.

The relevant inputs in the main input file *eznss.i.defaults* are shown in figure E.38. The case is declared to be elastic (IELAST=1), with solution of the dynamic aeroelastic equation following each iteration (ITER_ELAST_STAGE=1). The remaining aeroelastic parameters are set to their default values. Spline parameters are defined in namelist SPLINEP. Ten modes (N_MODES=10, no rigid boy modes, N_RBMOES=0) are read from Nastran punch file (MFORM=1). The Nastran finite-element model is in length units of ft, and mass units of slugs, hence the scaling factors (SCALE = 3.2808 and SCALEM = 0.06852).

A single flap (NFLAP=1) is defined in the SUBINP (sub-domain) namelist. Flap sinusoidal excitation is prescribed in the excitation namelist, EXCINP (IEXC = 13). The excitation frequency of the first (and only) flap is set to 1 Hz (EXC_FREQ(1)

= 1.0). Note that the the flap amplitude is NOT defined by the EXC_XIMAX that follows, but rather by XIFLAP in file *flap.i* (description follows). EXC_XIMAX is set to its default value of zero. VTIME0 defines the end time of the static analysis from which the current analysis is restarted. The rest of the inputs in name list EXCINP do not apply for flap motion.

The rest of the flap inputs are in file *flap.i*, shown in figure E.39. The flap virtual grid is kept at the default size, as indicated by the empty IFLAPDIM and JFLAPDIM entries. IFLAP declares that flap number one resides in grid zone number 1. IFLAPLT defines flap number one to be a trailing edge flap. The flap hinge runs from $x = 1.8, y = 27.7$ (defined by variables XFLAPR, YFLAPR) to $x = 1.8, y = 32.9$ (defined by variables XFLAPT, YFLAPT). Finally, flap deflection amplitude of 5° down is defined by variable XIFLAP. Figure E.37 shows the surface mesh after flap deflection, showing the smooth mesh and transition between the flap region and the rest of the wing.

```
$ELAINP
 IELAST = 1 ! Default is 0 !
 IDEFMET = 1 ! Default is 1 !
 ITER_ELAST_STAGE = 1 ! Default is 0 !
 ELAST_FACT = 1.0 ! Default is 1.0 !
 DAMPING = 0.0 ! Default is 0.0 !
$END

$SUBINP
 NSUBD = 0 ! Default is 0 !
 NFLAP = 1 ! Default is 0 !
$END

 $SPLINP
 IS_FSP = 0 ! Default is 0 !
 IS_TAS = 1 ! Default is 1 !
 NSSURF = 1 ! Default is 1 !
 ISFLAG = 1 ! Default is 1 !
 N_MODES = 10 ! Default is 1 !
 N_RBMODES = 0 ! Default is 0 !
 MFORM = 1 ! Default is 1 !
 SCALE = 3.2808 ! Default is 1.0 !
 SCALEM = 0.06852 ! Default is 1.0 !
 SCALEA = 1.0 ! Default is 1.0 !
 PLOT_FACTOR = 10.0 ! Default is 1.0 !
$END

 $EXCINP
 IEXC = 13 ! Default is 0 !
 EXC_FREQ(1) = 1.0 ! Default is 0.0 !
 EXC_XIMAX(1) = 0.0 ! Default is 0.0 !
 VTIME0 =  0.0 ! Default is 0.0 !
 ISTEPOSS = 0 ! Default is 0 !
 NIMP = 0 ! Default is 0 !
 N_EXC_ST = 0 ! Default is 0 !
$END
```

Figure E.38: Defining elastic and flap inputs in the main input file for the case of flap prescribed sinusoidal motion

```
$IFLAPDIM
$END
$JFLAPDIM
$END
$IFLAP
1 1
$END
$IFLAPLT
1 1
$END
$XFLAPR
1 1.8
$END
$YFLAPR
1 27.7
$END
$XFLAPT
1 1.8
$END
$YFLAPT
1 32.9
$END
$XIFLAP
1 5.0
$END
```

Figure E.39: Flap input file for the case of flap prescribed sinusoidal motion

Bibliography

[1] Benek J. A.and Buning P. G. and Steger J. L. a 3-d chimera grid embedding technique. In *7th Computational Physics Conference*, Cincinnati, OH, July 1985. AIAA Paper 85-1523.

[2] Peyret R. and Viviand H. Computation of viscous compressible flows based on the navier-stokes equations. 1975. AGARD-AG-212.

[3] Viviand H. Conservation forms of gas dynamic equations. *La Recherche Aerospatiale*, 1:65–66, 1974.

[4] Vinokur M. Conservation equations of gasdynamics in curvilinear coordinate systems. *Journal of Computational Physics*, 14:105–125, 1974.

[5] Beam R. M. and Warming R. F. An implicit finite-difference algorithm for hyperbolic systems in conservation law form. *Journal of Computational Physics*, 22:88–110, 1976.

[6] Beam R. M. and Warming R. F. An implicit factored scheme for the compressible navier-stokes equations. *AIAA Journal*, 16:393–402, 1978.

[7] Baldwin B. S. and Lomax H. Thin layer approximation and algebraic model for separated turbulent flows. In *AIAA 16th Aerospace Sciences Meeting*, Reno, Nevada, January 1978. AIAA Paper 78-257.

[8] Steger J. L. and Warming R. F. Flux vector splitting of the inviscid gasdynamic equations with applications to finite-difference methods. *Journal of Computational Physics*, 40:263–293, 1981.

[9] Steger J. L., Ying S. X., and Schiff L. B. A partially flux-split algorithm for numerical simulation of unsteady viscous flows. In *Workshop on Computational Fluid Dynamics*, University of California, Davis, 1986.

[10] Ying S. X., Baganoff D., Steger J. L., and Schiff L. B. Numerical simulation of unsteady, viscous, high-angle-of-attack flows using a partially flux-split algorithm. In *AIAA 13th Atmospheric Flight Mechanics Conference*, August 1986. AIAA Paper 86-2179-CP.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

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

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

[19] C. C. Rossow. A flux-spltting scheme for compressible and incompressible flows. *Journal of Computational Physics*, 164(1):104–122, 2000.

[20] J. R. Edwards. A low-diffusion flux-splitting scheme for Navier-Stokes calculations. *Computers and Fluids*, 26(6):653–659, 1997.

[21] Antony Jameson. Artificial diffusion, upwind biasing, limiters and thier effect on accuracy and multigrid convergence in transonic and hypersonic flows. In *Computational Fluid Dynamics Conference*, Orlando, FL, July 1993. AIAA - 93 - 3359.

[22] C. C. Rossow. Efficient computation of compressible and incompresible flows. *Journal of Computational Physics*, 220(2):879–899, 2007.

[23] Cebeci T., Smith A. M. O., and Mosinkis G. Calculation of compressible adiabatic turbulent boundary layers. *AIAA Journal*, 8:1974–1982, 1970.

[24] Degani D. and Schiff L. B. Computation of turbulent supersonic flows around pointed bodies having crossflow separation. *Journal of Computational Physics*, 66(1):173–196, 1986.

[25] Degani D., Schiff L. B., and Levy Y. Numerical prediction of subsonic, turbulent flows over bodies at large incidence. *AIAA Journal*, 29(12):2054–2061, December 1991.

[26] Degani D. and Levy Y. Asymmetric turbulent vortical flows over slender bodies. *AIAA Journal*, 30(9):2267–2273, September 1992.

[27] Goldberg U. and Peroonian O. Hypersonic flow heat transfer prediction with wall-distance-free turbulence model. *Computational Methods and Experimental Measurements*, pages 261–270, 1999.

[28] Goldberg U. Hypersonic flow heat transfer prediction using single equation turbulence model. *Journal of Heat Transfer*, pages 65–69, 2001.

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

[30] 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.

[31] C. Kok, Johan, H. S. Doi, B. Oskam, and H. van der Van. Extra-large eddy simulation of massively separated flows. In *42th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 2004. AIAA paper 2004 - 264.

[32] P.R. Spalart, S. Deck, M.L. Shur, K.D. Squires, M.K. Strelets, and A. Travin. A new version of detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and Computational Fluid Dynamics*, 20(3):181–195, 2006.

[33] K. Abe. A hybrid LES/RANS approach using an anisotropy-resolving algebraic turbulence model. *International Journal of Heat and Fluid Flow*, 26(2):204–222, 2005.

[34] Mikhail L. Shur, Philippe R. Spalart, Mikhail Kh. Strelets, and Andrey K. Travin. A hybrid LES-RANS approach with delayed-DES and wall-modelled LES capabilities. *International Journal of Heat and Fluid Flow*, 29(6):1638–1649, 2008.

[35] Chan W. M. and Buning P. G. Surface grid generation methods for overset grids. *Computers & Fluids*, 24(5):509–522, 1995.

[36] S. E. Rogers, H. V. Cao, and T. Y. Su. Grid generation for complex high-lift configurations. In *29th AIAA Fluid Dynamics Conference*, Albuquerque, NM, June 1998. AIAA Paper 98-3011.

[37] Parks S. J., Buning P. G., Chan W. M., and Steger J. L. Collar grids for intersecting geometric components within the chimera overlapped grid scheme. In *10th AIAA Computational Fluid Dynamics Conference*, Honolulu, HI, June 1991. AIAA Paper 91-1587.

[38] Thompson J. F., Thames F. C., and Mastin C. W. Automatic numerical generation of body-fitted curvilinear coordinate systems for fields containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, 15(3):299–319, 1974.

[39] OpenMP. *OpenMP Fortran Application Program Interface*. http://www.openmp.org.

[40] *The Message Passing Interface Standard*. http://www-unix.mcs.anl.gov/mpi.

[41] *Parallel Virtual Machine*. http://www.csm.ornl.gov/pvm.

[42] Ayguade E., Gonzalez M., Martorelli X., and Jost G. Employing nested openmp for the parallelization of multi-zone computational fluid dynamics applications. In *Proceedings of the 18th International Parallel and distributed Symposium*, Santa Fe, NM, April 2004.

[43] Intel. *Intel Fortran Compiler for Linux*. http://www.intel.com:80/cd/software/products/asmona/eng/compilers/flin/index.htm.

[44] Cook Nick. Royal air force equips with raptor. *Interavia*, 54(629):59–61, March 1999.

[45] W. Haase, F. Bradsma, E. Elsholz, M. Leschziner, and Schwamborn D. EUROVAL - An European initiative on validation of CFD codes. Technical report, 1992. Notes on Numerical Fluid Mechanics.