A Space-Marching Compressible Flow Solver with Chemistry and Optimization

SM_3D^{PLUS}

Chris S. Craddock & Peter A. Jacobs

Department of Mechanical Engineering,

The University of Queensland,

Brisbane 4072, Australia.

June 1998

Abstract

As an extension to the Space-Marching Euler Solver for Supersonic Flow in a Three-Dimensional Geometry (sm_3d) [1], the set of Parabolized Navier-Stokes (PNS) equations was added to the numerical scheme. Due to the space-marching nature of the code, Vigneron's technique of splitting the streamwise pressure gradient in subsonic regions was employed to stabilize the flux calculations. Other additions to the code include a turbulence model, a finite rate chemistry model and fast thermodynamic equilibrium models.

Several test cases are included in this report to validate the new equation sets and numerical technique. The test cases are also intended to be used as tutorial examples for readers who wish to become familiar with the use of the code.

Contents

Nomenclature

A : side of cell area, m^2

A_j : constant in Arrhenius law for reaction j

a : local speed of sound, m/s

C : mass concentration, mol/kg

C_p : coefficient of heat capacity (constant pressure), J/(kg.K)

C_v : coefficient of heat capacity (constant volume), J/(kg.K)

CFL : Courant-Friedrichs-Lewy number

d : distance from wall used in turbulence model, m

E^o : molar internal energy, J/mole

E_j : activation energy of reaction j, J/mol

E : total specific internal energy, J/kg

e : specific internal energy, J/kg

$\mathbf{F}$: algebraic vector of fluxes in conservation equations

f_i : mass fraction of species i

G^o : molar Gibbs free energy, J/mol

H^o : molar enthalpy, J/mol

$(\Delta H_f^o)_{T_R}$: formation enthalpy for a reference temperature T_R, J/mol

$\mathbf{H}$: source vector in conservation equations

h : specific enthalpy, J/kg

$\hat{i}, \hat{j}, \hat{k}$: unit vectors in Cartesian coordinates

K : equilibrium constant

k_B : Boltzmann's constant, 1.38066×10^{-23}J/K

k_f : forward reaction rate

k_r : reverse reaction rate

k : coefficient of thermal conductivity

L : characteristic length across a cell, m

l : integer difference in stoichiometric coefficients

M : Mach number

M : Molecular weight, g/mol

N_R : number of reactions

N_S : number of species

$\hat{n}$: unit normal vector

$\overline{P}$: position vector

p : pressure, Pa

p_{atm} : standard atmospheric pressure, 101.3×10^3 Pa

Pr : Prandtl number, $C_p\mu/k$

$\mathbf{Q}$: algebraic vector of source terms

q : viscous heat flux, J/(m^2.s)

R^o : universal gas constant, 8.31451 J/(mol.K)

R_i : gas constant for species i, J/(kg.K)

$\tilde{R}$: mixture gas constant, J/(kg.K)

$\mathbf{R}$: steady state residual vector

Re : Reynolds number

r : radial coordinate, m

S^o : molar entropy, J/(mol.K)

S : control surface of the cell

dS : cell-interface area, m^2

T : temperature, K

T^* : reduced temperature

t : time, s

t_{vib} : vibrational relaxation time, s

t_{fluid} : fluid dynamic characteristic time, s

$\hat{t}_1, \hat{t}_2$: unit tangent vectors

Δt : time step, s

$\mathbf{U}$: algebraic vector of conserved quantities

u : velocity, m/s

$\tilde{u}$: diffusion velocity of species i, m/s

V : volume, m^3

X_i : mole fraction of species i

x, y, z : Cartesian coordinates, m

Z_i : chemical symbol of species i

α : stoichiometric coefficient

β : compression parameter in the MUSCL interpolation/extrapolation

γ : ratio of specific heats

ϵ : characteristic energy, J

ε : Vigneron's coefficient

θ : activation temperature, K

κ : spatial accuracy constant for MUSCL interpolation/extrapolation

λ : second coefficient of viscosity, Pa.s

τ : shear stress, Pa

μ : coefficient of viscosity, Pa.s

ν : diffusion velocity, m/s

ρ : density, kg/m^3

σ : collision diameter, ($\mathring{\text{A}}$)

Φ : safety factor for Vigneron's pressure splitting

ϕ : rate of heat added per unit volume, J/(m^3.s)

ψ : half angle for axisymmetric cells, rad

$\Omega^{(2,2)}$: elastic collision integral

$\dot{\omega}$: production rate of species per unit volume, kg/s/m^3

ξ, η, ζ : normalised coordinates

Superscripts

T : transpose

o : molar quantity

$'$: reactant

$''$: product

n : time level

u : pertaining to the up-stream vertex slice

$\cdot$: time differential

Subscripts

A, B, C, D	: vertex labels
ix, iy, iz	: cell-centre indices
$ix \pm \frac{1}{2}$	: "vertical, streamwise" interface
$iz \pm \frac{1}{2}$	: "vertical, cross-stream" interface
$iy \pm \frac{1}{2}$	: "horizontal, cross-stream" interface
inv	: inviscid contribution to fluxes
i	: species number
j	: reaction number
lam	: laminar
MIN	: minimum allowable value
N, S, E, W	: North, South, East, West interface or boundary
n	: normal to interface
R	: reference
t	: tangent to interface
turb	: turbulent
vis	: viscous contribution to fluxes
wall	: wall value
x, y, z	: Cartesian components
L, R	: left state, right state

1 Introduction

The space-marching Euler solver, *sm_3d*, [1] was written primarily to compute the flow field through scramjet modules. Recently, it was used to compute the inviscid flow through a three-dimensional scramjet module and as the flow solver for the optimization of a two-dimensional scramjet combustor and exhaust nozzle [2]. The solver was based on a finite-volume formulation and used an *upwinding* technique to "capture" shocks. It was capable of computing inviscid supersonic flows through domains bounded by four side surfaces, an inlet plane and an outlet plane. The gas was modeled as calorically perfect (constant specific heats) making the thermodynamics model very simple. Heat could be added as a source term in the energy conservation equation to model combustion in scramjet combustors. This heat was added to individual cells based on a specified distribution which was hard-coded for each case. In this form the solver was simple and fast and could be used to obtain thrust estimates of scramjets and provide an insight into complex internal shock distributions. However, its capabilities fell short of modelling real flow applications with high accuracy. This report details the new version called *sm_3d* $^{\text{PLUS}}$ (or abbreviated to *sm_3d*$^+$) that has been written to extend *sm_3d* in a couple of ways.

1.1 Motivation for Extending the Code

The motivation for extending the modelling capabilities of *sm_3d* stems from two avenues of hypersonic research being conducted in the Mechanical Engineering Department. They are the design of high quality nozzles for hypersonic impulse test facilities and optimal scramjet combustor/exhaust nozzles. In both cases, the development of boundary layers and finite rate chemical kinetics are important design issues. To extend *sm_3d* and include these effects, the governing equation set is changed from the Euler equations to the Parabolized Navier-Stokes (PNS) equations. The PNS equations are used in favour of the full Navier-Stokes equations because they are computationally cheaper to solve [3]. A model for chemical kinetics is also implemented along with an appropriate thermodynamics model [4].

The new version of the solver is also capable of modelling turbulent boundary layers. Turbulent boundary layers can develop in nozzles and scramjets resulting in significant increases of boundary layer thickness, skin friction and wall heating when compared with laminar flows. A simple algebraic turbulence model by Baldwin & Lomax [5] is implemented because it is relatively cheap in comparison to more complex models which seem to offer little improvement when the flow remains unseparated [6] [7].

Computer-assisted design optimization has become quite popular in recent times because of increasing computing power which makes the technique feasible. The new solver is accompanied by a nonlinear simplex minimisation procedure that is based on the technique proposed by Nelder & Mead [8]. The procedure is an adaptation of the FORTRAN code written by O'Neill [9][10]. This optimization procedure was used to optimize a two-dimensional scramjet combustor and exhaust nozzle by treating the generated thrust as the objective function, and using control points which specify the duct and nozzle shape as the optimization parameters (see Section 6.8). The Nelder-Mead optimizer is one of many optimization techniques currently available and it provides reasonable performance without the need for objective function derivatives.

1.2 Attributes of the Space-Marching Code

The original space-marching formulation of the original *sm_3d* is maintained in the present code. Srinivas [11] showed that space-marching techniques require an order of magnitude less computational time than whole domain time-marching techniques for supersonic flows, where upstream condition are independent of downstream conditions. The system of equations that govern such a flow regime can be shown to be hyperbolic in nature. Viscous flow is elliptical in nature and therefore needs to be treated with a time-marching technique to produce a truly accurate solution. However, if the subsonic regime is confined to thin boundary layers and the remaining flow is supersonic, then the Parabolized Navier-Stokes (PNS) equations can be used as a more efficient set of governing equations. The elliptic viscous Navier-Stokes (NS) equations are made parabolic by neglecting all streamwise viscous terms. Korte [12] has shown that a space-marching flow solver based on the PNS equations shows good agreement with a time-marching NS solver for steady supersonic and hypersonic viscous flow problems that do not have a strong downstream influence. Therefore, the supersonic viscous flow can be solved efficiently and with reasonable accuracy for engineering applications using the PNS equations in a space-marching technique.

The basis of the space-marching technique is sub-dividing the flow domain into slices orthogonal to the streamwise direction and solving the steady state for each slice in turn starting from the inflow plane. If the flow is mostly supersonic in the streamwise direction, the inflow to the upstream face of each slice can be taken as fixed at the outflow conditions of the immediately-upstream slice. Each slice is discretised into a two-dimensional grid of hexahedral cells and the flow equations for all cells are iterated in time until a steady state is reached. Flow properties for the outflow plane of the active slice are extrapolated from the current and previous

upwind slices (see Section 2.7). To perform the time-marching at each slice, only enough slices to reconstruct the downwind face are required in memory. Therefore, the technique is very efficient in comparison to a purely time-marching scheme that requires the storage of the whole computational domain.

The PNS equations are formulated on a Cartesian coordinate system without using a generalized coordinate system. This approach reduces coding complexity but requires the dominant supersonic flow direction to be approximately aligned with the x-axis. Therefore, the domain slices need to be approximately perpendicular to the x-axis. This limits the permissible duct geometrys somewhat, however, the bounding domain walls can have significant curvature in both cross-stream and streamwise directions as long as this condition is maintained.

The space-marching technique assumes that the flow is steady and supersonic in the free stream. An implication of this assumption is that the technique cannot simulate flows with streamwise recirculation regions. However, recirculation regions in the cross flow are still attainable, which permits the study of cross flow separation on curved surfaces where cross stream pressure gradients may be large. Lastly, note that time-marching within each slice does not represent a physical evolution of the flow. The solution produced is forced to be a steady-state flow solution.

1.3 Outline of Report

The body of this report contains the details of the extensions incorporated in *sm_3d*$^+$ with an emphasis on the details of the new thermodynamic models and optimization technique. The numerical implementation of the models is shown as well as the grid generation techniques that *sm_3d*$^+$ supports.

The latter sections of the report explain how to use the solver and the source code that makes up *sm_3d*$^+$. Some examples and test cases are presented to demonstrate the solver's capabilities and provide a set of tutorial problems.

2 Formulation of Flow Solver

As for *sm_3d*, the present solver is based on a pseudo-time dependent, finite volume formulation with space-marching in the streamwise direction. The differences lie in the viscous components of the governing equations and the thermodynamic/chemical models that are built on top of the original solver. The Euler equations in the original solver are replaced with the Parabolized Navier-Stokes (PNS) equations to allow simulation of viscous flow. A multiple-species thermodynamic equilibrium model is added along with a number of other simple thermodynamic models for non-reacting gases. Heat addition, resulting from combustion in the flow, can now be modeled with a more complete combustion model that includes finite-rate chemical reactions. The original simplification of adding heat through a source term may still be used in the present code for fast approximations. Other additions include a new and faster flux solver, turbulence model and optimizer all of which are described in the following sections.

For the present formulation, three main assumptions have been made. Firstly, the flow is assumed to be in thermodynamic equilibrium such that the internal energy of the gas is always a function of the vibrational temperature, $e = f(T_{\text{vib}})$. This has direct implications for the thermodynamic model, which will be discussed in Section 2.2. Secondly, transport of species due to diffusion is assumed to be negligible. Therefore the diffusion velocity of all species is zero. Lastly, the flow is assumed to always have a positive momentum value in the x direction which is the space marching direction and predominant flow direction.

2.1 Governing Equations

For applications such as complete scramjet simulation and hypersonic nozzles with thick boundary layers, a solution of the full Navier-Stokes (NS) equations is required for accurate simulations. However, a price is paid for the computational effort which can be quite large even on current "super computers". If the flow in the streamwise direction is always supersonic (excluding boundary layers), the more efficient Parabolized Navier-Stokes (PNS) equation set can be used. PNS solutions are close to NS solutions for steady supersonic and hypersonic viscous flow problems which do not have strong downstream influence [12]. The conservative, chemically reacting PNS equations in three dimensions are used in the *sm_3d*$^+$ solver.

2.1.1 Parabolized Navier-Stokes Equations

Rudmann & Rubin [13] were responsible for one of the earliest studies involving the use of the PNS equations. They derived and numerically solved a composite

set of equations that were valid in both the inviscid and viscous regions of steady supersonic and hypersonic flow. The PNS equations are obtained by deleting all the viscous terms containing partial derivatives with respect to the streamwise direction from the steady NS equations. The PNS equations can be used to solve viscous-inviscid interaction problems with a fraction of the computation time used for solving the same problems with the NS equations.

The PNS equations can only be applied to flows where the inviscid region is supersonic and the streamwise velocity component is always positive (that is only cross flow separation is permitted). If the streamwise velocity is assumed to be aligned with one of the coordinate axis, then applications are limited to cases where the predominant flow direction is in the axial direction. This is the assumption made for the *sm_3d*$^+$ formulation. Several other codes avoid this restriction by using a generalized coordinate system [14, 12]. However this adds extra complexity to the code add is not currently required. Having made this restriction, the condition of a supersonic x component of velocity in the inviscid part of the shock layer must be maintained when considering solving a flow.

The integral form of the three-dimensional PNS equations for a chemically reacting, multi-species flow without body forces or external heat addition may be written as

$$\frac{\partial}{\partial t}\int_V \mathbf{U}dV + \oint_S (\mathbf{F_i} - \mathbf{F_v})\cdot\hat{\mathbf{n}}dS = \int_V \mathbf{Q}\mathrm{d}V \quad , \tag{1}$$

in the control volume V bounded by the surface S. The algebraic vector of dependent flow variables is

$$\mathbf{U}= \begin{bmatrix} f_i\rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ \rho E \end{bmatrix} \quad , \tag{2}$$

the inviscid flux vectors is

$$\mathbf{F_i} = \begin{bmatrix} f_i\rho u_x \\ \rho u_x^2 + p \\ \rho u_y u_x \\ \rho u_z u_x \\ \rho E u_x + p u_x \end{bmatrix} \hat{i} + \begin{bmatrix} f_i\rho u_y \\ \rho u_x u_y \\ \rho u_y^2 + p \\ \rho u_z u_y \\ \rho E u_y + p u_y \end{bmatrix} \hat{j} + \begin{bmatrix} f_i\rho u_z \\ \rho u_x u_z \\ \rho u_y u_z \\ \rho u_z^2 + p \\ \rho E u_z + p u_z \end{bmatrix} \hat{k} \quad , \tag{3}$$

and the viscous flux vector is

$$\mathbf{F_v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \hat{i} + \begin{bmatrix} f_i \rho \nu_{y,i} \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ u_x\tau_{xy} + u_y\tau_{yy} + u_z\tau_{zy} - q_y \end{bmatrix} \hat{j} + \begin{bmatrix} f_i \rho \nu_{z,i} \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u_x\tau_{xz} + u_y\tau_{yz} + u_z\tau_{zz} - q_z \end{bmatrix} \hat{k}. \quad (4)$$

The source term is

$$\mathbf{Q} = \begin{bmatrix} \dot{\omega}_i \\ 0 \\ 0 \\ 0 \\ \phi \end{bmatrix} , \quad (5)$$

where $(i = 1, 2, ..., N_S)$. These equations specify the conservation of mass, three components of momentum, and total energy. The fluxes are separated into inviscid, $\mathbf{F_i}$ and viscous, $\mathbf{F_v}$ components. The above equations can be slightly modified to form the PNS equations for axisymmetric flow. These equations appear in Appendix A.

The final term, ϕ, in the source vector $\mathbf{Q}$ is a heat addition term which can be used in place of heat addition due to chemical reactions. It provides a simple and fast method of approximating the heat release due to combustion without the computational cost of finite rate chemical kinetics.

For a non-reacting gas, the total energy is defined by

$$E = e + \frac{1}{2}(u_x^2 + u_y^2 + u_z^2) \quad . \quad (6)$$

However, for reacting gas mixtures, the total energy includes the total formation enthalpy of the gases (see Eq. (25)). The formation enthalpy provides a mechanism for heat addition or absorption in chemical reactions and will be considered later (Section 2.2).

The viscous stresses are given by

$$\begin{aligned} \tau_{xx} &= 0 \\ \tau_{yy} &= \frac{2}{3}\,\mu\,(2\,\frac{\partial u_y}{\partial y} - \frac{\partial u_z}{\partial z}) \end{aligned}$$

$$
\begin{aligned}
\tau_{zz} &= \frac{2}{3}\,\mu\,(2\,\frac{\partial u_z}{\partial z} - \frac{\partial u_y}{\partial y}) \\
\tau_{xy} &= \mu\frac{\partial u_x}{\partial y} = \tau_{yx} \\
\tau_{xz} &= \mu\frac{\partial u_x}{\partial z} = \tau_{zx} \\
\tau_{yz} &= \mu\,(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}) = \tau_{zy} \quad , \qquad (7)
\end{aligned}
$$

where μ is the first coefficient of viscosity. This formulation of viscous stresses assumes negligible bulk viscosity since we are not concerned with the study of the structure of shock waves and absorption and attenuation of acoustic waves [15]. The viscous heat fluxes are

$$
\begin{aligned}
q_x &= \rho\sum_{i=1}^{N_S} h_i f_i \nu_{x,i} \quad , \\
q_y &= k\frac{\partial T}{\partial y} + \rho\sum_{i=1}^{N_S} h_i f_i \nu_{y,i} \quad , \\
q_z &= k\frac{\partial T}{\partial z} + \rho\sum_{i=1}^{N_S} h_i f_i \nu_{z,i} \quad . \qquad (8)
\end{aligned}
$$

Currently, *sm_3d*$^+$ convects the species without considering their diffusion (i.e. $\nu_{x,i} = \nu_{y,i} = \nu_{z,i} = 0$). The evaluation of the viscous transport coefficients, μ and k, depend on the specific gas model and are modified when using an algebraic turbulence model. The evaluation of these terms will be covered in Section 2.4.

The $\dot{\omega}_i$ terms in the source vector $\mathbf{Q}$, are the production rates of species i in volume V due to the chemical reactions taking place in the volume. The evaluation of these terms is given in Section 2.3.

2.1.2 Streamwise Pressure Gradient

When implementing a space-marching method for the PNS equations, it is necessary to either remove or modify the streamwise pressure gradient term in the streamwise flux vector. It has been shown [16], that an exact representation of the streamwise pressure gradient will cause acoustic waves to be propagated upstream through the subsonic region of the boundary layer. Signals propagating upstream in a space-marching formulation will cause exponentially growing solutions called "departure solutions".

Several methods exist for eliminating this problem and are compared by Lin & Rubin [17]. One of the most effective methods is that proposed by Vigneron[18].

A stability analysis of the PNS equations was performed by Vigneron and later extended by Davis *et. al* [19]. In these studies it was shown that only a fraction $(1-\varepsilon)$ of the streamwise pressure term in the streamwise momentum equation, is responsible for the upstream propagation of information. If this fraction is dropped, the eigenvalues of the PNS equations will remain real, even in the subsonic regions thus stabilising the space-marching procedure.

The remaining fraction, ε, is is called "Vigneron's coefficient" and is given as,

$$\begin{aligned} \varepsilon &= 1\ , & M_x \geq M_{\text{limit}} \quad , \\ \varepsilon &= \frac{\Phi\ \gamma\ M_x^2}{1+(\gamma-1)M_x^2}\ , & M_x < M_{\text{limit}} \quad , \end{aligned} \tag{9}$$

where the streamwise Mach number is denoted by M_x, Φ is a safety factor and the limiting Mach number is

$$M_{\text{limit}} = \sqrt{\frac{1.0}{1.0+\gamma(\Phi-1.0)}}\ . \tag{10}$$

Typically, Φ ranges from 0.75 for complex shock-boundary layer interactions to 1.0 for simple boundary layer flows where there is less uncertainty in determining the viscous stability limit (see Fig. 1).

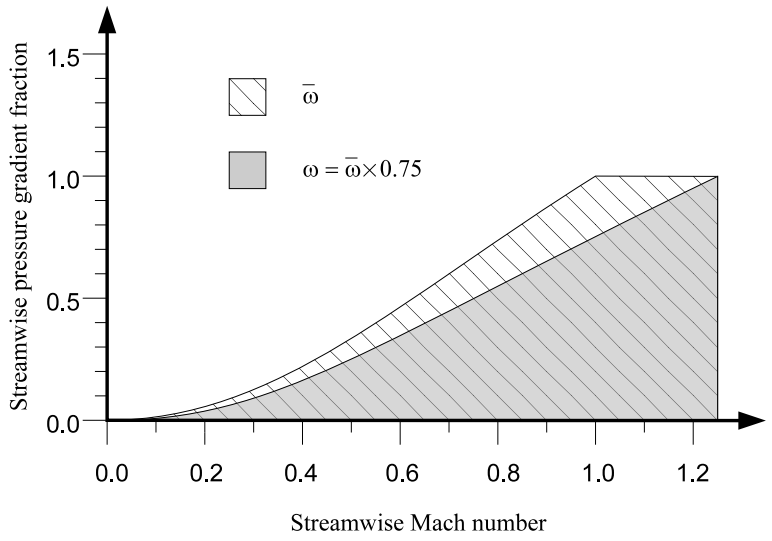

Figure 1: Fraction of the streamwise pressure gradient versus Mach number

The inviscid streamwise flux vector then becomes

$$\begin{bmatrix} f_1\rho u_x \\ f_2\rho u_x \\ \vdots \\ f_{N_S}\rho u_x \\ \rho u_x^2 + \varepsilon p \\ \rho u_y u_x \\ \rho u_z u_x \\ \rho E u_x + p u_x \end{bmatrix} \hat{i} \tag{11}$$

Even though this approach only retains as much of the streamwise pressure gradient as a stability analysis permits, it is sometimes necessary to set the pressure gradient to 0 for the first few marching steps (that is, set ε to 0) [20, 14]. The presence of a pressure gradient during the first few marching steps can cause solutions to become unstable. The number of steps that require the pressure gradient to be set to 0 depends on the step size and problem. A trial and error method is typically used to obtain the correct number of steps, however, 7 steps usually works well.

2.2 Thermodynamic Models

The assumption of a calorically perfect gas in the original version of the solver, limited the range of gas flows that could be modeled. The assumption begins to fail for air at about 600 K where the specific heats begin to increase from near constant values. To extend the range of the thermodynamic model, we start with the assumption that we have a thermally imperfect gas with

$$p = \rho \tilde{R} T \quad . \tag{12}$$

For a mixture of species, Dalton's Law holds whereby the mixture pressure is the summation of partial pressures. This leads to the function of pressure

$$p = \sum_{i=1}^{N_S} \rho_i R_i T = \rho \tilde{R} T \quad , \tag{13}$$

where the density of the mixture is

$$\rho = \sum_{i=1}^{N_S} \rho_i \quad , \tag{14}$$

and the mixture gas constant, $\tilde{R}$ is defined as

$$\tilde{R} = \sum_{i=1}^{N_S} \frac{\rho_i}{\rho} R_i = \sum_{i=1}^{N_S} f_i R_i \quad . \tag{15}$$

In the original version of the code, it was assumed that the gas could be classed as a calorically perfect gas with constant specific heats C_p and C_v. It followed that the ratio of specific heats $\gamma = C_p/C_v$, was constant, and the enthalpy and internal energy where linear functions of temperature. For a calorically perfect gas, we used

$$h = C_p T \quad , \quad e = C_v T \quad \text{and} \quad p = \rho(\gamma - 1)e \quad . \tag{16}$$

2.2.1 Thermodynamic Equilibrium Model

For single species and non-reacting gas mixtures in thermodynamic equilibrium, the sensible energy can be derived as a function of static temperature. Solutions of these functions for various species can be found in tables where thermodynamic properties are listed against temperature [21, 22]. For computational applications it is common to fit a polynomial equation to the tabulated specific heat data and integrate this equation to get the thermodynamic quantities h and s. The polynomial curve fits used for *sm_3d*$^+$ are those found in the NASP Technical Memorandum 1107 [23]. All the predominant species involved in hydrogen combustion in air up to a temperature of 6000 K are presented.

The polynomial used for the molar C_p^o is expressed as,

$$C_p^o/R^o = a_1 T^{-2} + a_2 T^{-1} + a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4 \quad , \tag{17}$$

where the coefficients $a_1 ... a_7$ are the polynomial coefficients specific to each species. Integration of this expression for C_p^o will give the value of molar enthalpy for the particular species at temperature T as

$$H^o = \int_{T_R}^{T} C_p^o dT + D \quad , \tag{18}$$

where D is the integration constant that sets enthalpy to zero at the reference

temperature T_R. The substitution of Eq. (17) into Eq. (18) gives,

$$H^o/R^o = -a_1T^{-1} + a_2 lnT + a_3T + a_4T^2/2 + a_5T^3/3 + a_6T^4/4 + a_7T^5/5 + a_9 \quad . \tag{19}$$

The curve fits listed in reference [23] are referenced to a temperature of 298.15 K such that the integration constant a_9R^o in Eq. (19) is equivalent to D plus the heat of formation at the standard reference temperature of 298.15 K. This poses a problem computationally since we require the sensible energy of the gas to always maintain a positive value for all temperatures. To maintain this condition, the expression for enthalpy was referenced to 0 K by deducting the heat of formation at 298.15 K and adding the difference in enthalpy between 0 K and 298.15 K. In this form, all the species sensible enthalpies and thus sensible internal energies, will be 0 at a temperature of 0 K. The formation enthalpy at 0 K is not added here since it is accounted for later in the equation for total energy Eq. (25).

If the molar specific heat at constant pressure in Eq. (17) is divided by T and then integrated from temperature T_R to T, the difference in entropy between these two temperatures is given as

$$S^o = \int_{T_R}^{T} \frac{C_p^o}{T} dT + D \quad . \tag{20}$$

The corresponding polynomial function is,

$$S^o/R^o = -a_1T^{-2}/2 - a_2T^{-1} + a_3 lnT + a_4T + a_5T^2/2 + a_6T^3/3 + a_7T^4/4 + a_{10} \tag{21}$$

where this function evaluates to the entropy difference from a reference state of 0 K using the polynomial coefficients from reference [23].

The Gibbs free energy,

$$G^o = H^o - TS^o \quad , \tag{22}$$

is used to calculate the equilibrium constant for a particular reaction as shown later in Eq. (39).

The molar internal energy for each species can also be calculated using the equation

$$E^o = H^o - R^oT \quad . \tag{23}$$

For a gas mixture, the specific sensible internal energy in J/kg can be computed

from the individual species molar values at a given temperature from the equation

$$e = \sum_{i=1}^{N_S} f_i \frac{E_i^o}{M_i} \quad , \tag{24}$$

where f_i is the mass fraction of species i and is equivalent to ρ_i/ρ.

The total energy is given as

$$E = e + \frac{1}{2}(u_x^2 + u_y^2 + u_z^2) + \sum_{i=1}^{N_S} \frac{(\Delta H_f^o)_{T_R,i}}{M_i} f_i \quad , \tag{25}$$

where $(\Delta H_f^o)_{T_R,i}$ is the molar heat of formation of species i at the reference temperature $T_R = 0$ K. This term provides the energy released or absorbed from chemical reactions. In the governing Eq. (1), a flux of formation enthalpy across the boundary is calculated, which provides the heat of reaction in that cell.

The other common method for accounting for the heat of reaction is to use a value of the heat of reaction for a particular reaction listed in the reaction rate data. The energy added or consumed by the reaction is then simply determined by multiplying the heat of reaction term by the reaction rate. These quantities are then summed over all reactions and added to the source term ϕ in the governing equations. The method of calculating the chemical energy from heats of formation is used in *sm_3d*$^+$ because heat of reaction data is not as consistent between data sets.

In Eq. (25), the total energy is written as a function of temperature, flow speed and species mass fractions. However, in the solution procedure, it is necessary to calculate the temperature, given the total energy, flow speed and mass fractions. Since the energy equation is a polynomial and can not be rearranged to give a simple expression for the temperature, an iterative secant method is used to find the temperature to within a set tolerance. The initial guess is obtained from a look up table of energies and corresponding temperatures for the given species composition. The other properties such as pressure and sound speed are then "backed out" using the equation of state, which is a function of temperature and density as

$$p = p(\rho, T) \quad . \tag{26}$$

Dalton's Law can be applied whereby the mixture pressure is the summation of the

partial pressures. This leads to the function of pressure as

$$p = \sum_{i=1}^{N_S} \rho_i R_i T = \rho \tilde{R} T \quad . \tag{27}$$

Other thermodynamic quantities for a multi-species gas are worked out in a similar fashion.

$$C_p = \sum_{i=1}^{N_S} f_i \frac{C_{p,i}^o}{M_i} \tag{28}$$

$$\tilde{R} = \sum_{i=1}^{N_S} f_i \frac{R_i^o}{M_i} \tag{29}$$

The frozen speed of sound is used for all computations in the current code and is defined as

$$a^2 = \gamma \frac{p}{\rho} = \left(\frac{C_p}{C_p - \tilde{R}} \right) \tilde{R} T \quad , \tag{30}$$

where C_p and $\tilde{R}$ are the mixture quantities [24]. Note that the γ used is the ratio of specific heats corresponding to the internal energy being in thermodynamic equilibrium.

2.2.2 Fast Equilibrium Models

In addition to the general equilibrium model presented above, three extra models have been included in the code to model nitrogen and air in thermodynamic equilibrium. These models have been specifically written to be computationally efficient.

The first nitrogen model is for non-dissociating nitrogen in vibrational equilibrium. This model does not consider dissociation, thus, should be limited to temperatures where nitrogen dissociation does not occur. The specific internal energy of nitrogen is given by a simple relation that sums the components of translational and vibrational energy as

$$e_{N_2} = \left(2.5 + \frac{\frac{\theta_{\text{vib}}}{T}}{\exp\left(\frac{\theta_{\text{vib}}}{T}\right) - 1.0} \right) R_{N_2} T \quad , \tag{31}$$

where θ_{vib} is the characteristic temperature for the vibrational mode of nitrogen.

The second nitrogen model is used to calculate the mixture internal energy of dissociating nitrogen in thermodynamic equilibrium up to temperatures of 15000 K [25]. Given a temperature and density, it works out the equilibrium mass fractions of

N and N_2, and then the mixture internal energy using a series of curve fits for several temperature ranges. This model has limited applicability at large temperatures because it does not consider ionization.

Lastly, an equilibrium air model is included which models non-dissociating air in thermodynamic equilibrium [26]. The model is based on a set of curve fits for oxygen and nitrogen. Care should be taken when using this model to limit the cases to non-dissociating flows.

An iterative solver is included in each model to solve for temperature given the specific internal energy and density.

2.3 Chemistry Model

The species production terms $\dot{\omega}_i$ in Eq. (1) represent the production rate of species i in a particular volume due to the chemical reactions taking place in that cell. The source of the reaction rate data is nominally the NASP Technical Memorandum 1107[23]. Currently, there are reaction schemes for (i) hydrogen and oxygen [27], (ii) hydrogen and air, (iii) carbon dioxide and oxygen [28], and (iv) air. Each of these schemes is made up of reactions describing paths for dissociation and recombination of the chemical species present in the test flow. Each reaction has the general form,

$$\sum_{i=1}^{N_S} \alpha'_{ij} Z_i \underset{\longleftarrow}{\longrightarrow} \sum_{i=1}^{N_S} \alpha''_{ij} Z_i \qquad (j = 1, 2, ..., N_R) \quad , \tag{32}$$

where Z_i are the chemical symbols and α', α'' are the reactant and product stoichiometric coefficients respectively. Species production rate expressions are determined by summing the contributions from each relevant reaction route to obtain the total rate of change of each species $\dot{\omega}_i$. Each path is assumed to be governed by a "law-of-mass-action" expression in which the rate constants can be determined from a temperature dependent Arrhenius law. The net rate of change of concentration of species i by reaction j is given by

$$(\dot{C}_i)_j = (\alpha''_{ij} - \alpha'_{ij}) \left(k_{f,j} \prod_{l=1}^{N_S} C_l^{\alpha'_{lj}} - k_{r,j} \prod_{l=1}^{N_S} C_l^{\alpha''_{lj}} \right) \quad , \tag{33}$$

where k_f, k_r are the forward and reverse reaction rates respectively and C are the species concentrations. The rate change in concentration of species i by N_R reactions

is then found by summing the contributions from each reaction which is written as

$$\dot{C}_i = \sum_{j=1}^{N_R} (\dot{C}_i)_j \quad . \tag{34}$$

Finally, the production rate of species i in $kg/(m^3.s)$, is found from

$$\dot{\omega}_i = \dot{C}_i M_i \quad , \tag{35}$$

where M is the molecular weight in kg/mol.

The forward reactions rates are computed from the modified Arrhenius law

$$k_{f_j} = A_j T^{N_j} \exp\left(\frac{-\theta_j}{T}\right) \quad . \tag{36}$$

for each reaction j, where A_j and N_j are constants for reaction j, and θ_j is the activation temperature. The activation temperature is equal to the activation energy divided by R^o. The reverse rate can be found given the forward rate and equilibrium constant K_j for each reaction j as

$$k_{rj} = \frac{k_{fj}}{K_j} \quad . \tag{37}$$

The equilibrium constant is a function of the difference between Gibbs free energy of the reactants and products and temperature as given by

$$K_j = \left(\frac{R^o T\ 10^6}{p_{\text{atm}}}\right)^{l_j} \exp\left(\frac{-\Delta G_j^o}{R^o T}\right) \quad . \tag{38}$$

In the above equation, l_j is equal to the integer sum of the stoichiometric coefficients of the reactants minus the sum of coefficients of the products for reaction j. The standard Gibbs free energy difference for the reaction j is,

$$\Delta G_j^o = \sum_{i=1}^{N_S} \alpha''_{ij} G_i^o - \sum_{i=1}^{N_S} \alpha'_{ij} G_i^o \quad . \tag{39}$$

Table 1: Sutherland's viscosity coefficients from [30], [31]*

Gas	A	B	T range for 2% error
Hydrogen, H_2	6.899×10^{-7}	97	220-1100
Helium*, He	1.461×10^{-6}	79	
Carbon Monoxide, CO	1.503×10^{-6}	136	130-1500
Nitrogen, N_2	1.400×10^{-6}	107	100-1500
Air	1.461×10^{-6}	111	170-1900
Oxygen, O_2	1.753×10^{-6}	139	190-2000
Argon, Ar	1.964×10^{-6}	144	120-1500
Carbon Dioxide, CO_2	1.503×10^{-6}	222	190-1700

2.4 Transport Coefficients

The coefficients of viscosity and thermal conductivity contained in the viscous terms of Eq. (1) are composed of laminar and turbulent components.

$$\begin{aligned} \mu &= \mu_{\text{lam}} + \mu_{\text{turb}} \\ \frac{k}{C_p} &= \frac{\mu_{\text{lam}}}{Pr} + \frac{\mu_{\text{turb}}}{Pr_{\text{turb}}} \end{aligned} \tag{40}$$

If the flow is completely laminar then the turbulent components are set to 0.

The laminar coefficient of viscosity is calculated using Sutherland's formula for single species gases due to its simplicity [29]. The formula is written as

$$\mu_{\text{lam}} = A\frac{T^{3/2}}{T+B} \tag{41}$$

where A and B are constants dependent on the gas. These constants are listed for various gases in Table 1 with a temperature range of applicability. The Sutherland formula is valid for single component gases, however, air is included because its two principal components, oxygen and nitrogen, are nearly identical diatomic molecules. For gas mixtures that are composed of dissimilar components, the mixture viscosity varies strongly with species concentration.

For multi-species gases, values of viscosity for pure components are combined in a mixing rule. The approximation of Wilke [32] is used for mixture viscosity

$$\mu_{\text{lam}} = \sum_{i=1}^{N_s} \frac{X_i \mu_i}{\sum_{j=1}^{N_s} X_j \phi_{ij}} \quad , \tag{42}$$

where X_i is the mole fraction of species i, and ϕ_{ij} is defined as

$$\phi_{ij} = \frac{1}{\sqrt{8}}\left(1+\frac{M_i}{M_j}\right)^{-\frac{1}{2}}\left[1+\sqrt{\frac{\mu_i}{\mu_j}}\left(\frac{M_j}{M_i}\right)^{\frac{1}{4}}\right]^2 \quad . \tag{43}$$

The mole fractions are computed from the species densities and molecular weights.

$$X_i = \frac{\rho_i/M_i}{\sum_{j=1}^{N_S}\rho_j/M_j} \tag{44}$$

The coefficient of viscosity for each species i in a mixture is given by an equation derived by Hirschfelder *et al.* [33], rather than the Sutherland's equation. This is due to the large availability of constants for different species applicable to the former equation. The equation proposed by Hirschfelder *et al.* [33] is

$$\mu_i = 2.6693\times 10^{-6}\frac{\sqrt{M_i T}}{\sigma_i^2\Omega_i^{(2,2)}} \quad , \tag{45}$$

where σ_i is the collision diameter in ($\mathring{\text{A}}$), and $\Omega^{(2,2)}$ is the elastic collision integral. The collision integral for species i can be expressed as an empirical function [34] of the reduced temperature T^*

$$\Omega_i^{(2,2)} = [A(T^*)^{-B}] + C[\exp(-DT^*)] + E[\exp(-FT^*)] \quad , \tag{46}$$

where the reduced temperature is

$$T^* = \frac{k_B T}{\epsilon_i} \quad , \tag{47}$$

and the constants are

$$\begin{aligned} &A = 1.16145, \quad B = 0.14874, \quad C = 0.52487, \\ &D = 0.77320, \quad E = 2.16178, \quad \text{and} \quad F = 2.43787. \end{aligned} \tag{48}$$

Equation (46) is applicable over the range $0.3 \leq T^* \leq 100$ with an average deviation of only 0.064 percent. For species that take part in hydrogen combustion with air, this reduced temperature range translates to a real temperature range of approximately 87 K $\leq T \leq$ 3700 K. This range is based on the heaviest and lightest substances in the range of species present.

The collision diameter σ_i and the characteristic energy ϵ_i are molecular constants found in the Lennard-Jones 6-12 intermolecular potential function. Tabulated potential parameter data (ϵ_i/k_B and σ_i) for various species are found in [35]. The potential parameters for the species that are involved in air-hydrogen combustion are listed in Appendix B.

The thermal conductivity coefficient k_{lam} is calculated for a single species gas using the Reynold's analogy assuming a constant Prandtl number.

$$k_{\text{lam}} = \frac{\mu C_p}{Pr} \tag{49}$$

For multi-species gas mixtures, the same equation is used where a constant Prandtl number is assumed and μ_{lam} and C_p are the mixture values.

2.5 Turbulent Viscosity and Thermal Conductivity

For the study of turbulent hypersonic flow, many investigators employ algebraic turbulence models rather than one-equation and two-equation models. Algebraic turbulence models are attractive because of their efficiency, simplicity, and robustness. Also, more sophisticated models require larger storage space and greater computational time. For attached flows, some studies [6, 7, 36] have shown that there is often little improvement using a more complicated model.

Two popular algebraic models have been developed by Cebeci & Smith [37] (CS) and Baldwin & Lomax [5] (BL). An evaluation of these models for supersonic and hypersonic flows is presented by Shirazi & Truman [20]. This study shows that the differences in the two models are due to the near-wall damping term used, the outer eddy-viscosity formulation, and the effects of outer-layer intermittency. They are both two layer eddy-viscosity model formulations, primarily differing in the choice of length and velocity scales in the outer layer. The CS turbulence model uses the displacement thickness as the length scale, and the BL turbulence model uses a length scale based on the vorticity distribution. For complex separated flows it is not a simple matter to determine the displacement thickness, so for this reason the BL model is the model used in *sm_3d*$^+$.

The turbulent Prandtl number was assumed to be 0.9 for both reacting and non-reacting systems. The turbulent viscosity coefficient μ_{turb} is given by

$$\mu_{\text{turb}} = \begin{cases} (\mu_{\text{turb}})_{\text{inner}} & \text{if} \quad d \leq d_c \\ (\mu_{\text{turb}})_{\text{outer}} & \text{if} \quad d > d_c \end{cases} , \tag{50}$$

where d is the local distance measured normal to the body surface, and d_c is the smallest value of d at which the values from the inner and outer region formulae are equal (see Fig. 2).

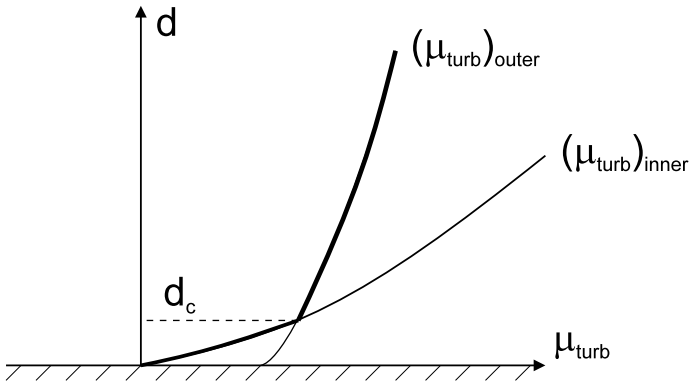

Figure 2: Switching from inner to outer value of eddy viscosity

For corner flows the distance d is replaced with, d^*, and is given by an equation for a "modified distance" as suggested in reference [38]

$$d^* = \frac{2yz}{(y+z) + \sqrt{y^2+z^2}} \quad , \tag{51}$$

which is used to account for the size of turbulent eddy or the turbulent mixing length near the axial corner under the influence of both the y and z walls.

The inner layer turbulent viscosity is given by,

$$(\mu_{\text{turb}})_{\text{inner}} = \rho l^2 |\omega| \quad , \tag{52}$$

where

$$l = kd[1 - \exp(-d^+/A^+)] \quad . \tag{53}$$

The bracketed term in the above equation is the van Driest damping. This is the original damping term that appeared in the paper by Baldwin & Lomax [5], which is for incompressible flow. The damping term can be modified to include compressibility effects which can be quite prevalent in the inner layer of hypersonic flows. The van Driest damping term for compressible flow as presented by Cebeci & Smith [37] is,

$$D = 1 - \exp\left[\frac{-d^+(\rho/\rho_{\text{wall}})^{1/2}(\mu_{\text{wall}}/\mu)}{A^+}\right] \quad . \tag{54}$$

This damping term was used in the present solver since it gives appreciably better results for turbulent compressible flow [20]. Shirazi & Truman [20] note that the

difference between the CS model and the BL model is almost entirely due to the form of damping used.

For corner flows, calculation of the damping term is based on the proximity to neighbouring walls. Referring to Fig. 3, the damping term in regions 1 and 2 is evaluated from the wall $y = 0$, and in regions 3 and 4, the wall $z = 0$. The search for $F_{\max}$ and its corresponding $d_{\max}$ proceeds outward from the wall, either from $y = 0$ for region 4, or from $z = 0$ for region 1. The values of $F_{\max}$ in regions 2 and 3 are constants, equal to the value of $F_{\max}$ at m and n, respectively.

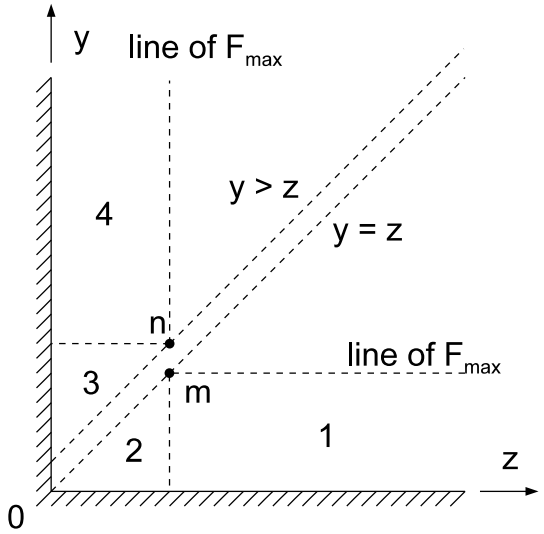

Figure 3: Division of computational space for turbulent calculations

The law of the wall coordinate, d^+ is given by

$$d^+ = \frac{\sqrt{\rho_{\text{wall}}\tau_{\text{wall}}}\ d}{\mu_{\text{wall}}} \tag{55}$$

and the PNS vorticity in three-dimensions is equal to,

$$\omega = \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial z}\right)^2} \quad . \tag{56}$$

For the outer layer,

$$(\mu_{\text{turb}})_{\text{outer}} = KC_{\text{cp}}\rho F_{\text{wake}} F_{\text{Kleb}}(d) \quad , \tag{57}$$

where F_{wake} is the smaller of

$$F_{\text{wake}} = \begin{cases} d_{\max} F_{\max} \\ C_{\text{wk}} d_{\max} u_{\text{diff}}^2 / F_{\max} \end{cases} \quad . \tag{58}$$

The term $d_{\max}$ is the value of d corresponding to the maximum value of F, $F_{\max}$, where

$$F(d) = d|\omega|D \tag{59}$$

and F_{Kleb} is the Klebanoff intermittency factor given by

$$F_{\text{Kleb}} = [1 + 5.5(nC_{\text{Kleb}}/d_{\max})^6]^{-1} \quad . \tag{60}$$

The quantity u_{diff} is the difference between maximum and minimum total velocity along a line of ascending d which is simply the maximum velocity for boundary layer flows

$$u_{\text{diff}} = (\sqrt{u^2 + v^2 + w^2})_{\max} \quad . \tag{61}$$

Transition to turbulence can be simulated by setting the computed value of turbulent viscosity, μ_{turb}, equal to zero until the maximum in the profile normal to the wall is less than a specified value, that is,

$$\mu_{\text{turb}} = 0.0 \quad \text{if} \quad (\mu_{\text{turb}})_{\text{max. in profile}} < C_{\text{mutm}}\, \mu_\infty \quad . \tag{62}$$

The values of the constants appearing in Eqs. (53) to (62) are listed in [5] as

$$\begin{aligned} A^+ &= 26, & C_{\text{wk}} &= 0.25, \\ C_{\text{cp}} &= 1.6, & k &= 0.4, \\ C_{\text{Kleb}} &= 0.3, & K &= 0.0168, \\ Pr_{\text{turb}} &= 0.9, & C_{\text{mutm}} &= 14 \quad . \end{aligned} \tag{63}$$

A modification proposed by Degani & Schiff [39] to more accurately determine the length scale in strong vortical flows was implemented. Strong vortical flows exist when there are large separations of boundary layers. In these vortical flows, there exists two or more maximum values of the function $F(d)$ (see Eq. (59)) with the outermost maximum being the largest. The selection of this maximum for $F_{\max}$ will result in an outer eddy viscosity that is as much as two orders of magnitude too high. To avoid this, Degani & Schiff [39] selected the first maximum closest to the

wall, where the value of $F(d)$ drops to less than 90% of the local maximum as d increases (see Fig. 4).

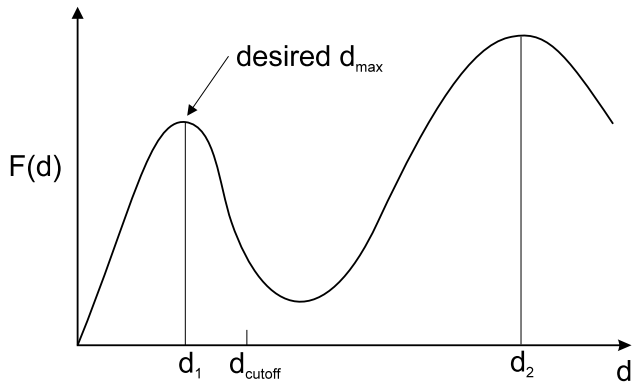

Figure 4: Selection of correct $F(d)$ in separated flow

A further modification was proposed by Degani & Schiff [39] for the determination of $F_{\max}$ and $d_{\max}$ at separation points. At the separation point, the $F_{\max}$ value can rapidly increase due to the merging of recirculating flow from vortex structures and the attached boundary layer. This merging produces high values of $F_{\max}$ at the outer regions of the merging and blends the inner peak in $F_{\max}$ such that it can not be determined. On each ray of d (except on symmetry planes), a cutoff distance is specified in terms of $d_{\max}$ from the previous ray

$$d_{\text{cutoff}}(\zeta) = 1.5\ d_{\max}(\zeta - \Delta\zeta) \quad , \tag{64}$$

where ζ is the coordinate along the wall. If no peak in $F(d)$ is found along a ray for $d \leq d_{\text{cutoff}}$, the values of $F_{\max}$ and $d_{\max}$ are taken from those found on the previous ray.

2.6 Finite-Volume Discretization

The integral in Eq. (1) is evaluated over the computational domain in finite volumes. These volumes are hexahedral cells as shown in Fig. 5. If the cell volumes are denoted V and the area of each face as dS, then the integral Eq. (1) can be written as

$$V\frac{\partial < \mathbf{U} >}{\partial t} + \oint_S \mathbf{F}\cdot\hat{\mathbf{n}}_{\mathbf{x}}dS + \oint_S \mathbf{F}\cdot\hat{\mathbf{n}}_{\mathbf{y}}dS + \oint_S \mathbf{F}\cdot\hat{\mathbf{n}}_{\mathbf{z}}dS = \int_V \mathbf{Q}\mathrm{d}V, \tag{65}$$

where $< \mathbf{U} >$ are the cell averaged values of the conserved variables stored for each cell at the centroid. Applying this equation to a six-sided cell and using average fluxes at the midpoints of each interface, the semi-discrete approximation to the

governing equations becomes,

$$\frac{d<\mathbf{U}>}{dt} + \frac{1}{V}\Big[\overline{\mathbf{F}}_{ix-\frac{1}{2}}\cdot\hat{\mathbf{n}}_{ix-\frac{1}{2}}\,dS_{ix-\frac{1}{2}} + \overline{\mathbf{F}}_{ix+\frac{1}{2}}\cdot\hat{\mathbf{n}}_{ix+\frac{1}{2}}\,dS_{ix+\frac{1}{2}} + \overline{\mathbf{F}}_{iy-\frac{1}{2}}\cdot\hat{\mathbf{n}}_{iy-\frac{1}{2}}\,dS_{iy-\frac{1}{2}} + \overline{\mathbf{F}}_{iy+\frac{1}{2}}\cdot\hat{\mathbf{n}}_{iy+\frac{1}{2}}\,dS_{iy+\frac{1}{2}} + \overline{\mathbf{F}}_{iz-\frac{1}{2}}\cdot\hat{\mathbf{n}}_{iz-\frac{1}{2}}\,dS_{iz-\frac{1}{2}} + \overline{\mathbf{F}}_{iz+\frac{1}{2}}\cdot\hat{\mathbf{n}}_{iz+\frac{1}{2}}\,dS_{iz+\frac{1}{2}}\Big] = <\mathbf{Q}> \quad . \tag{66}$$

To evaluate the terms is this equation, the cell geometry is defined by the cell face normals, areas, and volumes. The Cartesian coordinate system (x, y, z) is used to define the positions of the cell vertices along with the normalised (ξ, η, ζ)-coordinate system (see Fig. 5). The marching direction ξ is approximately aligned with the supersonic flow direction. The dashed vertices (a', b', c', d') define the upstream face of the cell while the undashed vertices define the corresponding downstream face.

Cell faces that have all of their vertices in a single ξ = constant plane are called $XFaces$ within the code. $YFaces$ and $ZFaces$ have all of their vertices in η = *constant* and ζ = *constant* planes respectively. The cycle *abcd* is chosen to correspond to an $XFace$ having its unit normal positive in the ξ (streamwise) direction (see Fig. 6). To get the geometric properties of an $XFace$, four *edge*

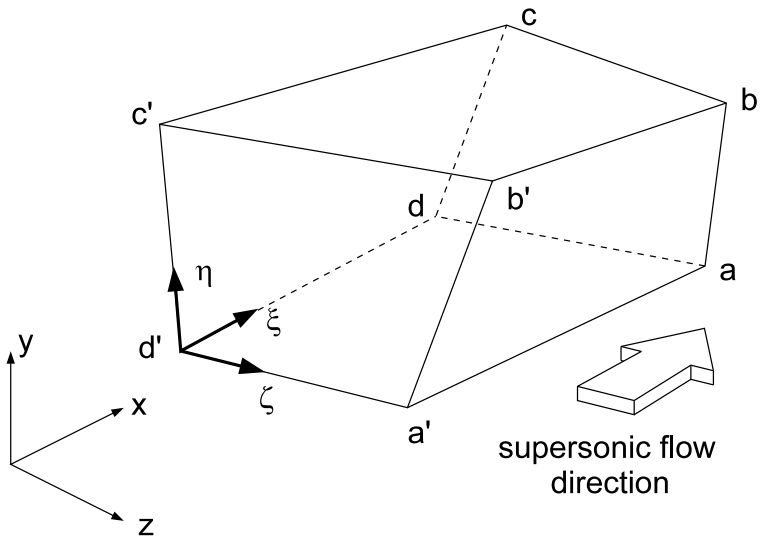

Figure 5: Finite-volume cell with coordinate directions and vertex labels.

vectors are defined as

$$\begin{aligned}
\overline{ad} &= (x_d - x_a)\hat{i} + (y_d - y_a)\hat{j} + (z_d - z_a)\hat{k} \quad , \\
\overline{ab} &= (x_b - x_a)\hat{i} + (y_b - y_a)\hat{j} + (z_b - z_a)\hat{k} \quad , \\
\overline{cb} &= (x_b - x_c)\hat{i} + (y_b - y_c)\hat{j} + (z_b - z_c)\hat{k} \quad , \\
\overline{cd} &= (x_d - x_c)\hat{i} + (y_d - y_c)\hat{j} + (z_d - z_c)\hat{k} \quad .
\end{aligned} \tag{67}$$

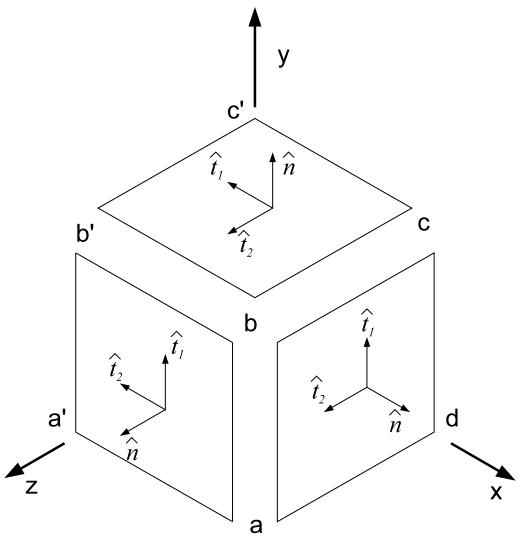

Figure 6: $XFace$, $YFace$ and $ZFace$ vertices with unit normal and tangent vectors.

The face is split into two triangular facets and the *area vectors*

$$\begin{aligned} \overline{A}_1 &= \overline{ad} \times \overline{ab} \\ \overline{A}_2 &= \overline{cb} \times \overline{cd} \end{aligned} \tag{68}$$

are computed, with the *average* area vector of the face being defined as

$$\overline{A} = \frac{1}{2}\left(\overline{A}_1 + \overline{A}_2\right) \quad . \tag{69}$$

From this, the nominal surface area of the face is

$$dS_{ix+\frac{1}{2}} = |\overline{A}| \quad , \tag{70}$$

and the corresponding unit normal is

$$\hat{n}_{ix+\frac{1}{2}} = \frac{\overline{A}}{|\overline{A}|} \quad . \tag{71}$$

Two tangent vectors can then be defined as

$$\hat{t}_1 = \frac{\hat{n} \times \overline{ad}}{|\hat{n} \times \overline{ad}|} \quad , \quad \hat{t}_2 = \hat{n} \times \hat{t}_1 \quad . \tag{72}$$

Similar quantities can be defined for the $YFaces$ using vertices $cc'b'b$ and the $ZFaces$ using vertices $abb'a'$ (see Fig. 6).

The volume of each cell is computed by summing the volume of the six tetrahe-

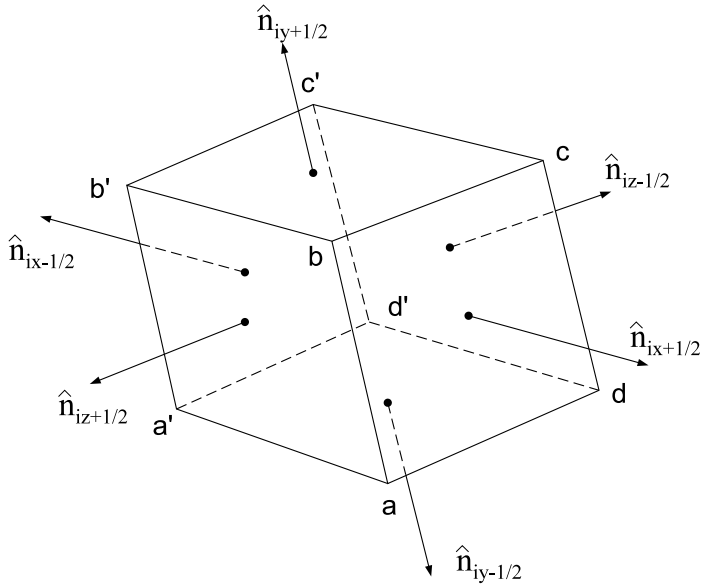

Figure 7: Finite-volume cell showing the unit normal at each interface.

drons that are contained within the cell (see Fig. 8). The volume of each tetrahedron is computed using a vector triple product

$$V_{it} = \frac{1}{6}\,\overline{d}\cdot\left(\overline{a}\times\overline{b}\right) \quad , \tag{73}$$

and the centroid is evaluated as an average of the vertex coordinates

$$\overline{P}_{it} = \frac{1}{4}\left(\overline{P}_1 + \overline{P}_2 + \overline{P}_3 + \overline{P}_4\right) \quad , \tag{74}$$

where $\overline{P}_1$ to $\overline{P}_4$ are the position vectors of the vertices. The original (hexahedral) cell volume is then evaluated as

$$V = \sum_{it=1}^{6} V_{it} \quad , \tag{75}$$

and the centroid is defined as

$$\overline{P} = \frac{1}{V}\sum_{it=1}^{6} \overline{P}_{it} V_{it} \quad . \tag{76}$$

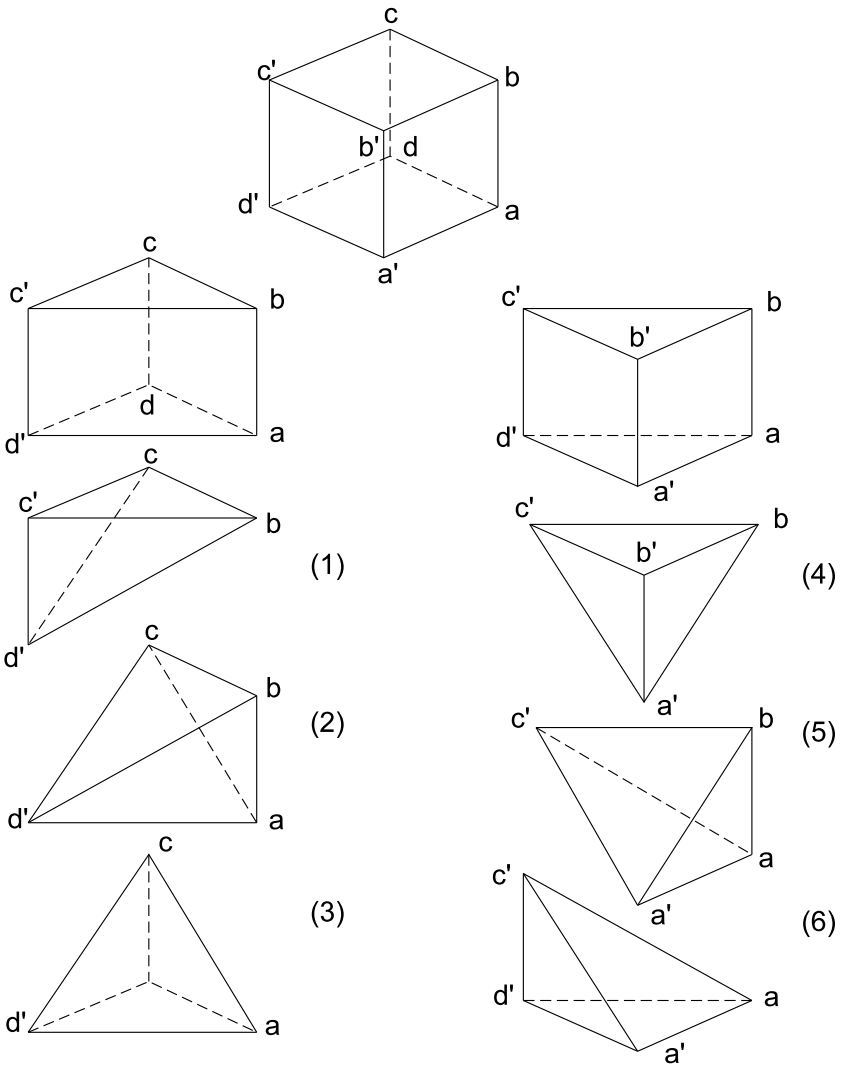

Figure 8: Dissection of the hexahedral cell into six tetrahedrons. The faces of the original cell are cross-hatched.

2.7 Flow Field Reconstruction & Inviscid Flux Calculation

To solve the semi-discrete form of the governing Eq. (66), the fluxes across the cell faces need to be evaluated. The evaluation of the inviscid fluxes will be covered in this section, followed by the viscous fluxes as described in the proceeding section.

Inviscid fluxes on the cell faces are calculated after reconstructing a flow state description from the cell-averaged values. The reconstruction process converts known cell-averaged flow properties ($\rho, u_x, u_y, u_z, e, p$ and f_i), into point-wise data located at the middle of each bounding face of the control volume. The accuracy of reconstructing the primitive-variable field at the cell faces determines the spatial accuracy of the solution [40].

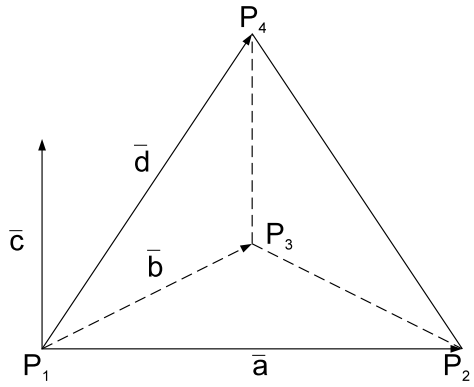

Figure 9: Tetrahedron with edge vectors labelled.

2.7.1 *XFace* Inviscid Fluxes

The reconstruction of the flow states on the downwind $XFace$ cell-interfaces (denoted with the subscript $ix + \frac{1}{2}$), is an extrapolation from the cell-averaged values of the previous two cells upwind ($i-1$ and $i-2$ respectively) and the current cell (i). The extrapolation is done separately for each of the flow variables and species mass fractions, then the estimated flow field properties are combined to form the cell-interface inviscid fluxes. The fluxes for the upwind $XFace_{ix-\frac{1}{2}}$ are simply the values from the steady-state solution of the previous upwind cell. This characteristic-based (upwind) technique is accurate enough for calculating fluxes on $XFaces$, because the core flow in the x-direction is generally supersonic and thus the conservation equations are hyperbolic in nature.

The extrapolation technique used is the Monotone Upstream-centred Scheme for Conservation Laws approach, or MUSCL [40], which is spatially second order accurate. As an example, the downwind cell-interface value for density would be,

$$\rho_{int} = \rho_i + \frac{1}{4}[(1-\kappa)\overline{(\Delta-)}_i + (1+\kappa)\overline{(\Delta+)}_i] \quad , \tag{77}$$

where

$$\begin{aligned} \overline{(\Delta-)}_i &= \text{MINMOD}(\Delta_i, \beta\Delta_{i+1}) \quad , \\ \overline{(\Delta+)}_i &= \text{MINMOD}(\beta\Delta_i, \Delta_{i+1}) \quad , \end{aligned} \tag{78}$$

and

$$\Delta_{i-1} = \rho_{i-1} - \rho_{i-2} \quad , \tag{79}$$

$$\Delta_i = \rho_i - \rho_{i-1} \quad , \tag{80}$$

$$\Delta_{i+1} = \Delta_i + \Delta_i - (\Delta_{i-1}) \quad . \tag{81}$$

The current cell-averaged value is ρ_i, the next upwind value is ρ_{i-1} and the next value is ρ_{i-2} (see Fig. 10). The MUSCL scheme used here [41] is slightly modified

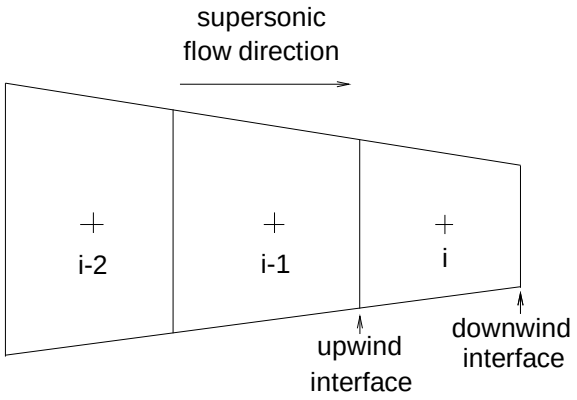

Figure 10: Cells used for upwind extrapolation of interface fluxes on the downwind interface $XFace_{ix+\frac{1}{2}}$.

from the original in [40]. The downstream gradient Δ_{i+1} is an approximation using the two upstream gradients. This limits the accuracy of the scheme to second order. The MUSCL parameter κ is either set to -1 for a fully upwind scheme or $1/3$ for an upwind-biased scheme. The fully upwind scheme has less of an upstream influence and appears to be more suited to flows with large subsonic regions (see the turbulent ramp test case for an example). The compression parameter for the limiter is set to $\beta = 2$.

The minimum modulus (MINMOD) limiter function returns the argument with the minimum magnitude if both arguments have both the same sign and returns zero otherwise. The purpose of the limiter is to maintain stability and eliminate numerical oscillations in regions with large gradients such as shocks. The MINMOD limiter is computationally inexpensive, but it does not resolve contact discontinuities well and can cause limit cycles in the convergence process. Many other limiters are available and a good treatment of the different types is given in Sweby [42].

Once the fluxes are calculated for the $XFace$ cell-interfaces, the Vigneron's pressure splitting is applied. The fraction of the pressure that is dropped from the face streamwise momentum term, is determined from the cell-centre flow terms rather than the reconstructed face terms. This is done so that a nonphysical acceleration caused by a variation in ε across the cell does not result.

2.7.2 Inviscid Boundary Conditions

Before the cross-stream fluxes are determined, boundary conditions are applied around the duct walls by setting up a layer of ghost cells (two deep) along each boundary (see Fig. 11). The flow properties in the ghost cells are determined from

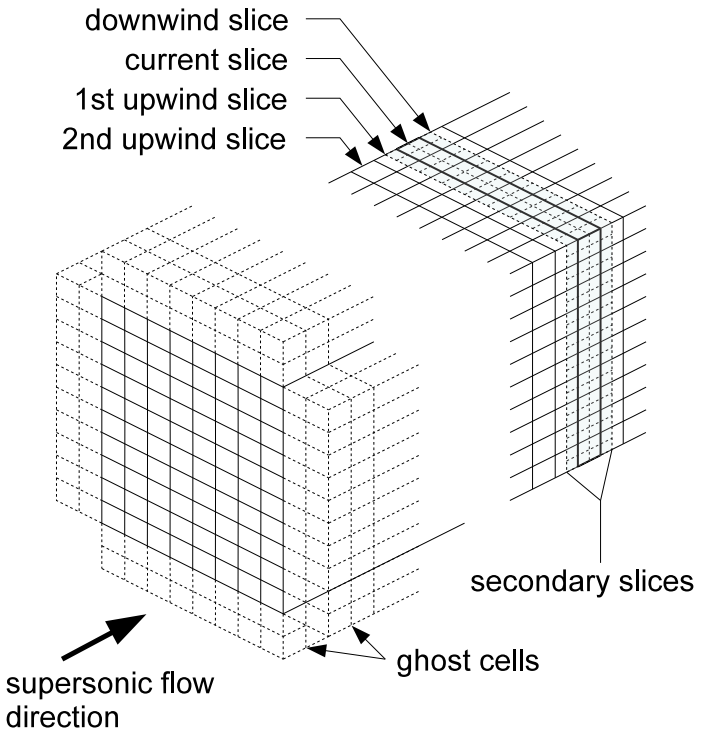

Figure 11: Ghost and secondary cells used for the calculation of boundary conditions and viscous derivatives .

the interior cells and/or the free stream conditions depending upon the imposed bounding condition. The boundary conditions are updated every iteration.

Each of the four walls bounding the flow domain can be set to a different condition. The four walls are denoted $North, South, East$ & $West$ which correspond to the top, bottom, right and left walls respectively looking from the inflow plane downstream (as long as the corners of the grid are defined in the order shown in Section 3). The boundary conditions that are available, and their associated codes, are:

1 : supersonic inflow condition

2 : supersonic outflow (not used)

3 : solid wall with inviscid (slip) tangency condition

4 : solid, no-slip, adiabatic wall

5 : solid, no-slip, fixed temperature wall

For conditions 3,4 and 5, the inviscid boundary condition is applied for the velocity vector by reflecting the component normal to the cell face. The first ghost cell to the wall contains the reflected velocity vector from the first interior cell. The second ghost cell contains the reflected velocity vector from the second interior cells. The wall normal is used to find the normal components of the velocity vectors. The temperature, internal energy, pressure, mass fractions and sound speed are all copied from the source cell. A supersonic inflow condition is applied by filling the ghost cells with the specified free stream quantities. The reconstruction procedure may now be applied (uniformly) across the entire slice irrespective of an interface lying on the domain boundary.

2.7.3 Cross-Stream Inviscid Fluxes

The calculation of the cross-stream interface fluxes is a little more involved than the calculation of the streamwise fluxes because we don't have a dominant (supersonic) flow direction. The cross-stream interfaces, $iy \pm \frac{1}{2}$ and $iz \pm \frac{1}{2}$, require a more robust flux calculation procedure so that oblique shocks within the flow can be captured without large oscillations in the flow properties. Here, a Godunov-type scheme is used which is based on MUSCL reconstruction of the flow properties on the "left" and "right" sides of the interface, followed by the application of an approximate Riemann flux calculator, or an AUSM flux calculator, to resolve differences at each of the interfaces. The details of these flux calculators will be covered in proceeding sections.

The generalised MUSCL reconstruction scheme [41] is again applied independently to each of the primary flow variables to give estimates of the flow properties on each side of the cell interfaces. This is very similar to the $XFaces$, however, an extrapolation version of MUSCL was used that gave estimates of the interface values directly and was only second-order accurate. For the $YFace$ and $ZFaces$, a third-order interpolation scheme is used which gives left and right states. As an example, the density estimates either side of the $iy + \frac{1}{2}$ interface are

$$\begin{aligned} \rho_L &= \rho_{iy,iz} + \frac{1}{4}\left[(1-\kappa)\overline{(\Delta-)}_{iy,iz} + (1+\kappa)\overline{(\Delta+)}_{iy,iz}\right] \quad , \\ \rho_R &= \rho_{iy+1,iz} - \frac{1}{4}\left[(1+\kappa)\overline{(\Delta-)}_{iy+1,iz} + (1-\kappa)\overline{(\Delta+)}_{iy+1,iz}\right] \quad , \end{aligned} \tag{82}$$

where

$$\begin{aligned} \overline{(\Delta-)}_{iy,iz} &= \text{MINMOD}(\Delta_{iy,iz}, \beta\Delta_{iy+1,iz}) \quad , \\ \overline{(\Delta+)}_{iy,iz} &= \text{MINMOD}(\beta\Delta_{iy,iz}, \Delta_{iy+1,iz}) \quad , \end{aligned} \tag{83}$$

and

$$\Delta_{iy,iz} = \rho_{iy,iz} - \rho_{iy-1,iz} \quad . \tag{84}$$

The compression parameter for the limiter is set to $\beta = 2$ and the spatial accuracy constant is set to $\kappa = 1/3$, giving third-order accurate interpolation. Note that the reconstruction is applied independently in the η and ζ directions.

In order to apply a one-dimensional flux calculation procedure, the local flow velocities of the interpolated left and right states are rotated into a local frame of reference. In this frame of reference the "local streamwise" direction is aligned with the unit normal for the particular cell interface. This is done by taking the dot product of the velocity with each of the unit vectors associated with the interface

$$\begin{aligned} u_n &= u_x n_x + u_y n_y + u_z n_z \quad , \\ u_{t1} &= u_x t_{1x} + u_y t_{1y} + u_z t_{1z} \quad , \\ u_{t2} &= u_x t_{2x} + u_y t_{2y} + u_z t_{2z} \quad . \end{aligned} \tag{85}$$

The normal and tangential velocities of the left and right state are then given to the flux calculator along with the other interpolated flow properties. The interface flow properties estimated by the flux calculator are rotated back to the global Cartesian coordinates using

$$\begin{aligned} u_x &= u_n n_x + u_{t1} t_{1x} + u_{t2} t_{2x} \quad , \\ u_y &= u_n n_y + u_{t1} t_{1y} + u_{t2} t_{2y} \quad , \\ u_z &= u_n n_z + u_{t1} t_{1z} + u_{t2} t_{2z} \quad , \end{aligned} \tag{86}$$

and then combined to form the inviscid flux components of Eq. (66).

Two flux calculators are available in *sm_3d*$^+$ for calculating the inviscid fluxes. These are an approximate Riemann calculator [43] and an Advection Upstream splitting Method (AUSM) by Liou & Steffen [44].

The approximate Riemann calculator has the ability to capture shock waves and other sharp features with optimal resolution and with reduced spurious oscillations of traditional finite difference methods with artificial viscosity. Its conservative character ensures correct positions of the computed shock waves and its robustness ensures good stability in high gradient flow situations.

The AUSM technique is a relatively new and is remarkably simple in comparison to the Riemann technique. It converges as fast as the Riemann calculators, has a comparable accuracy and is also robust.

2.7.4 Approximate Riemann Flux Calculator

The Riemann flux calculator is an implementation of the approximate calculator described in [43] without the strong-shock stage and a linearized Riemann solver for slowly varying data. Fluxes are computed by first calculating the intermediate pressure and velocity due to interactions of isentropic compression and rarefaction waves between adjacent cells. If the left and right moving waves are shown to be neither both rarefactions or shocks, the linearized Riemann solver of Toro [45] is used. The linearized solver is less computationally expensive than the approximate Riemann solver of Osher [46] and produces comparable results in the regions of applicability. After the intermediate states have been computed, it is used in conjunction with the left and right states to calculate the interface state.

This flux calculator has been shown to be accurate in viscous and inviscid flows [43, 1] however, it lacks numerical dissipation and the computational cost is high despite the addition of the linearized solver. See Ref. [43] for a more complete description of the approximate Riemann calculator.

2.7.5 AUSM Flux Calculator

The AUSM scheme [44] treats the convective and pressure terms within the flux vectors separately. The convective terms are upstream biased using a properly defined cell-interface velocity, while the pressure term is computed by considering acoustic waves. The development of the AUSM flux solver is based on splitting the inviscid flux vector into two physically distinct parts. If a one-dimensional inviscid flux vector $\mathbf{F}$ is considered, a convective and pressure term can be formed.

$$\mathbf{F} = \begin{pmatrix} \rho \\ \rho u \\ \rho E + p \end{pmatrix} u + \begin{pmatrix} 0 \\ p \\ 0 \end{pmatrix} = \mathbf{F}^{(c)} + \begin{pmatrix} 0 \\ p \\ 0 \end{pmatrix} \tag{87}$$

The velocity u at the interface convects the first vector of passive quantities (including the flux of all species when chemistry is invoked). The pressure flux term is governed by the acoustic wave speed. At an interface $L < \frac{1}{2} < R$, the convective terms can be written as

$$\mathbf{F}^{(c)}_{1/2} = u_{1/2} \begin{pmatrix} \rho \\ \rho u \\ \rho E + p \end{pmatrix}_{L/R} = M_{1/2} \begin{pmatrix} \rho a \\ \rho a u \\ \rho a E + p a \end{pmatrix}_{L/R}, \tag{88}$$

where

$$(\cdot)_{L/R} = \begin{cases} (\cdot)_L, & \text{if} \quad M_{1/2} \geq 0 \quad , \\ (\cdot)_R, & \text{otherwise.} \end{cases} \tag{89}$$

The advective velocity $M_{1/2}$, is a combination of the wave speeds travelling towards the interface $(1/2)$ from the L and R cells such that,

$$M_{1/2} = u_{1/2}/a = M_L^+ + M_R^- \quad , \tag{90}$$

where

$$a = \begin{cases} a_L & \text{if} \quad M_{1/2} \geq 0.0 \quad , \\ a_R & \text{otherwise.} \end{cases} \tag{91}$$

Of the ways of defining the split Mach numbers $M^{\pm}$, the van Leer splitting technique [47] is used where

$$M^{\pm} = \begin{cases} \pm\frac{1}{4}(M \pm 1)^2, & \text{if} \quad |M| \leq 1 \quad , \\ \frac{1}{2}(M \pm |M|), & \text{otherwise.} \end{cases} \tag{92}$$

The pressure term is split at the interface into left and right terms

$$p_{1/2} = p_L^+ + p_R^- \quad . \tag{93}$$

The pressure splitting is weighted using the first order polynomial expansion of the characteristic speeds $(M \pm 1)$ as

$$p^{\pm} = \begin{cases} \frac{p}{2}(1 \pm M), & \text{if} \quad |M| \leq 1 \quad , \\ \frac{p}{2}(1 \pm |M|)/M, & \text{otherwise.} \end{cases} \tag{94}$$

The AUSM scheme, as shown above, is quite simple and has the same order of computational efficiency as Flux Vector Splitting schemes such as those proposed by Steger & Warming [48] and van Leer [47]. It also achieves a high level of accuracy attributed only to flux difference splitting methods such as that proposed by Roe [49]. The efficiency and accuracy of the AUSM scheme makes it suitable for viscous and chemically reacting flows.

2.8 Calculating the Viscous Fluxes

To calculate the viscous fluxes, the derivatives in Eqs. (7) and (8) need to be numerically approximated. The derivatives to be calculated are the velocity and temperature derivatives across each cell face. The transport coefficients also need to be calculated, however, they are determined as functions of the gas state at the interfaces and boundaries where the gas state is explicitly specified.

The divergence theorem is used to calculate the derivatives at the cell vertices. The four vertex values making up each face are then averaged to get the interface derivative. The divergence theorem can be written as

$$\frac{1}{V}\int_V \nabla\phi \; dV = \frac{1}{V}\oint (\phi \; \hat{n}) dA \quad , \tag{95}$$

where ϕ represents some arbitrary scalar variable.

To apply the divergence theorem, secondary cells need to be formed around the primary cell vertices. The primary cell vertices are the secondary cell centres and the neighbouring primary cell centres are the secondary cell vertices (see Fig. 11). For each cell, four upstream and four downstream secondary cells need to be formed. The flow properties u_x, u_y, u_z and T at the cell centres of the primary cells are copied to the vertices of the secondary cells and the areas, volumes and unit normals on the faces are calculated. For secondary cells on the boundaries, the vertices of the secondary cells are constructed using the primary cell centres of the cells nearest to the boundary and the centres of the cell faces that form the boundaries. This constitutes a half cell; similarly, corner secondary cells are eighth cells. The semi-discrete form of the divergence theorem can be applied to the secondary cells as

$$\left(\frac{\partial\phi}{\partial x}\right)_{i,j} = \frac{1}{4}\frac{1}{V_{i,j}} \sum_{i=\text{face 1}}^{\text{face 6}} A_i(\hat{n}_i.\hat{i})(\phi^1+\phi^2+\phi^3+\phi^4) \quad , \tag{96}$$

$$\left(\frac{\partial\phi}{\partial y}\right)_{i,j} = \frac{1}{4}\frac{1}{V_{i,j}} \sum_{i=\text{face 1}}^{\text{face 6}} A_i(\hat{n}_i.\hat{j})(\phi^1+\phi^2+\phi^3+\phi^4) \quad ,$$

$$\left(\frac{\partial\phi}{\partial z}\right)_{i,j} = \frac{1}{4}\frac{1}{V_{i,j}} \sum_{i=\text{face 1}}^{\text{face 6}} A_i(\hat{n}_i.\hat{k})(\phi^1+\phi^2+\phi^3+\phi^4) \quad , \tag{97}$$

where $V_{i,j}$ is the volume of the secondary cell, face 1 to face 6 are the six faces of the secondary cell, and $\phi^1, \phi^2, \phi^3, \phi^4$ are the variables at the four vertices of each secondary cell face. The derivatives evaluated at the centres of the secondary cells are then averaged to get the derivative values at the primary cell interfaces.

2.8.1 Viscous Boundary Conditions

Viscous boundary conditions are applied to calculate the viscous derivatives as outlined above and also the transport coefficients on the boundaries. For this reason, only flow properties on the cell interfaces that form the boundaries need to be changed to match the boundary conditions.

When the boundary conditions are set to solid, no-slip, adiabatic or isothermal walls, all the velocity components are set to zero at the boundary interfaces. Isothermal or adiabatic wall conditions can be set by maintaining a fixed temperature on the boundary interfaces for isothermal conditions or by setting the temperature to the adjacent primary cell centred value for adiabatic conditions.

If a solid wall with an inviscid tangency condition is selected, cell-centred values of the first inviscid ghost cell and the first adjacent interior cell are averaged to form the boundary interface values. This condition would be selected for a plane of symmetry.

The supersonic inflow condition is the same as the inviscid condition with the boundary interface flow properties set to the free-stream values.

2.9 Time Integration for Each Slice

The simplest way of integrating the governing equations is to use an explicit time-stepping scheme. Once the time differentials of dependent flow variables for each cell in the slice are obtained from Eq. (66), the solution is stepped forward in time by a small amount Δt by applying the scheme

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \, \frac{d\mathbf{U}^n}{dt} \quad . \tag{98}$$

This simple scheme is used in preference of more elaborate implicit and predictor-corrector schemes. The latter schemes require increased computational effort which is an important consideration for large three-dimensional problems. The explicit scheme is also simple to code and requires less data storage than implicit methods. However, this approach is not as robust as implicit schemes so time steps have to be restricted to maintain stability.

The time step Δt in the explicit scheme, is a functions of the smallest time scale present in the solution slice.

$$\Delta t = \mathrm{CFL} \quad . \quad \mathrm{MIN}\left(\Delta t_\xi,\ \Delta t_\eta,\ \Delta t_\zeta,\ \Delta t_{\mathrm{viscous}},\ \Delta t_{\mathrm{chem}}\right) \tag{99}$$

There are three inviscid acoustic time scales, a viscous time scale and a time scale based on the fastest reaction rate. The three acoustic time scales are approximate times for acoustic waves to travel through a cell in each each of the three coordinates, (ξ, η, ζ).

$$\begin{aligned} \Delta t_\xi &= \frac{u_x + a}{L_\xi} \\ \Delta t_\eta &= \frac{u_{\text{max ws, YFace}}}{L_\eta} \\ \Delta t_\zeta &= \frac{u_{\text{max ws, ZFace}}}{L_\zeta} \end{aligned} \tag{100}$$

(101)

The wave speed used in the time scale Δt_ξ is calculated by summing the component of velocity in the x direction and the sound speed that has been reconstructed on the downwind $XFace$. The other two inviscid time scales in the cross-stream directions, use the maximum wave speed at each cell face returned by the flux calculator. The characteristic length scales (L_ξ, L_η and L_ζ) are approximate distances from respective centres of opposing cell faces. The viscous time scale [50] is approximated as

$$\Delta t_{\text{viscous}} = \frac{Pr\,\rho}{4\mu\gamma}\left(\frac{1}{L_\xi^2} + \frac{1}{L_\eta^2} + \frac{1}{L_\zeta^2}\right)^{-1} \quad . \tag{102}$$

Stability of chemical reactions in the time integration is maintained by limiting the time step with a chemical time scale. The chemical time scale is selected such that no change in species density (ρf_i) greater than 1×10^{-4} kg/m^3 occurs over a time step. The chemical time step is written as

$$\Delta t_{\text{chem}} = \frac{1 \times 10^{-4}}{\dot{\omega}_{\text{max}}} \quad . \tag{103}$$

The CFL value is the "Courant-Friedrichs-Lewy number " which is a number obtained from a Von Neumann stability analysis of the scaler wave equations. A limit on the value of the CFL number is obtained from reference [51] as

$$\text{CFL} \leq \frac{4}{5 - \kappa + \beta(1+\kappa)} \quad . \tag{104}$$

With $\kappa = 1/3$ and $\beta = 2$, the upper limit for CFL is 0.55, however, because we cannot always make good estimates of the wave speeds across all cells, a value of CFL $= 0.25$ is suggested. In cases where the cells become highly elongated (such as circular duct cross-sections), a smaller value may be appropriate.

When checking to see if the flow has reached a steady state across the whole slice, the following convergence criteria are used:

- Relative changes in density for a time step are less than a specified tolerance (typically 10^{-4}).
- At least five flow lengths have passed through the cell that contained the smallest time scale. If the flow speed is subsonic in the ξ direction for the determining cell, the sound speed is used to calculate the time for a flow length to pass.
- A maximum number of time steps has not been exceeded (typically 100 for non-reacting and 500 for reacting flows).
- If chemical kinetics has been invoked, the maximum fractional change in species production from the maximum production 5 time steps ago, is less than 1×10^{-4}.

Once a steady-state solution is obtained for the current slice, the solver cycles through the data structures maintaining the data for the last two slices for extrapolation, and then proceeds to work on the next downwind slice.

3 Grid Generation

The solver uses structured grids composed of slices of hexahedral cells as shown in Fig. 12. The "bounding box" that contains the slices can be specified using simple

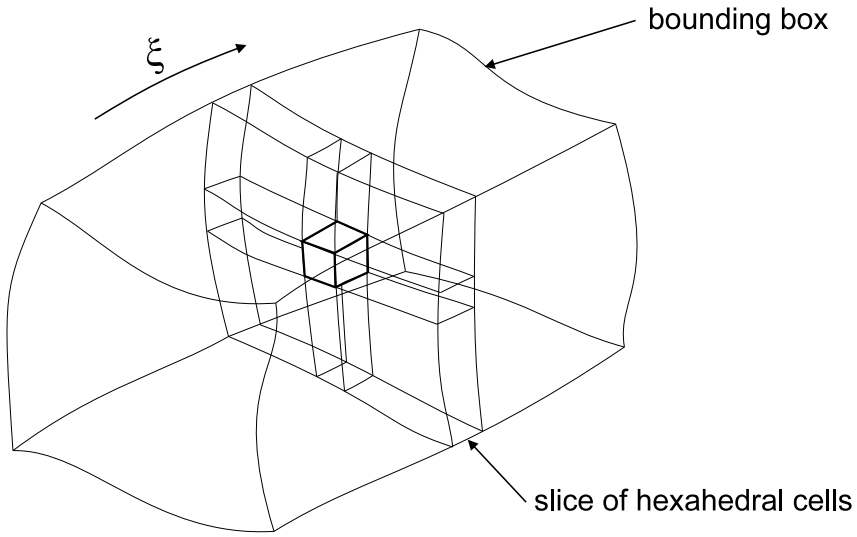

Figure 12: Grid formulation for the space-marching solver.

straight edge panels, panels whose edges are defined by bezier curves or B-spline surfaces. Examples of these different techniques are shown in Fig. 13. Straight edge panel grids and bezier curve grids are defined in the problem parameter file (see section 5.3). In this file, there are sets of (x, y, z) position data for four corner points A, B, C, and D at several ξ locations. Two ξ slices and their associated corner points are shown in Fig. 14. The straight edge panels are made up of linearly interpolated surfaces between neighbouring points where the lines (A'A), (B'B), (C'C), and (D'D) specify the locations of corner points for each vertex slice. For any ξ value, the corner points are located on these lines using linear interpolation. These points are then joined to form the bounds of the slice. Bezier grids are created in a similar fashion, except for the lines specifying the locations of vertex slice corner points which are replaced with bezier curves. B-spline surfaces are used for more complex contoured geometries where the boundaries, of each vertex slice, need to be curved. B-spline surfaces are made from four control net data files which are generated from the suite of programs in [52]. The point data in the parameter file is not required for the B-spline surfaces.

Once the edges of the constant ξ slices are defined, an array of vertices in the (η, ζ)-plane is generated as shown in Fig. 15. The vertices are generated from a set of parameterized interpolation points $\overline{P}_{DA}(\zeta)$, $\overline{P}_{CB}(\zeta)$, $\overline{P}_{DC}(\eta)$, and $\overline{P}_{AB}(\eta)$ for $0 \leq \eta \leq 1$ and $0 \leq \zeta \leq 1$. Transfinite interpolation (or a Coons patch [53]) is then

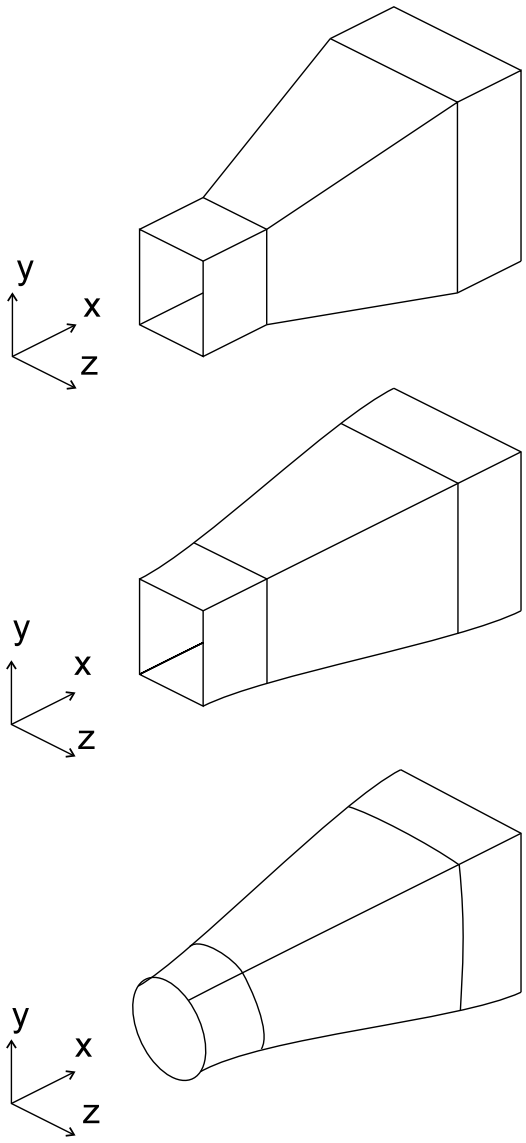

Figure 13: Examples of grids that can be generated; *top:* straight edge panels, *middle:* panels with ξ-edges defined by bezier curves, *bottom:* B-spline surfaces forming the $\eta - \zeta$ exterior surfaces.

used to obtain the position of the vertex as

$$\begin{aligned} \overline{P} &= (1-\eta)\overline{P}_{DA} + \eta\overline{P}_{CB} + (1-\zeta)\overline{P}_{DC} + \zeta\overline{P}_{AB} \\ &\quad -(1-\eta)(1-\zeta)\overline{P}_{D} - (1-\eta)\zeta\overline{P}_{A} - \eta(1-\zeta)\overline{P}_{C} - \eta\zeta\overline{P}_{B} \quad , \end{aligned} \tag{105}$$

where $\overline{P}_A$ to $\overline{P}_D$ are the positions of the corners of the vertex slice. Once a slice of vertex points has been constructed, a small step is taken in ξ to generate another downstream slice of vertex points. These two vertex slices form the corner points of the hexahedral cells.

The interpolation points and corner points need not be coplanar in the Cartesian coordinate system and that the corners need not be corners in a physical sense. For

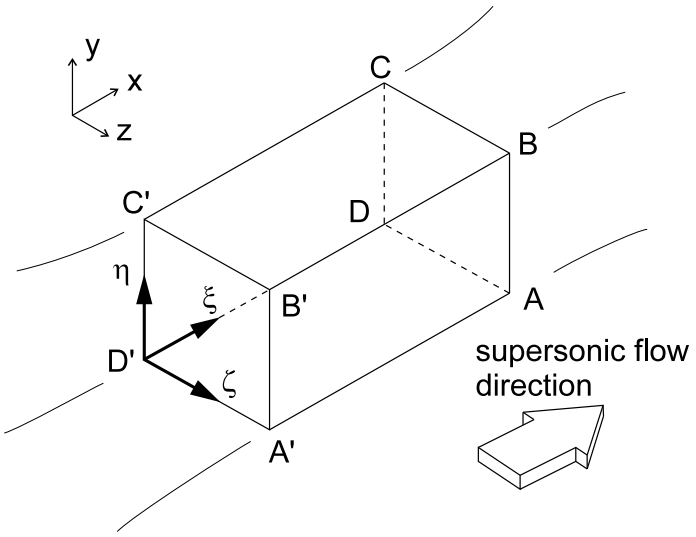

Figure 14: Exterior surfaces of the grid and defining corner points.

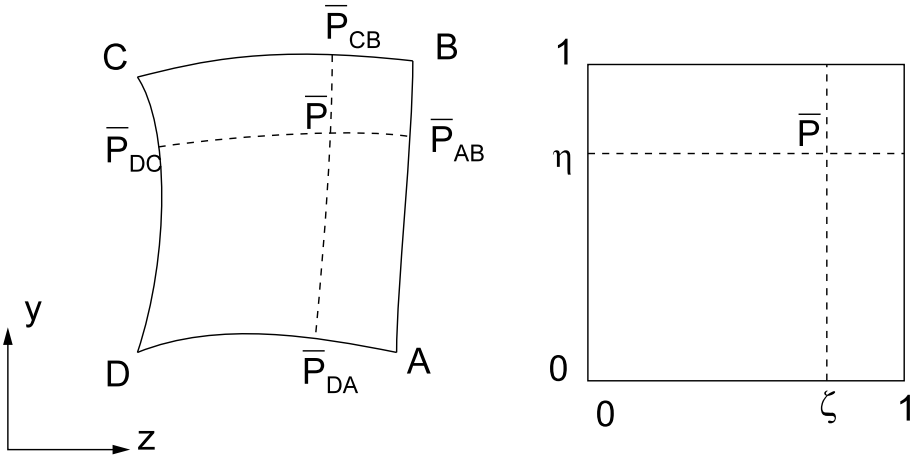

Figure 15: Coons patch interpolation in the (η, ζ)-plane.

example, a circle in the Cartesian coordinate system may be mapped to a square in the (η, ζ)-plane as in Fig. 16. The circular grid in Fig. 16 is an example of a difficult duct geometry because the cells on the 45^o diagonals tend to collapse and have little volume. Better grid generation schemes are available (see e.g. [54]) and may need to be implemented at a later date. There will be an extra computational cost for the better schemes but we can probably achieve a *reasonable* quality of grid with a *reasonable* CPU cost. The alternative is the potential failure of the transfinite interpolation with zero- or negative-volume cells.

As well as solving internal flow problems, sm_3d$^+$ can be used to solve external flow problems over bodies such as cones and cylinders by wrapping the grid around it. An example of a grid for a cone is given in Fig. 17.

The interpolation points used in the transfinite interpolation (Eq. (105)) can be clustered using a Roberts stretching function [55]. The clustering technique uses an

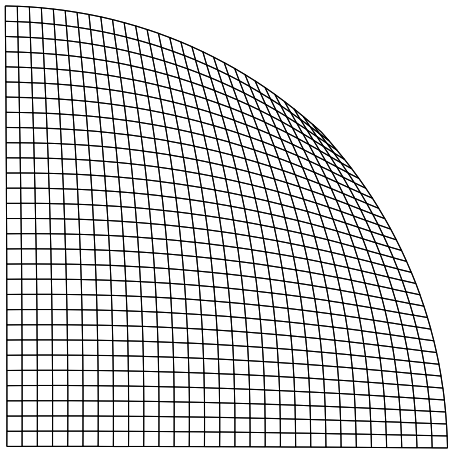

Figure 16: Example of a cross-stream grid for a duct with circular cross-section (quarter sector shown).

exponential formula to cluster parametric points either to one side or both. For a given parametric point η, the clustered point $\bar{\eta}$ would be

$$\bar{\eta} = \frac{(\beta + 2\alpha)\left(\frac{\beta+1}{\beta-1}\right)^{\left(\frac{\eta-\alpha}{1-\alpha}\right)} - \beta + 2\alpha}{(2\alpha+1)\left(1 + \left(\frac{\beta+1}{\beta-1}\right)^{\left(\frac{\eta-\alpha}{1-\alpha}\right)}\right)} \quad , \tag{106}$$

where β is the clustering parameter and α determines the position of the clustering. The range of the clustering parameter is $(1 < \beta < +\infty)$ where the closer it is to 1, the greater the clustering. If α is equal to 0.0, the points will be clustered to the end where $\eta = 0$. If α is equal to 0.5, the points will be clustered at both extremes.

Two-dimensional and axisymmetric cases can also be solved with *sm_3d*$^+$ however the grids are limited to straight edge or bezier bounding boxes. The corner data points A, B, C, and D are read in from the parameter file and the z-coordinate is set to 0.0 for points C and D, and 1.0 for A and B regardless of the parameter file value. A three-dimensional mesh is then made that is 2 cells thick in the ζ direction. During the computation, all fluxes in the z direction are set to zero and calculations are limited to the first plane of cells in this direction.

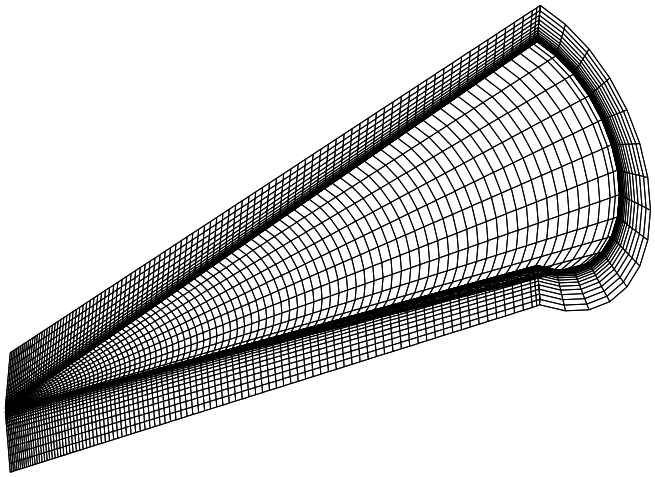

Figure 17: Example grid for flow over a cone.

4 Shape Optimization

In addition to using *sm_3d*$^+$ as a flow solver for cases with known geometries, it can be used as part of the objective function evaluation for an optimization routine which finds optimal shapes of aerodynamic surfaces. With the solver's present modelling capabilities, it is intended to be used for optimizing scramjet combustor/exhaust nozzle configurations and high Mach number axisymmetric and three-dimensional hypersonic wind tunnel nozzles.

For the scramjet case, the objective of the optimization will be to maximise the thrust developed from the expansion ramp for a given equivalence ratio. The design variables are the bezier control points that define the shape of the thrust surface. This is a design case where optimization may be useful because of the highly nonlinear relationship of the finite rate chemistry, viscous effects and the thrust developed by the scramjet engine.

High Mach number nozzles with thick boundary layers, have strong interactions of acoustic waves with subsonic regions of boundary layers. These interaction cannot be determined accurately with "classical" design techniques which use the Method of Characteristics (MOC) to compute an "inviscid" shape, and then modify this shape by adding the displacement thickness of the boundary layer. A better technique is to use optimization with a Navier-Stokes solver which is capable of resolving these interactions. The added complexity of three dimensional flow also makes the use of characteristic based design methods more difficult.

Optimization schemes coupled with a flow solver have been used extensively in the literature for many aerodynamic design problems similar to the ones mentioned above. Some examples of recent applications are airfoils with with high lift to drag ratios [56, 57, 58], three-dimensional hypersonic lifting bodies [59], turbofan engines [60], hypersonic wind-tunnel nozzles [61, 62, 63] and ramp shapes of scramjet-after-body configurations for maximum axial thrust [64]. Various optimization techniques are used in the examples cited, all of which have different advantages and disadvantages. The more common techniques are gradient-search techniques such as least-squares [65] and simplex minimization [8, 10] and, in recent times, Genetic algorithms (GA) which are nongradient stochastic methods [66, 67, 68].

The least-squares technique is based on calculating how much each component of the objective function changes for a small change in each design variable. These derivatives are combined to form a Jacobian matrix, which can be very large if there are many components of the objective function and many design variables. Each derivative must be determined first before any step forward in the design space is performed. The computational effort of solving the Jacobian matrix is balanced by

the rapid convergence of this scheme as demonstrated by Korte [61] and Keeling [62]. These two studies were concerned with optimizing the design of hypersonic axisymmetric nozzles. The optimization technique was successful in these studies because the problems were well posed and the design space had a clear minimum point.

Simplex minimization is similar to the least-squares technique in that it uses a gradient search method, however, simplex minimization is a "direct" method that does not require the evaluation of derivatives. The technique involves forming a "simplex", which is a set of objective function evaluations for a given set of initial design variables where each design variable is individually perturbed a small amount for each evaluation. Therefore, a simplex is formed with $(n+1)$ vertices where n is the number of design variables. Movement is then made away from the vertex with the poorest objective value to find a replacement for it. Spendly *et al.* [69] introduced this idea and it was later improved upon by Nelder & Mead [8]. Nelder & Mead made the process adaptive whereby the simplexes are continually revised to conform to the nature of the response surface. The simplex then contracts to the final minimum. Further modification were made by Routh *etal.* [70] and Parker *etal.* [71] where a curve was fitted to the search direction to find the optimum distance to move in the search space. Both methods are slightly more complicated than the Nelder-Mead method but offer better convergence on relatively simple response surfaces.

GAs represent a class of adaptive algorithms whose search methods are based on the simulation of natural selection and natural genetics [72]. A GA performs a multi-directional search by maintaining a population of potential solutions and encourages information formation and exchange between these directions. The population of each generation undergoes a simulated evolution where the relatively 'good' solutions reproduce, while the relatively 'bad' solutions die. The characteristics of GAs are that they show good exploration of the design space and avoidance of local minima. However, they are poor at converging to an optimal solution in a localised search space [73] and require large amounts of CPU time to maintain population levels [59]. A good overview of GAs is presented by Golberg [67].

The choice the best optimization technique is problem specific. As outlined above, there are advantages and disadvantages of all optimization techniques. However, recent studies [74] indicate that GAs and other stochastic methods are generally not as efficient as gradient methods when considering an equal distribution of all possible objective functions. That being said, GAs are superior in searching entire design spaces. Ideally, the most general optimization technique would be a hybrid of a non-gradient and gradient search method. However, for hypersonic duct flow

where the design space of possible geometries is confined by structural and manufacturing limits, a gradient method is more suited. The gradient search technique of Nelder-Mead is selected for implementation in the current solver due to its simplicity and robustness over other gradient based methods.

4.1 Nelder-Mead Simplex Optimization

The nonlinear "simplex" minimization technique by Nelder and Mead is described comprehensively in [8]. The coding for the technique in *sm_3d*$^+$ is a 'C' translation of the 'FORTRAN' code presented by O'Neill [9]. A brief review of the technique is given below.

Consider a function of n variables that is to be minimized. A simplex is formed in n-dimensional space where the simplex is a set of $(n+1)$ function evaluations with each evaluation having a different set of variables. The best and worst function evaluations of the simplex are identified and a move is made away from the worst evaluation. This move may be in the form of a reflection, contraction, or shrinkage depending on the characteristics of the response surface.

As an example, consider a two-dimensional case where the function to be minimized is a function of two variables x_1 and x_2. A simplex **ABC** is formed as illustrated in Fig. 18 where each point represents an evaluation of the function to be minimized. If **A** represents the highest function evaluation, then a reflection

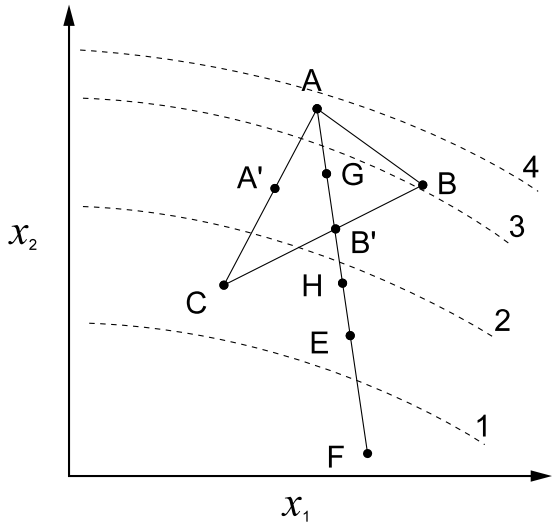

Figure 18: Two-Dimensional Simplex

is made through the centroid or average of the remaining points, which for a two-dimensional case is the bisection point **B′** of the line joining **B** and **C**. The scale of the reflected segment **B′E**, is selected by the user and is typically 1:1 . If the evaluation of the function at the reflected point **E** results in a lower value than the

Table 2: Conditions governing the formation of subsequent simplexes

Condition	Action	New Simplex
$f(C) \leq f(E) \leq f(B)$	Reflect	BCE
$f(E) < f(C)$	Extend	BCF
$f(A) < f(E)$	Contract	BCG
$f(B) < f(E) \leq f(A)$	Contract	BCH
$f(A) \leq f(G)$ or $f(E) \leq f(H)$	Shrink	A′B′C′

remaining points, then an extension to point **F** would ensue. The resulting simplex would then be **BCF**. The complete list of possible actions following the evaluation of point **E** are given in Table 2.

The incorporation of the Nelder-Mead scheme into *sm_3d*$^+$ is in the form of a separate program that calls the flow solver as a subroutine. The solver returns the data at the exit plane of the flow domain as well as integrated wall pressures and moments that were calculated as the solver marched down the flow domain. Using this data, the objective function can be evaluated for the set of design variables given to the solver. The definition of the design variables can be set by the user in the coding of *sm_opt.c* . Typically the design variables are the control points of bezier curves, increments in transverse distances between subsequent points along a duct (thus indicating slope), control points of B-Spline surfaces or scaling parameters for a group of points defining the boundary of the domain. The optimizing routine starts with a given set of design variables and an increment for each design variable to form the simplex. Once the initial simplex is formed and the solver has been called to solve the flow for each point, the objective functions can be determined resulting in a move across the response surface away from the maximum.

The objective function for each case is also set in *sm_opt.c* . A few examples of objective functions are functions that express the variation of a particular flow quantity at the flow exit plane, such as Mach number and flow angularity. Other terms may be deviations from a desired thrust or I_{sp} for a scramjet engine design, total drag on a body, and mass fractions of species resulting from incomplete burning in scramjet engine at the exit plane. The objective function must be formulated so that the desired design is achieved when the evaluation of the function is a minimum as is the case with these examples.

Limits or constraints can be placed on the design variables in two ways. The objective function can be set to a very high number (say of the order 10^{20}) when a design variable goes outside its bounds resulting in a reflection. This represents the so-called method of "external penalty function". Alternatively, if the constraint

for a design variable is that it must always maintain a positive value, then the scale of the variable can be transformed (by using the logarithm, say) so that negative values are excluded.

The optimizer continues to search for the minimum until one of two conditions is reached. Either the number of function evaluations exceeds a set limit, or the "standard error" of the simplexes objective function evaluations falls below a set value. The "standard error" is defined as

$$\text{error} = \sqrt{\frac{\sum (y_i - \bar{y})^2}{n}} \quad , \tag{107}$$

where y_i are the evaluations of the objective functions at each point in the simplex and n is the number of design points. At this point, the values of the objective function are returned as being the best that were found during the search.

5 Use of the Computer Codes

The complete solver and associated programs consist of approximately 25000 lines of code and 2000 lines of header files all written in ANSI C. There are 91 functions that are combined into file modules. Table 3 lists the file modules in classes and provides a brief explanation of their contents. The modules whose names are prefixed with "sm_" have been written specifically for this project. The remaining modules were written for other projects and are used for gas properties, vector geometry, and B-spline surfaces [52].

All of the functions, header files, parameter files for test cases, and *makefile* are available on request to the authors. These files can be copied onto any machine that has an ANSI C compiler. It is expected that most simulations will be run on UNIX based systems so the *makefile* has been included for these operating environments. To build the programs, edit the header file *compiler.h* and choose the appropriate environment and compiler, then run the *makefile*. The *makefile* supplied will link and compile all of the functions and header files into the executable programs having the same prefix names as the bold file modules in Table 3.

Once the code has been compiled, a simulation may be conducted. A simulation is typically performed in a number of stages:

(1) Definition of Problem

To define the problem, a *parameter* file is created along with any associated files for geometry and chemistry if they are required. The macro definitions and gas type are then set in the main header file *sm_3d.h* (see sections 5.1 and 5.2). The *parameter* file contains information about the number of cells, time stepping, inflow conditions, and computational domain geometry. The name of the file must have the extension "*par*" to identify it. The first part of the file name is called the "base" file name, $<bfn>$, and is common for all the input and output file names.

The data entered into the *parameter* file for the geometry can be used to generate simple panel grids and grids with streamwise edges defined by bezier curves (see Fig. 13). There are two other options for grid generation. A set of B-Spline surfaces can be generated with the suite of codes written by Craddock [52]. If B-Spline surfaces are selected, the grid generating module looks for the data files $<bfn>.q1, q2, q3, q4$. Each file contains the control points for one boundary surface of the flow domain. The other option is to hard code the geometry in the file *sm_geom.c* (see section 5.4). Hard coding the geometry requires a knowledge of the code structure. The simplest way of hard coding a geometry is to copy an existing example and modify it.

Table 3: File modules containing functions

Module name	Content
	Pre-processing
sm_prep.c	preprocessing (produces an inlet-plane solution)
	Solving
sm_3d.c	main space-marching program
sm_ausm.c	AUSM flux solver (inviscid fluxes)
sm_chem.c	chemical kinetic functions
sm_gas.c	gas properties (equation of state etc.)
sm_geom.c	geometry definition (cell volumes etc.)
sm_ibc.c	inviscid boundary conditions
sm_iflux.c	inviscid-flux calculations
sm_io.c	input/output routines
sm_misc.c	anything that wouldn't fit elsewhere
sm_rivp.c	Riemann solver (inviscid fluxes)
sm_step.c	space- and time-stepping functions
sm_src.c	source terms (heat addition)
sm_trans.c	viscous transport properties
sm_turb.c	Baldwin-Lomax turbulence model for 2 and 3 dimensions
sm_vflux.c	viscous flux functions
sm_vbc.c	viscous boundary conditions
sm_vgeom.c	viscous geometry for secondary cells
tgas1.c	equilibrium-air gas properties
n2eq.c	equilibrium nitrogen gas properties
n2vibe.c	vibrational equilibrium nitrogen gas properties
geom.c	three-dimensional vector routines
roberts.c	grid distribution stretching functions
bspatch.c	B-spline routines for duct wall definition
bezier.c	Bezier polynomial routines
	Post-processing
sm_post.c	extract multiple slices from the *.out* file
sm_slice.c	extract a single slice from the *.out* file
sm_mf.c	extract multiple slices from the *.mf* file
sm_prof.c	extract strips of data from the *.out* file
sm_probe.c	extract data for individual cells from the *.out* file
	Optimization
sm_opt.c	case specific design variables and objective functions
sm_nelmn.c	Nelder Mead Simplex minimisation function

If a chemically reacting solution is required, the user must specify a chemistry model to use. The four models supplied with the code in the *chm_files* directory are listed below.

AIR.chm	:	simple five species, 3 reaction model for nitrogen and oxygen chemistry
CO2.chm	:	four species, 3 reaction model for carbon dioxide and oxygen chemistry
H2O2.chm	:	18 reaction mechanism of Jachimowski [27] for combustion of hydrogen in air
NASP.chm	:	The NASP 31 reaction mechanism for combustion of hydrogen in air [23]

The simplest way of using the chemistry files is to copy one of these files into the problem directory and rename it with the problem base file name $<bfn>.chm$. The list of mass fractions contained in this file can then be changed to suit the required inflow conditions. A completely new chemistry file can be created by the user by writing it in the same format as those supplied. Special care should be taken to write the thermodynamic polynomial curve fit data in the "New NASA Lewis" format [23].

(2) Compilation & Pre-processing

Once the required input files have been created and the required modifications have been made to the source code and header files, the code needs to be re-compiled to optimize the solver's internal coding structure by omitting unnecessary routines and memory storage. The preprocessor *sm_prep.x* can then be run to produce a single-slice solution file for the inlet plane. When the program is started, it requests the "base" file name $<bfn>$ so that it can read and create the correct input and output files. If the format of all input files is correct, *sm_prep.x* will generate the output file containing the first slice of data. This file is identified by an "*ini*" extension.

(3) Simulation

The next stage involves the use of the main space-marching program *sm_3d.x*. It too requests the "base" file name $<bfn>$ so that it can find all the required input files. If the user does not require an optimized solution, the code simply reads the parameter file *par* and the *ini* file as a starting solution, and produces an output file $<bfn>.out$ that contains one or more slices of the flow solution along the

computational domain. If finite-rate chemical kinetics was selected, an additional output file <*bfn*>.*mf* will have been generated. This file lists all the species mass fractions for every cell in the solution slices.

During the calculation process, the solver will print to the screen a summary of the present problem and various pieces of information after each slice has reached a steady state. The current slice position, time steps, errors, total wall forces and moments about the origin are all printed for each slice. At every slice where information is to be written to the output file, the total integrated forces, moments and heat added to the slice are also printed. In most circumstances, this information is redirected into a log file, <*bfn*>.*log*, and the solver is run as a background process. The way to accomplish this is to use a batch file. A batch file may look something like;

```
cd $HOME/sm_3d/data/cyl
date
time sm_3d_cyl.x < cyl.sm > cyl.log
date
```

Line one of the batch file makes the current directory the working directory. The second line prints the current date and time to the screen. The third line, times the execution of the solver where the executable has been renamed to a *sm_3d_cyl.x* in the working directory. The required keyboard input into the solver has been provided by the file *cyl.sm* and the output of the solver has been redirected to the log file *cyl.log*. The file *cyl.sm* is a one line file that contains the *base file name* which in this case is *cyl*. Line four prints the date and time to the screen again when the solver has finished.

To run this batch file in background, issue the following command at a T shell or C shell UNIX prompt:

```
> (source cyl.bat)&
```

Optimization cases will generate a <*bfn*>.*opt* file which reports on the progress of the optimization. Each line of the file lists the iteration number, values of the design variables, and the value of the objective. When running an optimization case, the information being printed to the screen by the solver will be automatically reduced.

(3) Post-processing

After the solver has completed the simulation, the post-processing programs can be used to extract meaningful data from the output data files. All the post-

processing programs generate files that are in a format that can be read and displayed with the commercial visualisation package TECPLOT (TM). The four post-processing programs are listed below.

sm_post.x : extracts multiple slices from the *.out* file

sm_slice.x : extracts a single slice from the *.out* file

sm_prof.x : extracts strips of data

sm_mf.x : extracts multiple slices of mass fractions from the *.mf* file

The first three programs will generate a $<bfn>.tec$ file that can be viewed directly with TECPLOT. The mass fraction program, **sm_mf.x**, will generate a file with the extension $\_mf.tec$, which contains mass fractions at point locations. Each post-processing program lets the user plot any flow quantities that are desired. Note that the the vertex locations in the post-processing output files, correspond to the cell-centroid locations in the computed solution.

A step by step overview of the typical problem processing procedure is presented in Fig. 19.

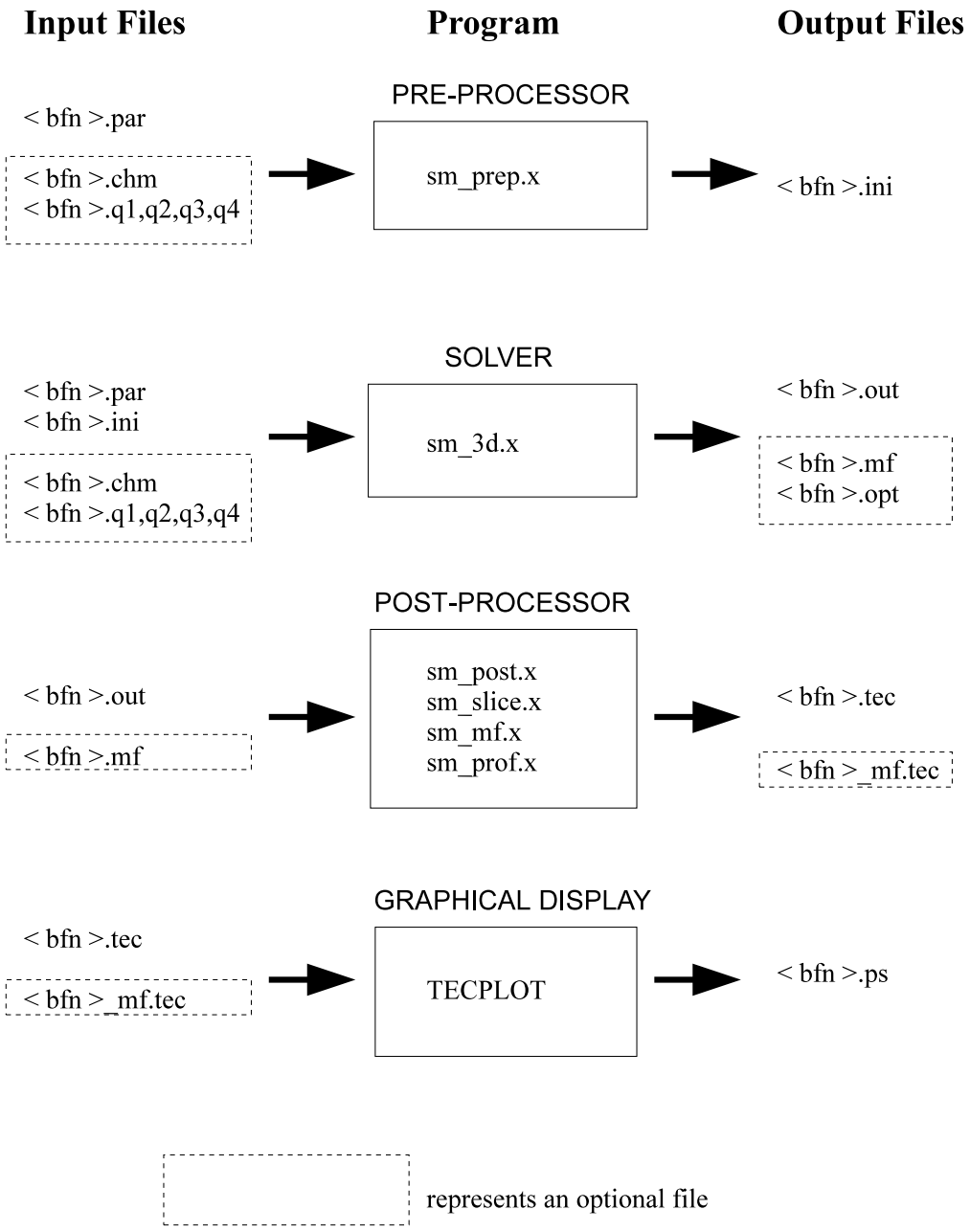

Figure 19: Flow chart of steps taken to perform a simulation showing files required and generated.

5.1 Macro Definitions

The type of computation performed is determined by a number of macro-definitions in the *sm_3d.h* header file (see Appendix E). These should be checked before any simulation is performed to confirm the appropriate settings. After they are checked, the code will need to be re-compiled to optimize the computational performance. The executable generated after complilation, *sm_3d.x*, should then be copied into the problem working directory to avoid confusion with other executables.

The macro-definitions available to the user are:

- DEBUG
  - 0 : no debugging information will be written
  - 1 : (default) print messages on entry to infrequently-called functions
  - 2 : print messages on entry to frequently-called functions
  - 3 : dump data every step (although this is tedious it is sometimes the only way to find a subtle problem)
- EXTRAP_TYPE
  - 0 : (default) second-order upwind-biased extrapolation is used to reconstruct the fluxes on the downwind $XFace$
  - 1 : second-order fully upwind reconstruction is used
- INTERP_TYPE
  - 0 : (default) third-order upwind-biased reconstruction is used for the *YFaces* and *ZFaces*.
  - 1 : fully upwind reconstruction is used
- FLUX_SPLIT
  - 0 : (default) the Riemann solver is used
  - 1 : AUSM flux splitting is used
- DIMENSION
  - 0 : axisymmetric flow solution is computed
  - 1 : one-dimensional flow solution is computed (not used)
  - 2 : two-dimensional flow solution is computed
  - 3 : three-dimensional flow solution is computed
- VISC
  - 0 : the Euler equations are used
  - 1 : the PNS equations are used
- TURB
  - 0 : laminar flow solution
  - 1 : turbulent flow solution (Baldwin-Lomax model)
- CMUTM

0 : no transition to turbulence; always turbulent

14 : default setting for transition condition in the Baldwin-Lomax turbulence model (see section 2.5).

- CHEM

  0 : use perfect gas or equilibrium models for equation of state (EOS)

  1 : use the finite rate chemical kinetics routines to determine EOS and species mass fractions; note that a (*.chm*) file must be available to the code

- FROZEN

  0 : do nothing

  1 : if chemistry enabled, set all the production rates to 0

- ADD_HEAT

  0 : do not add heat in the source-term routine

  1 : include the heat production terms as coded for each particular case; the heat terms are coded in the code module *sm_src.c*

- N_SP

  declaration of maximum number of species in a reaction set; that is we can not have more than N_SP species in a reaction set; the default value is 11; the value only needs to be changed if there are problems with memory or lack there of

- N_RE

  declaration of maximum number of reactions that the code can deal with; the default value is 20; the value only needs to be changed if there are problems with memory

- BETA_X, BETA_Y, BETA_Z

  grid clustering factors for each direction;
  the range of BETA is $1 < \text{BETA} < +\infty$, where the closer BETA is to 1, the greater the clustering

- X_START, X_END, Y_BOTTOM, Y_TOP, Z_LEFT, Z_RIGHT

  These macros indicate where to cluster the cells. A value of one clusters the mesh to the side indicated by the variable. Note that if a 2D or axisymmetric simulation is being performed, the clustering macros for the Z direction should both be set to 0.

- NDVMAX

  declared vector size for the design variables (only relevant for optimization cases)

- VOLUME_TOL

  the magnitude of the tetrahedra (negative) volume that will be tolerated (it seems that for some difficult (read distorted) geometries, the cells have some negative tetrahedra volumes which could be a result of the grid lines crossing; default setting $1.0 \times 10^{-7}\text{m}^3$)

5.2 Gas Selection

After all the MACROs (or calculation switches) have been set, a gas needs to be selected unless CHEM has been set to 1. The selection of a gas is performed by uncommenting one of the TYPE_OF_GAS definitions which proceed the descriptions in the header file. The available gases consist of a number of perfect gases, nitrogen and air in chemical equilibrium, and nitrogen in vibrational equilibrium. If the CHEM macro has been set to 1, the choice of gas will not be of any consequence and the code will look for the $<bfn>.chm$ file for EOS data. The list of gases to chose from is presented below.

PERF_AIR_14	:	perfect gas air with $\gamma = 1.4$
LOWT_AIR_14	:	perfect gas air for very low temperatures, $\gamma = 1.4$
PERF_AIR_13	:	perfect gas air for scramjet work, $\gamma = 1.32$ (this lower value of γ is used to model some of the high-temperature effects)
PERF_HE_167	:	perfect gas Helium with $\gamma = 1.667$
PERF_AR_167	:	perfect gas Argon with $\gamma = 1.667$
PERF_N2	:	perfect gas Nitrogen with $\gamma = 1.4$
PERF_CO2	:	perfect gas Carbon Dioxide, $\gamma = 1.28$
EQ_AIR_1SP	:	high-temperature air in chemical equilibrium (the equation of state is coded in module *tgas1.c* which was adapted from the FORTRAN versions of Ref. [26])
EQ_N2	:	Nitrogen in chemical equilibrium (the equation of state is coded in module *n2eq.c* which contains curve fits for the EOS)
VIBEQ_N2	:	Nitrogen in vibrational equilibrium (the equation of state is coded in module *n2vibe.c*; Nitrogen is assumed not to dissociate and the equilibrium quantities are worked out using the characteristic temperature for vibration)
WEIRD_167	:	Nitrogen with an unusual perfect gas EOS for DSMC comparison

5.3 Parameter File

The parameter file $<bfn>.par$ is the primary data file that defines the problem. The contents of this file are shown below and examples of actual files are provided in the Test Case section of this document.

Note that the format is specified using the C-language notation. A "%f" indicates that a floating-point number is expected, while "%d" indicates an integer, and "%s" indicates a string of characters.

line 1 %s : title string of up to 132 characters

line 2 %d : *case_id*

- A list of valid cases can be found in *sm_3d.h* (see Appendix E) while a one-line description can be found in the function *sm_initialize()*. To add another case, its definition in added in *sm_initialize()* and the macro name is used throughout the rest of the code.

line 3 %f %d %f : *cfl_target, max_t_steps, tolerance*

- Recommended values are *cfl_target* = 0.25, *max_t_steps* = (200 for non-reacting and 500 for reacting gases), and *tolerance* = 10^{-4}. As a consistency check, tolerance should be decreased by an order of magnitude to see if the solution at each slice has reached a genuine steady state.

line 4 %d %d %f : *Xorder, i_suppress, p_safety*

- *Xorder* specifies the order of reconstruction. A value of 1 sets low-order reconstruction (i.e. none) and a value of 2 sets high-order reconstruction.
- *i_suppress* indicates the number of steamwise slices from the initial slice, where the pressure gradient is set to 0 in the boundary layer to maintain stability (recommended value is 3). Used for viscous flow only.
- *p_safety* is the safety factor which maintains real positive eigenvalues in the boundary layer. It is typically set between 0.75 (high viscous interactions) and 1.0 (low interactions). Used for viscous flow only.

line 5 %f %f %f %d : *Xi, dXi, Xi_max, max_x_steps*

- *Xi*($= \xi$) is the starting point for the calculation in normalised (computational) coordinates $0 \leq \xi \leq 1.0$.
- *dXi*: streamwise step size. This should be selected to ensure that the cells do not become too elongated.
- *Xi_max*: a downstream limit for the calculation
- *max_x_steps*: a limit on the number of space-marching steps

line 6 %f : *dXi_plot*

- *dXi_plot* is the normalised distance between slices that are written to the output file. This data in the output file can then be picked up as a whole by the post-processing programs and used for plotting.

line 7 %d : *ident*

- *ident* is a slice identifier that is intended for use in the (later) multi-flow-path version of the code.

line 8 %d %d : *nnz, nny*

- *nnz* and *nny* are the numbers of cells in the ζ and η directions respectively.

line 9 %d %d : *smooth_grid, smooth_iter*

- *smooth_grid* is a switch for a Laplacian grid smoother. The grid smoother essentially averages neighbouring points to get the new coordinates of the point.
- *smooth_iter* is used to set the number of Laplacian iterations.

line 10 %d %d %d %d : *bc_N, bc_E, bc_S, bc_W*

- These integers set the wall boundary-condition and may take the following values:
  - 0 : adjacent to another flow path
  - 1 : supersonic inflow condition
  - 2 : supersonic outflow (not used)
  - 3 : solid wall with inviscid (slip) tangency condition
  - 4 : solid, no-slip, adiabatic wall
  - 5 : solid, no-slip, fixed temperature wall

line 11 %f %f %f %f : *Twall_N, Twall_E, Twall_S, Twall_W*

- Wall temperatures for viscous flows.

line 12 %f %f %f %f %f ; ρ, u_x, u_y, u_z, e

- These are the inlet flow conditions.

ρ	:	density in kg/m^3
u_x, u_y, u_z	:	velocity components in m/s
e	:	specific internal energy in J/kg

line 13 %d %d : *use_B_splines, use_bezier_box*

- *use_B_splines* is set to 1 to use the data from the control net files <*bfn*> .$q1, q2, q3, q4$ to create the grid.
- *use_bezier_box* is set to 1 to use the point coordinates on the following lines as control points for bezier curves that define the steamwise edges of the grid domain.
- If both set *use_B_splines* and *use_bezier_box* are set to 0 and there are no hard coded goemetries for the case being considered, the grid will be made up of quadrilateral panels between the cross-sections that follow.

For cases where the grid is "hard-coded" into *sm_geom.c*, the remaining lines are not necessary in the input file.

line 14 %d : *np*

- *np* is the number of quadrilateral cross-sections used to define the duct. Each cross-section is defined by its corner points in 3-D space.

line 15 %f %f %f : *PA.x, PA.y, PA.z* are the coordinates for point A in m.

line 16 %f %f %f : *PB.x, PB.y, PB.z* are the coordinates for point B in m.

line 17 %f %f %f : *PC.x, PC.y, PC.z* are the coordinates for point C in m.

line 18 %f %f %f : *PD.x, PD.y, PD.z* are the coordinates for point D in m.

Note that lines 15 to 18 are repeated for each *ip* cross-section $0 \leq ip < np$.

On line 2, a *case_id* is requested. This number is only essential for hard coded geometries and optimization routines. Otherwise, it can be set to 0 which is the generic case. The range for case numbers has been arbitrarily set as,

0	: Generic case
$1 \rightarrow 70$	: Perfect gas - nothing special
$71 \rightarrow 100$	: Axisymmetric cases
$101 \rightarrow 200$	: Optimization cases
$201 \rightarrow 300$	: Finite rate chemistry
$301 \rightarrow 400$	: Finite rate chemistry with optimization
$401 \rightarrow 1000$	: Empty
$1001 \rightarrow 1008$	: Test cases

A list of pre-defined cases appears in the header file *sm_3d.h*. The user may select one of these or append to the list in the appropriate group. If adding a case, an appropriate *printf* statement will need to be added to the *sm_io.c* file.

5.4 Hard Coded Geometry Example

This section gives an example of hard coding a geometry in the *sm_geom.c* file. Before the grid generation function is written, an addition must be made in the function *sm_generate_vertex_grid()* to tell the code where to find the new grid function. For this example, the new grid function will be called *sm_generate_example_grid*. So the appropriate line in the function *sm_generate_vertex_grid()* would look like,

```
                    :
                    :
else if ( gd->case_id == DRUM_CONIC )
   sm_generate_drum_conic (s, d, gd, Xi);

else if ( gd->case_id == EXAMPLE )
   sm_generate_example_grid (s, d, gd, Xi);

else if ( d->use_B_spline == 1 )
   sm_generate_bspline_grid (s, d, gd, Xi);

else if ( d->bezier_box == 1 )
   sm_generate_bezier_box_grid (s, d, gd, Xi);
                    :
                    :
```

In the coding above, the example grid function will only be called if the *case_id* is equal to EXAMPLE which needs to be defined in the header file *sm_3d.h*. The new grid generation function must be added to the prototypes in the header file *sm_3d.h* as well. To create the actual function, a simple approach is to copy an existing function and then modify it. Below is a function that generates a cone grid at a 10 degree angle of attack with a length of 0.3048m.

```
/*---------------------------------------------------------------*/

#if (PROTO)

int sm_generate_example_grid ( struct slice_data *s,
                               struct duct_data *d,
                               struct global_data *gd,
                               double Xi )
#else

int sm_generate_example_grid ( s, d, gd, Xi )
struct slice_data *s;
struct duct_data *d;
struct global_data *gd;
double Xi;

#endif

{  /* begin sm_generate_example_grid() */
int    iy, iz, ii, jj;
double eta, zeta, dzeta;
struct point_3D *vtx_loc;
struct point_3D P;
double r, R1, R2, Rg00, Rg01, Rg10, Rg11;
double theta, L, Rmin, Rbase;
double H, W, Zwing, f;

/*
 * Distribute points along each parameter direction.
 */
distribute_points_1 ( 0.0, 1.0, s->nnz, s->zeta_v, 0, 0, 0.0 );
distribute_points_1 ( 0.0, 1.0, s->nny, s->eta_v, 1, 0, 1.12 );

/*
 * The boundaries of the grid for the 10 degree, 0.3048m cone.
 */
Rmin  = 2.645e-3;  /* Small, positive value to avoid a singularity */
Rbase = 53.74e-3;  /* Base radius                                  */
L     = 0.3048;    /* Cone length                                  */
Rg00  = 3.756e-3;  /* Starting radius of windward grid             */
Rg01  = 9.258e-3;  /* Starting radius of leeward grid              */
Rg10  = 0.0763;    /* Finishing radius of windward grid            */
Rg11  = 0.1881;    /* Finishing radius of leeward grid             */

/*
 * Generate the vertices using polar coordinates.
 */
for (iy = s->iymin - 1; iy <= s->iymax; ++iy)
   {
```

```
   ii = iy - s->iymin + 1;
   eta = s->eta_v[ii];
   for (iz = s->izmin - 1; iz <= s->izmax; ++iz)
      {
      vtx_loc = &(s->CVtx[s->vix][iy][iz].loc);
      jj      = iz - s->izmin + 1;
      zeta    = s->zeta_v[jj];

      /*
       * Point on cone surface is the base radius plus the local
       * height of the wing.
       */
      P.x   = Xi * L;
      R1    = Xi * Rbase;
      R1    = MAXIMUM(R1, Rmin);

      /*
       * Outer radius of the grid.
       * Interpolate using two parameters.
       */
      Xi   = MAXIMUM(0.0,Xi);
      Xi   = MINIMUM(1.0,Xi);
      zeta = MAXIMUM(0.0,zeta);
      zeta = MINIMUM(1.0,zeta);
      R2   = (1.0 - Xi) * (1.0 - zeta)*Rg00 +
             (1.0 - Xi) * zeta * Rg01 +
             Xi * (1.0 - zeta) * Rg10 + Xi * zeta * Rg11;
      /*
       * Interpolate to an interior point of the grid.
       */
      r     = (1.0 - eta) * R2 + eta * R1;
      theta = 3.1415927 * zeta;
      P.y   = -r * cos(theta);
      P.z   = r * sin(theta);

      vtx_loc->x = P.x;
      vtx_loc->y = P.y;
      vtx_loc->z = P.z;
      }  /* for (iz... */

   }  /* for (iy... */

return 0;
}  /* end of sm_generate_example_grid() */
/*-------------------------------------------------------------*/
```

When creating a grid function, the input variables must always be s, d, gd and Xi. The function must return the array, s→CVtx[s→vix][ ][ ].loc, with all of the coordinates for the vertex slice with the passed Xi value.

5.5 Hard Coded Optimization Example

The functions that handle the interface between the solver and the optimisation routine are contained within the function module *sm_opt.c*. The functions within

this module are written to handle each case individually. One function defines the design variables and the initial perturbations for each case, another updates the design variables according to the outcome of the optimizer, and another evaluates the objective function. In order to add a new optimization case, each of these functions must be updated with new code.

Presented below, is a section of code that defines the design variable for a simple example that has the *case_id* OPT_RAMP.

```
if ( gd->case_id == OPT_RAMP )
   {
   /*
    * Single ramp with heat addition.
    * The design variable is the angle (in radians) of
    * the thrust ramp (down from the horizontal).
    */
   ndv      = 1;       /* Number of design variables */
   reqmin   = 1.0e-6;
   abstol   = 1.0e-6;
   reltol   = 1.0e-6;
   konvge   = 5;
   maxfe    = 30;
   dv[0]    = 0.2618;  /* initial guess, 15 degrees */
   deldv[0] = 0.02;    /* initial perturbation      */
   }
```

The single design variable for this case is the slope of an expansion ramp, and the initial perturbation of the design variable is 0.02. In addition to defining the design variable/s, the optimization parameters (reqmin,abstol,...,maxfe) must be set. For an explanation of these parameters see the header of the function *sm_nelmin* in the code module *sm_nelmn.c*.

The next code addition is a section that is used to update the design variables after the optimizer has made an iteration. This section of code is written in the function *sm_objective()*. For the example case, the angle of the expansion ramp is translated to the coordinates of the last section defining the duct. This is done with the coding presented below.

```
/*
 * Apply the design variables.
 */
if ( gd->case_id == OPT_RAMP )
   {
   /*
    * Set the geometry for the single ramp.
    * The design variable is the angle (in radians) of the
    * thrust ramp.
    */
   theta = dv[0];
   xx = duct->PA[1].x + 0.3 * cos(theta);
   yy = -0.3 * sin(theta);
```

```
   /* Modify the downstream plane only. */
   duct->PA[2].x = xx; duct->PA[2].y = yy;
   duct->PB[2].x = xx;
   duct->PC[2].x = xx;
   duct->PD[2].x = xx; duct->PD[2].y = yy;
   }
```

In this case the south face of the duct, (AD), forms the expansion ramp, and the coordinates of the points at the end of the ramp are varied accordingly.

The last part of the coding for an optimization case is to define the objective function. This is done in the last section of the function *sm_objective()*. The objective function is defined so that it evaluates to a lower value as the goal of the optimization is approached. In the case of the expansion ramp example, heat has been added via the source term in the governing equations, and the expansion angle for maximum specific impulse is desired. The coding for such a case might look something like,

```
/*
 * Set the value of the objective function.
 */
if ( gd->case_id == OPT_RAMP )
   {
   /*
    * Single thrust ramp:
    * We want the largest specific impulse (assuming Hydrogen fuel).
    */

   deltaH = 90.0e6;  /* assumed heating value for hydrogen fuel */

   mdot_fuel = slice->Q_total / deltaH; /* mass flow of H_2 fuel  */
   thrust = -(slice->force.x);          /* thrust in Newtons      */

   /* specific impulse (in seconds) to make the objective function */
   impulse = thrust / (mdot_fuel * 9.8);
   obj = (3000.0 - impulse) / 1000.0;

   }
```

In the coding listed above, the value of the objective function will decrease as the specific impulse reaches the target value of 3000 sec. This is the only interaction required by the user to set up an optimization case apart from setting the *case_id* to the correct range to trigger the optimization routines.

6 Test Cases

The following eight test cases are provided both as a guide for the application of *sm_3d*$^+$ and as a demonstration of the capabilities of the code. The test cases have been selected to exercise the new features of the code and to show comparisons with published theoretical and experimental results.

The first test case is a flat plate boundary layer. This is a simple case that demonstrates the code's ability to correctly resolve the laminar boundary layer. Boundary-layer/shock interactions are considered in the second case of hypersonic flow past a compression corner, followed by a cross-flow separation of a boundary layer on the leeward side of a cone at an angle of attack in the third case. The versatility of the B-Spline surfaces is demonstrated in the fourth test case by simulating the flow through a complex three-dimensional scramjet design. In this test case combustion is simulated by adding heat through the energy source term in the governing equations. Combustion is then accurately modeled in the fifth case by using the finite rate chemistry routines. The finite rate kinetics of hydrogen burning in air is simulated in the supersonic flow of a constant area duct. The turbulence model is exercised in the sixth test case by simulating the turbulent two-dimensional flow over a flap. Viscous flow over a cylinder in the seventh test case, tests the axisymmetric implementation of the Parabolized Navier-Stokes equations, and finally, the optimizer is used in the last test case to optimize a simple thrust surface of a scramjet engine.

Each of the eight test casts are briefly described and the parameter file and calculation macros are listed. The detailed information on each simulation can be obtained from the parameter file and macros.

6.1 Flat Plate Boundary Layer

A simple exercise to test the implementation of the PNS equation set for modelling viscous flow, is a two-dimensional flat plate laminar boundary layer as shown in Fig. 20. The case chosen consists of a 1.0 m flat plate aligned with a uniform Mach 2 flow. The gas is considered calorically perfect with $\gamma = 1.4$, R = 287 J/kg/K and a constant Prandtl number of 0.72. This is the PERF_AIR_14 gas model in the main header file. The computational domain, as shown in Fig. 21, is shaped to include the leading-edge interaction shock (LEIS). The domain is divided into $1000 \times 100 \times 2$ cells where the DIMENSION macro has been set to 2. The cells are clustered towards the plate surface with a clustering value of 1.01 and also towards the inlet plane with a clustering value of 1.1 .

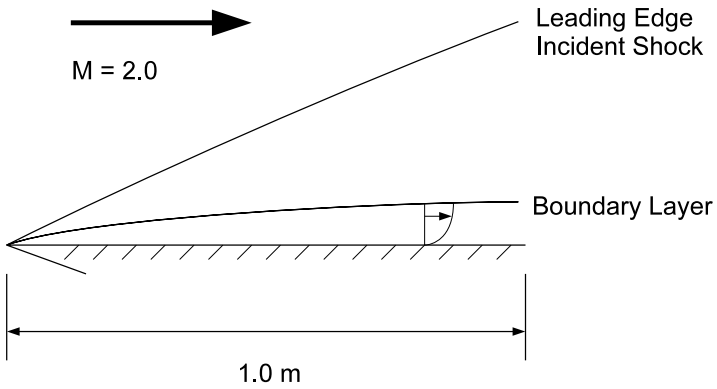

Figure 20: Boundary layer along a flat plate with M = 2.0 and $\mathrm{Re_L} = 1.65 \times 10^5$

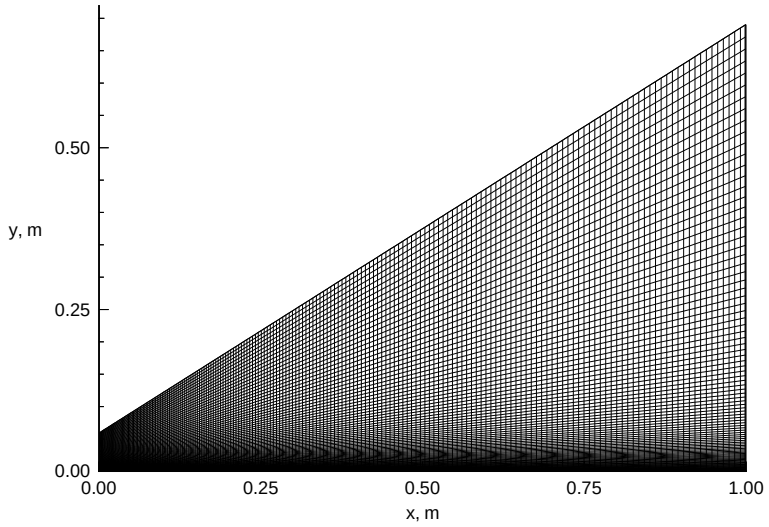

Figure 21: 1000 × 100 mesh joining the cell centres (every 5th cell in the x-direction shown).

The supersonic free-stream conditions of

$$\rho = 0.0404 \text{ kg/m}^3, \quad u_x = 597.3 \text{ m/s}, \quad u_y = u_z = 0.0 \text{ m/s},$$
$$e = 1.592 \times 10^5 \text{ J/kg},$$

were applied at the inlet plane. These conditions correspond to

$$M = 2, \quad Re_L = 1.65 \times 10^5 \text{ and } T = 222 \text{ K} \quad .$$

The South boundary condition, or plate surface, was set to be a no-slip wall with a constant temperature of 222 K. The East and West boundaries were set to solid walls with inviscid (slip) tangency conditions. The correct setting of the East & West boundaries is not essential when the two-dimensional macro is set since they

are automatically set regardless of the user settings. The North boundary is set to a supersonic inflow condition.

All of this data is data is shown in the parameter file as:

```
VISCOUS Flat Plate Boundary Layer Flow
1001                          case_id
0.25   350  0.0001            CFL, max_t_steps, tolerance
2 7 0.8                       Xorder, i_supress, p_safety
0.0  0.001  1.0  1000         Xi_0, dXi, X_max, max_x_steps
0.005                         dXi_plot
1                             slice_ident
100  2                        nny, nnz
0 10                          smooth_grid, smooth_iter
1  3  5  3                    bc_N, E, S, W
222.0 222.0 222.0 222.0       Twall_N, E, S, W
0.0404 597.3 0.0 0.0 1.59285e5 free stream rho, ux, uy, uz, e
0 0                           use_B_spline, bezier_box
2                             np
0.0  0.0  1.0                 0 A
0.0  0.06 1.0                   B
0.0  0.06 0.0                   C
0.0  0.0  0.0                   D
1.0  0.0  1.0                 1 A
1.0  0.7  1.0                   B
1.0  0.7  0.0                   C
1.0  0.0  0.0                   D
```

The macros of the header file are:

```
#define DEBUG          1
#define EXTRAP_TYPE_X  0
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  0
#define FLUX_SPLIT     0
#define DIMENSION      2
#define VISC           1
#define TURB           0
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           1
#define N_RE           1
#define BETA_X         1.1
#define BETA_Y         1.01
#define BETA_Z         0.0
#define X_START        1
#define X_END          0
#define Y_BOTTOM       1
#define Y_TOP          0
#define Z_LEFT         0
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7

#define  TYPE_of_GAS PERF_AIR_14
```

Once the pre-processor *sm_prep.x* was run on the case, the solver was executed and a solution was generated. A contour plot of pressure was created using Tecplot from a slice file generated by *sm_slice.x*. This contour plot is shown in Fig. 22. The

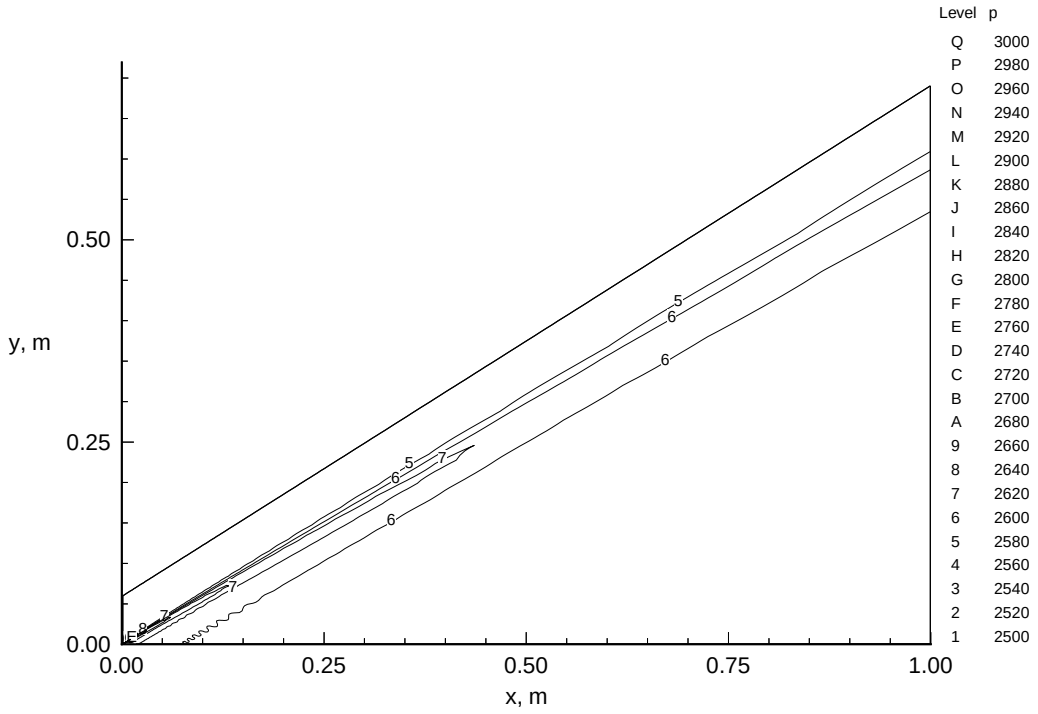

Figure 22: Pressure contours for the flat plate.

only apparent feature is the weak shock propagating into the flow from the leading edge of the plate. However, a boundary layer develops along the plate and attains a total thickness of approximately 5 mm by the end of the plate. Figure 23 compares the x-velocity and temperature profiles at $x = 0.9415$ m with profiles computed by a spectrally-accurate boundary layer code [75]. There is reasonable agreement with the spectral solution which, together with the LEIS resolution, verifies that the numerical scheme correctly captures the weak shock and resolves the viscous and heat fluxes.

For the x-station being considered, the first cell-centre from the wall has $y^+ \approxeq 4.7$ where,

$$y^+ = \frac{y\ \rho_{\text{wall}} \sqrt{\tau_{\text{wall}} / \rho_{\text{wall}}}}{\mu_{\text{wall}}} \quad . \tag{108}$$

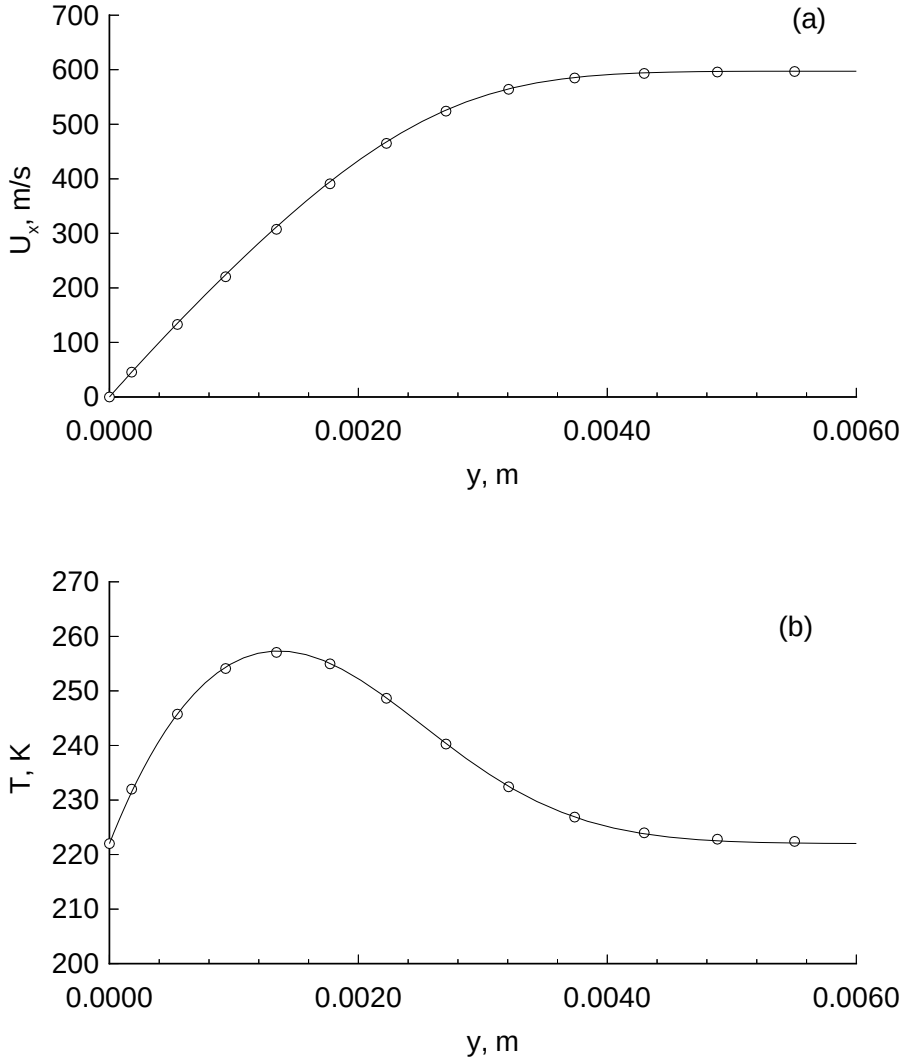

Figure 23: Comparison of the *sm_3d* solution (circles) with a spectral solution (solid line). (a) x-velocity profile at $x = 0.9415$ m; (b) temperature profile at the same location.

6.2 Hypersonic Flow Past a Compression Corner

The second test case involves a two-dimensional shock and boundary layer interaction of hypersonic flow past a compression corner. The problem consists of a flat plate connected to a 15^o ramp with Mach 15 flow in the free-stream as shown in Fig. 24.

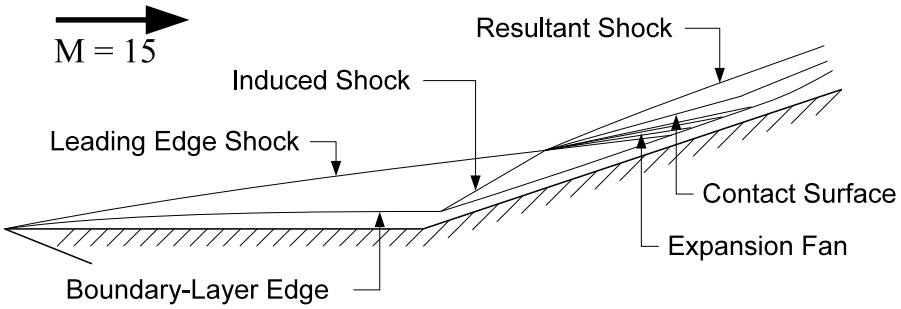

Figure 24: Flow field and grid for a hypersonic compression corner.

The inflow conditions are,

$$\rho_\infty = 4.832 \times 10^{-4} \text{ kg/m}^3, \quad u_x = 2401.56 \text{ m/s}, \quad u_y = u_z = 0.0 \text{ m/s},$$
$$e = 5.18035 \times 10^4 \text{ J/kg}, \quad M = 14.1 \ , \quad Re_L = 1.04 \times 10^5,$$
$$T_\infty = 72.2 \text{ K}, \quad \text{and} \quad T_{\text{wall}} = 297 \text{ K} \ .$$

The horizontal plate length, L, is 0.439m. The 15^o ramp with these flow conditions was studied experimentally by Holden and Moselle (1969) [76]. The distinguishing features of the experimental results were the interaction between the viscous and inviscid flows, and no separation at the base of the ramp. The inviscid flow structure consists of a leading edge shock which intersects the induced shock that is formed by the ramp. The two shocks running to the right intersect to form a single stronger shock, an expansion fan, and a contact surface.

The computational grid consisted of $2000 \times 90 \times 2$ cells that were clustered along the wall and inflow plane. The parameter file and calculation macros are as follows:

```
VISCOUS 2D Hypersonic laminar flow over a 15deg compression ramp
1002                                case_id
0.45 600 0.0001                     CFL, max_t_steps, tolerance
2 10 0.75                           Xorder, i_supress, p_safety
0.0 0.0005 1.0 2000                 Xi_0, dXi, X_max, max_x_steps
0.005                               dXi_plot
1                                   slice_ident
90  2                               nny, nnz
0 0                                 smooth_grid, smooth_iter
1  3  5  3                         bc_N, E, S, W
297.0 297.0 297.0 297.0             Twall_N, E, S, W
4.832e-4 2401.56 0.0 0.0 51803.5 free stream rho, ux, uy, uz, e
```

```
0 0                                  B_spline, bezier_box
4                                 np
0.000 0.0      1.0                0 A
0.000 0.075    1.0                  B
0.000 0.075    0.0                  C
0.000 0.0      0.0                  D
0.439 0.0      1.0                1 A
0.439 0.11495  1.0                  B
0.439 0.11495  0.0                  C
0.439 0.0      0.0                  D
0.730 0.077973 1.0                2 A
0.730 0.179    1.0                  B
0.730 0.179    0.0                  C
0.730 0.077973 0.0                  D
0.900 0.1235   1.0                2 A
0.900 0.203310 1.0                  B
0.900 0.203310 0.0                  C
0.900 0.1235   0.0                  D

#define DEBUG          1
#define EXTRAP_TYPE_X  0
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  0
#define FLUX_SPLIT     0
#define DIMENSION      2
#define VISC           1
#define TURB           0
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           1
#define N_RE           1
#define BETA_X         0.0
#define BETA_Y         1.04
#define BETA_Z         0.0
#define X_START        0
#define X_END          0
#define Y_BOTTOM       1
#define Y_TOP          0
#define Z_LEFT         0
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7

#define  TYPE_of_GAS PERF_AIR_14
```

The intersection of the leading edge shock and the wedge shock, the new combination shock, and the expansion fan are all sharply defined in the contour plots of Fig. 25.

The surface pressure C_p and heat transfer C_h coefficients were calculated along

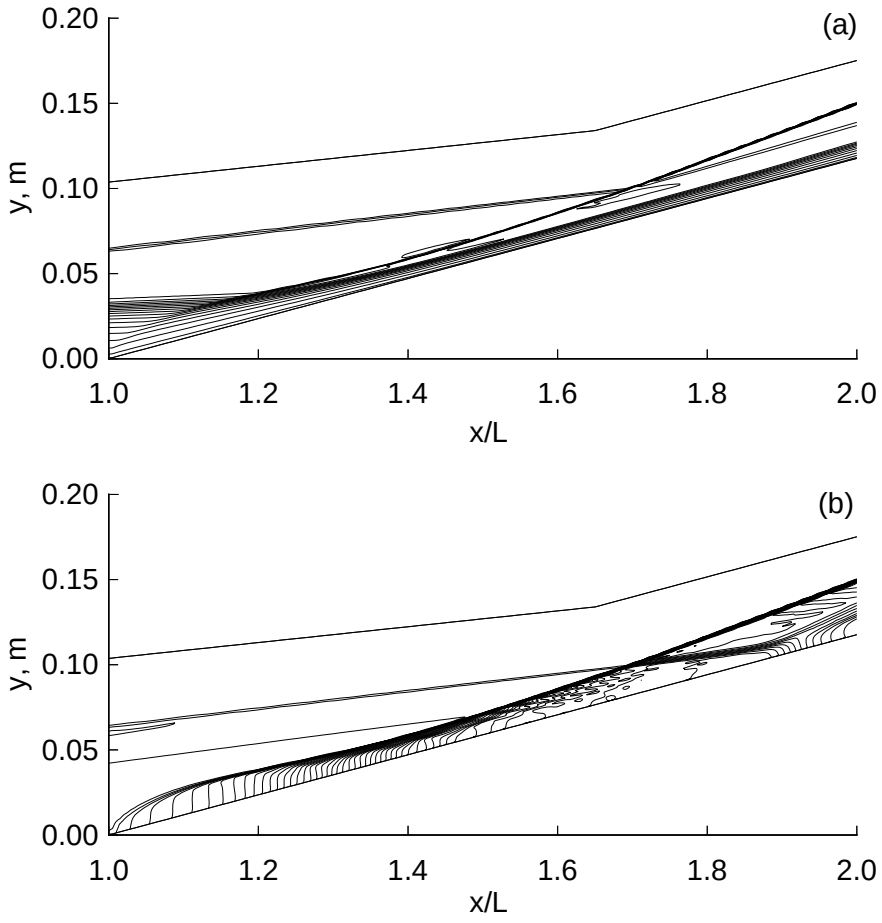

Figure 25: Contours of (a) Mach number and (b) pressure for the 15^o hypersonic compression ramp showing all the flow features described in Fig. 24.

the wall where the coefficients are defined by the relations,

$$C_p = \frac{p}{\frac{1}{2}\rho_\infty u_\infty^2}, \quad \text{and} \quad C_h = \frac{k\frac{\partial T}{\partial y}\sec\theta}{\rho_\infty u_\infty[(e+\frac{p}{\rho}+\frac{u^2}{2})_\infty - (e+\frac{p}{\rho})_{\text{wall}}]} \quad .$$

The slope of the wall where each coefficient is evaluated is equal to the angle θ. The coefficients are plotted in comparison with the experimental data of Holden and Moselle [76] in Fig. 26.

A slight over-prediction of the experimental data is observed which is consistent with other numerical simulations of the same problem [77, 12, 78]. This difference has been shown to be due to a flow misalignment with the ramp in Holden & Moselle's [76] experiment. It is evident from the computed skin friction coefficient,

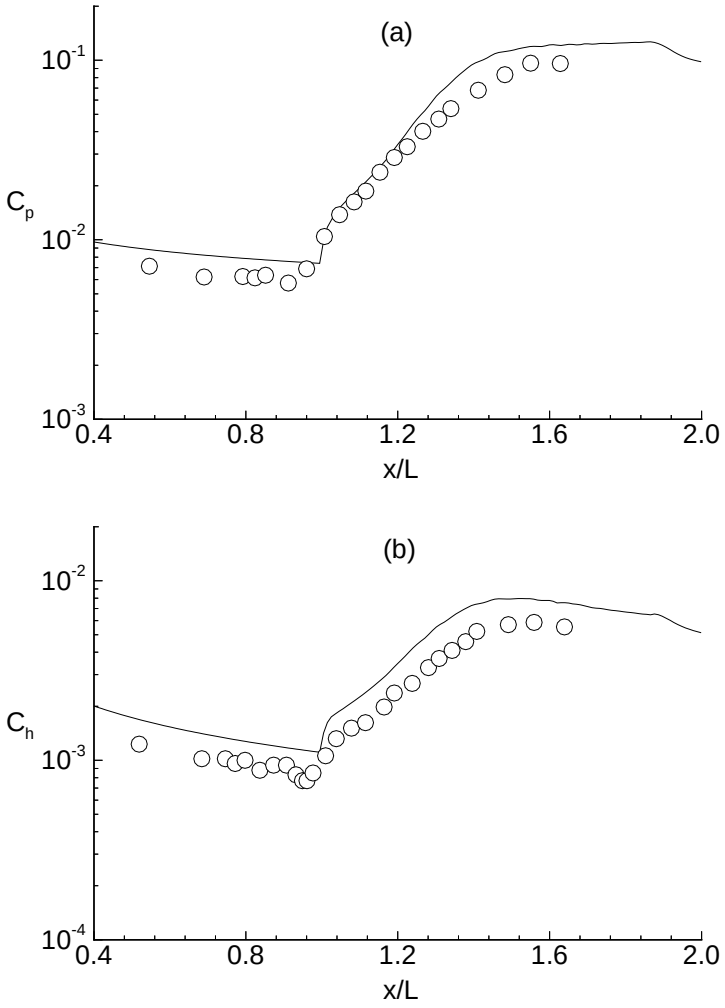

Figure 26: Comparison of (a) computed pressure coefficients and (b) heat transfer coefficients (solid line) with Holden and Moselle's data [76](circles).

that the PNS equations are unable to accurately estimate the correct value in the corner region of the flow. This is due to the inability of the PNS equations to account for the upstream propagation of information in the subsonic region of the ramp corner. The skin friction coefficient does not increase until the ramp is reached. Apart from this inadequacy, the solver produced accurate results before and after the corner.

6.3 Viscous Hypersonic Flow Over a Cone at an Angle of Attack

The experiment performed by Tracy [79] forms the third test case and involves viscous hypersonic flow over a cone at an angle of attack. It consists of a 10^o half angle cone with an axial length of 0.3048 m, at a 24^o angle of attack in Mach 7.95 flow. A diagrammatic representation is given in Fig. 27.

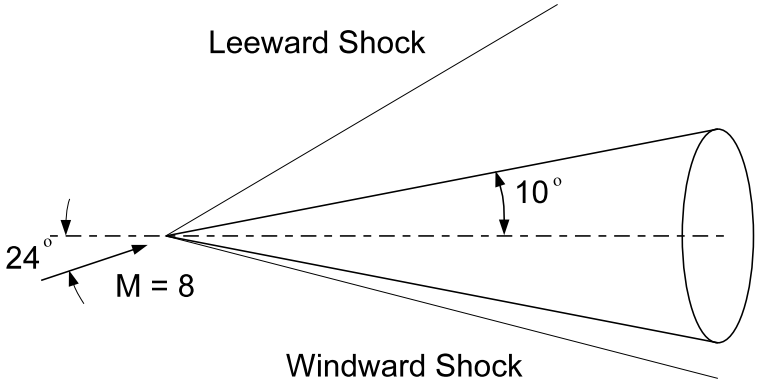

Figure 27: Cone at an angle of attack in a hypersonic flow.

The inflow conditions are,

$$\rho_\infty = 0.01254 \text{ kg/m}^3, \quad u_x = 1083.57 \text{ m/s}, \quad u_y = 482.44 \text{ m/s}, \quad u_z = 0.0 \text{ m/s},$$
$$e = 39.7495 \times 10^3 \text{ J/kg}, \quad M = 7.95, \quad Re_L = 1.25 \times 10^6,$$
$$T_\infty = 55.4 \text{ K}, \quad \text{and} \quad T_{\text{wall}} = 309.8 \text{ K}.$$

The computational problem is set such that the cone is aligned with the x axis and the flow is angled at 24^o to the cone axis in an upward direction (moving in a positive direction along the y-axis). The bottom half of the cone is the windward side and the top half the leeward side.

A complex flow pattern forms around the cone for the high angle of attack and Mach number being studied, making it a challenging problem for the solver. A conical shock forms around the body of the cone and weakens as the flow moves around to the leeward side. On the surface of the cone, a laminar boundary layer forms from the stagnated conditions on the windward side. As the boundary layer develops around the circumference of the cone, it thickens due to the expanding flow that is accelerating to supersonic speeds. The two boundary layers meet at the top of the leeward side where they eventually separate as the flow moves down the length of the cone. The displacement thickness of the separation region grows in size and in doing so moves the leeward side of the conical shock further away from

the surface. A secondary shock also forms around the separation region to provide the transition to subsonic speeds.

The computational grid used for this test case consisted of 50 cells normal to the surface and 56 cells around half of the circumference (from windward symmetry plane to leeward symmetry plane). Twenty thousand axial slices were clustered towards the nose. The cells were also clustered towards the surface. The grid was "wrapped" around half of the surface of the cone such that the outer windward corner of the grid was bounded by a 13^o ray from the nose, and the leeward corner was bounded by a 35^o ray. To avoid the grid discontinuity at the nose of the cone, the simulation was started at an axial position of $x = 0.015$ m. A yz cross-flow plane of the grid at the end of the cone is shown in Fig. 28.

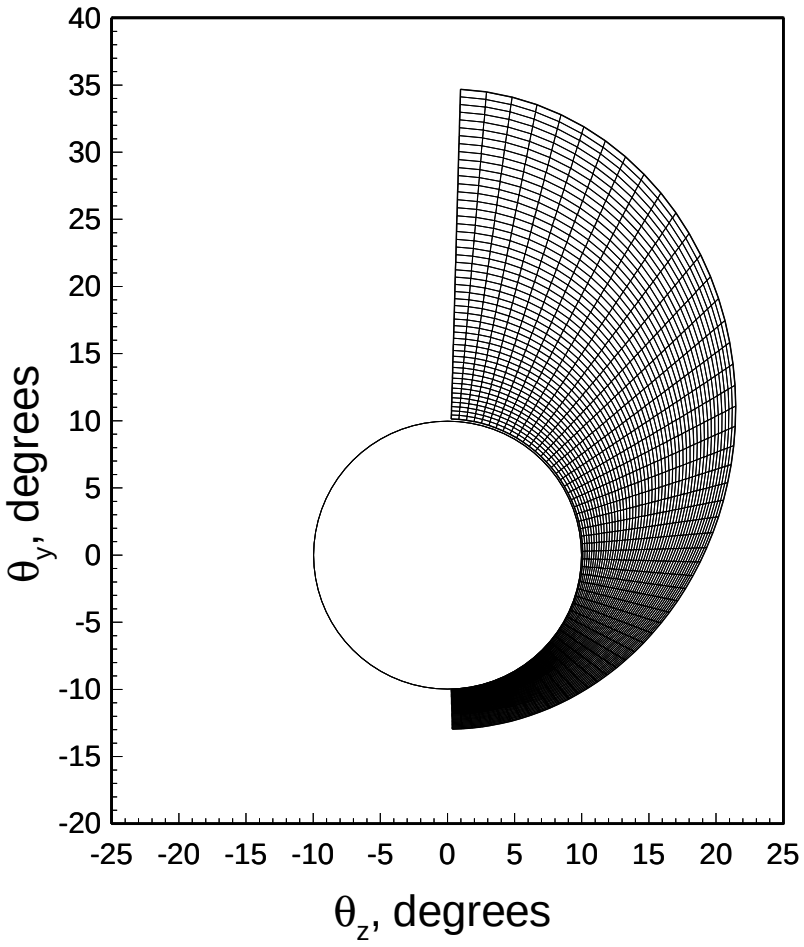

Figure 28: Computational grid for exit cross-flow plane (grid formed from cell centres).

Conical coordinates are used to display the profile of the grid where,

$$\theta_{y,z} = \frac{y,z}{R} \arctan\frac{R}{x} \quad \text{and} \quad R = \sqrt{y^2 + z^2} \quad .$$

The parameter file and calculation macros for the cone problem are as follows:

```
Viscous 10 degree cone from Tracy 1963
1003                                case_id
0.25   300  0.0001                  CFL, max_t_steps, tolerance
2 10 0.75                           Xorder, i_supress, p_safety
0.0  0.00005  1.0  20000             Xi_0, dXi, X_max, max_x_steps
0.005                               dXi_plot
1                                   slice_ident
50 56                               nny, nnz
0 10                                smooth_grid, smooth_iter
5  3  1  3                         bc_N, E, S, W
309.8 309.8 309.8 309.8             Twall_N, E, S, W
0.01254 1083.57 482.44 0.0 39749.5 free stream rho, ux, uy, uz, e
0 0                                 B_spline, bezier_box

#define DEBUG          1
#define EXTRAP_TYPE_X  1
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  1
#define FLUX_SPLIT     0
#define DIMENSION      3
#define VISC           1
#define TURB           0
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           1
#define N_RE           1
#define BETA_X         1.1
#define BETA_Y         1.25
#define BETA_Z         0.0
#define X_START        1
#define X_END          0
#define Y_BOTTOM       0
#define Y_TOP          1
#define Z_LEFT         0
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7

#define  TYPE_of_GAS PERF_AIR_14
```

The surface circumferential pressure distribution at $x/\mathrm{L} = 0.333$ is shown in Fig. 29. Tracy's experimental measurements [79] are also shown for comparison. Reasonable agreement is shown between the computed solution and Tracy's data.

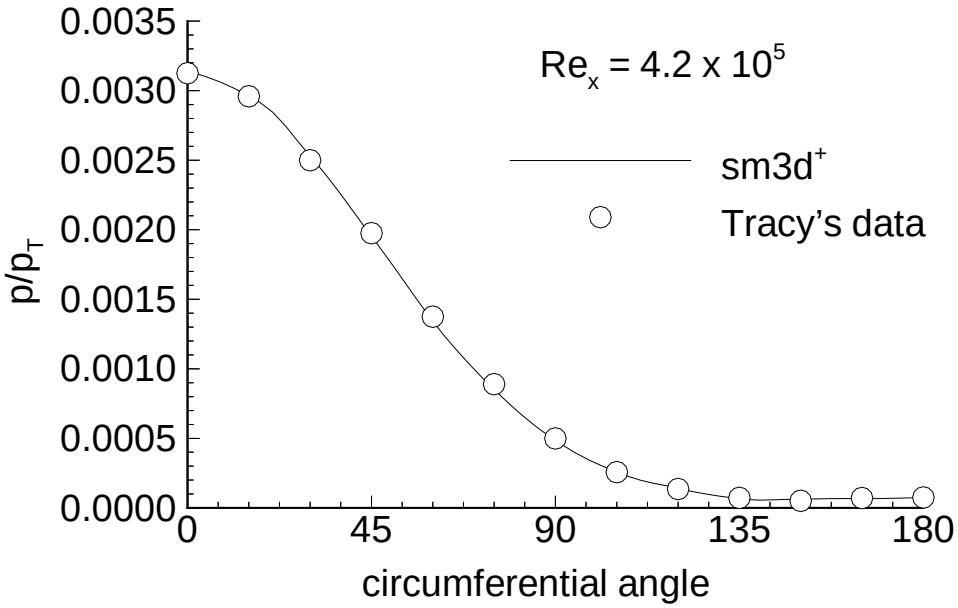

Figure 29: Computed and experimental surface pressures around half the circumference of the cone at an axial position of $x/\mathrm{L} = 0.333$.

The experiments of Tracy [79] also included a cross-sectional Pitot survey of the flow to determine shock wave positions and the edges of boundary layers. The Pitot survey was performed along surface normals at a position 86.3 mm from the apex of the cone. A computational slice was taken in a plane perpendicular to the axis whose axial location bisects the experimental rays. The computational Mach contours on this plane are shown in Fig. 30, along with Tracy's experimental results. The computational results are in good agreement with the experimental results.

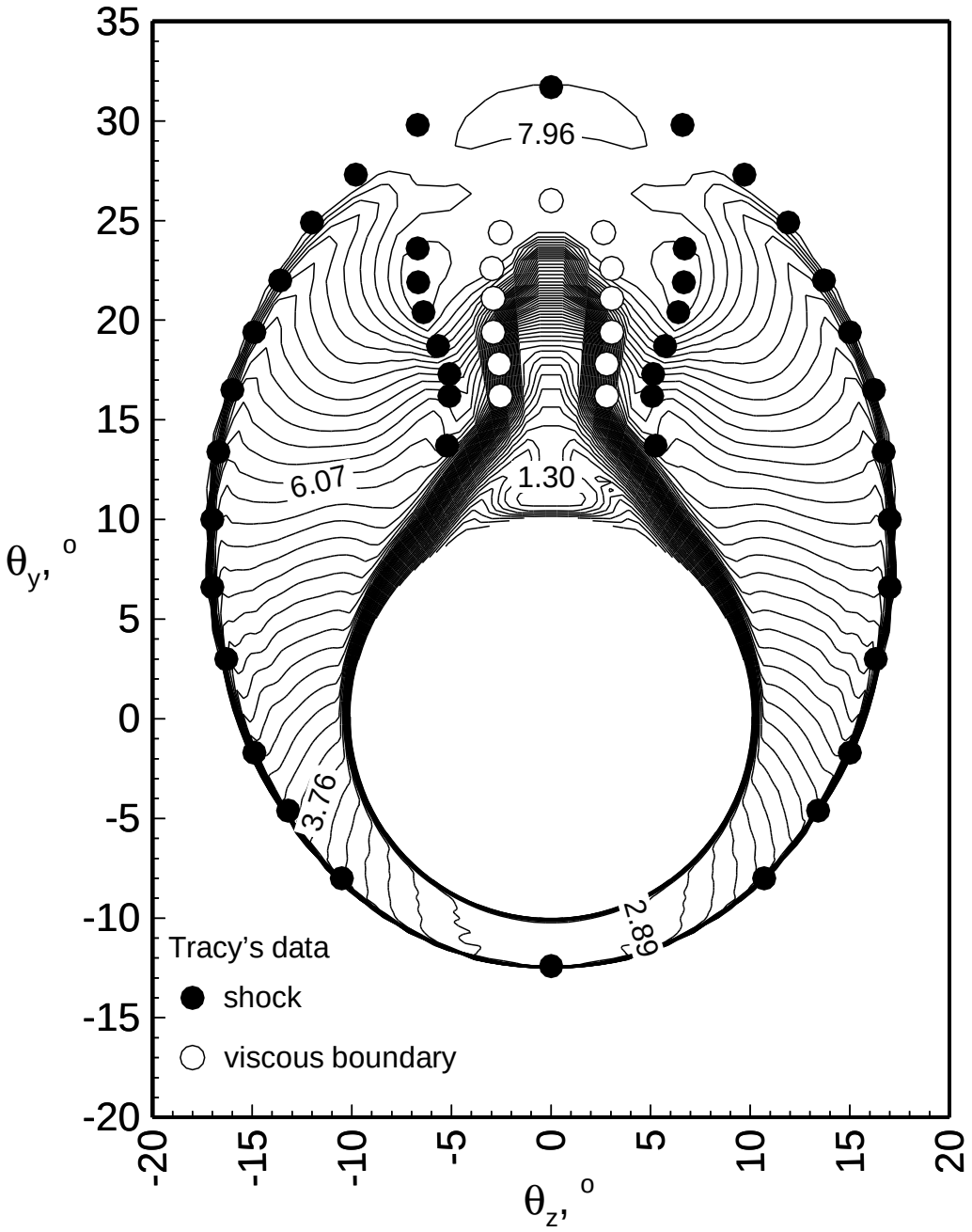

Figure 30: Computed Mach contours at x = 83 mm with Tracy's flow field Pitot survey.

6.4 Flow Through a Three-Dimensional Scramjet

A three-dimensional scramjet design with a complex internal structure is used for the fourth test case. It was chosen because it exercises the codes ability to apply B-Spline surfaces for generating complex grids. The scramjet module being considered for this test case was built and tested [80] in the T4 shock tunnel [81]. The module is one of six that are proposed to be placed circumferentially around a body which has a conical fore-body as shown in Fig. 31. This idea is similar to the SCRAM-MOD-1 missile design proposed by Billig [82].

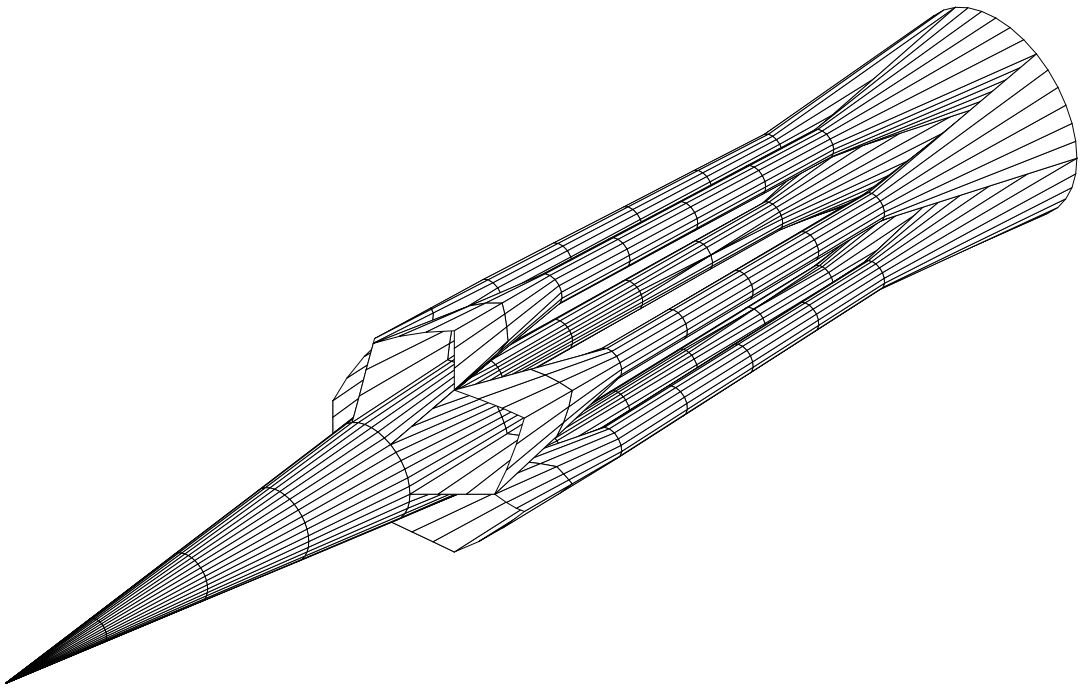

Figure 31: Baseline design for the scramjet-powered stage of a missile. The conical fore-body and the scramjet modules are shown with the cowl removed (pictured supplied by K. Austin).

The complexities of the internal surfaces are a result of the shape transitions in the inlet and expansion duct. The front projection of the inlet occupies a sector of a circle while the rear projection of the inlet is circular. The same applies for the exhaust duct but in reverse. The surfaces required to achieve these transitions are quite complex. Four B-Spline surfaces were fitted to these surfaces using data obtained from the construction plans for the module. The B-Spline surfaces and associated control net files were generated using the software written by Craddock [52]. The control net files were used by *sm_3d*$^+$ to generate the computational grid as shown in Fig. 32.

The inlet flow condition for the simulation has been selected to approximate the experimental flow condition used in Wendt *et al.'s* study [80]. The air test gas is

Figure 32: Exterior surface of the computational grid constructed from B-spline surfaces, used for the three-dimensional scramjet test case. Also shown are the lines on which static pressure was measured.

assumed to be perfect with gas constants $R = 287$ J/(kg.K) and $\gamma = 1.4$. Gas enters the compression inlet with the following properties:

$$P = 10.0 \text{ kPa}, \ \rho = 0.085 \text{ kg/m}^3,$$
$$u = 2375 \text{ m/s at an angle of } 0.5^\circ \text{ to the leading edge surface of the inlet,}$$
$$T = 410 \text{ K, and } \ M = 5.85 \ .$$

Energy is uniformly added to a block of cells to simulate hydrogen combustion. The location of the heating zone is centred on $x = 0.325$ m with a half-length of 0.025 m. Within this heating zone, 0.9 MJ/s is added to the flow giving an effective equivalence ratio of 0.43 (for a nominal, effective fuel heating value of 80 MJ/kg[1]) for gaseous hydrogen.

The input parameter file and macros used are as follows:

```
Composite Scramjet Module with B-spline surface, heat added
1004                                  case_id
0.1  200  0.0001                      CFL, max_t_steps, tolerance
2 3 0.75                              Xorder, i_supress, p_safety
0.0  0.002 1.0  500                   Xi_0, dXi, Xi_max, max_x_steps
0.005                                 dXi_plot
```

[1]This heating value was obtained from a graph in [83] assuming an average combustor temperature of 1800K

```
1                                   slice_ident
20  20                              nny, nnz
1   10                              smooth_grid, smooth_iter
3 3 3 3                             bc_N, E, S, W
296.0 296.0 296.0 296.0             Twall_N, E, S, W
0.0850 2480.0 318.5 0.0 2.942e5     free_stream rho, ux, uy, uz, e
1  0                                use_B_spline, bezier_box

#define DEBUG          1
#define EXTRAP_TYPE_X  0
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  0
#define FLUX_SPLIT     0
#define DIMENSION      3
#define VISC           0
#define TURB           0
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       1
#define N_SP           1
#define N_RE           1
#define BETA_X         0.0
#define BETA_Y         0.0
#define BETA_Z         0.0
#define X_START        0
#define X_END          0
#define Y_BOTTOM       0
#define Y_TOP          0
#define Z_LEFT         0
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7

#define  TYPE_of_GAS PERF_AIR_14
```

Figure 33 compares the simulated and measured wall pressures within the scramjet module with heat addition. The set of experimental data is for a nominal fuel equivalence ratio of approximately 0.5.

Despite a number of differences in detail, the comparison between the experimental data of Wendt *et al.* [80] and the computed pressure levels is reasonable. Viscous and turbulence effects were not included in the simulations. These effects can be expected to influence the data toward the end of the compression inlet ($x \simeq 0.2$ m), within the combustor, and near the beginning of the thrust nozzle ($x \simeq 0.75$ m).

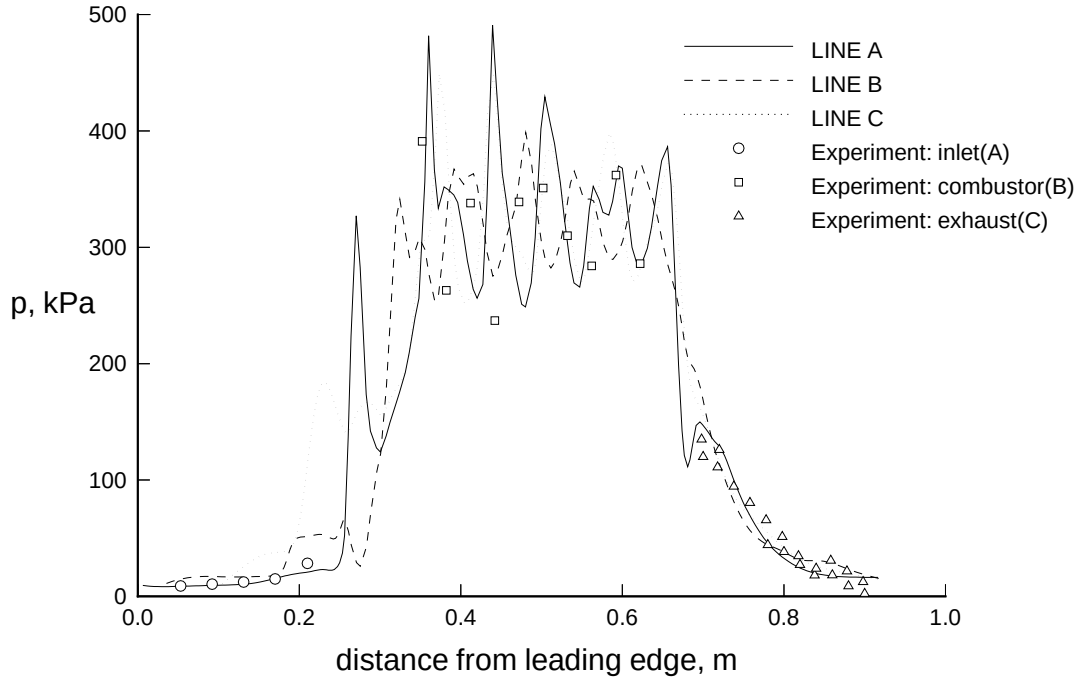

Figure 33: Comparison of simulated and measured wall pressures within the module with heat addition. The symbols denote measured pressures and the lines represent the computational pressures along the lines indicated in Fig. 32.

6.5 Hydrogen Combustion in a Scramjet Combustor

Supersonic hydrogen-air ignition and combustion in a constant area duct (effectively a scramjet combustor), is used as the fifth test case for the finite rate chemistry functions within *sm_3d*$^+$. The test conditions used are the same as those used in the manual for the chemical kinetics code, NASAC, written by Bittker and Scullin [84]. The NASAC code is a fully implicit code that uses set values for heats of reactions, rather than using formation enthalpies to determine how much energy is released or used up in a particular reaction (as is the case in *sm_3d*$^+$).

The air and hydrogen are assumed to be perfectly mixed in a stoichiometric ratio of 1. The initial conditions of the mixture entering the duct are,

$$\begin{aligned} \rho = 0.15628 \text{ kg/m}^3,\ u = 4551.7 \text{ m/s},\ e = 1.711 \times 10^6 \text{ J/kg}, \\ p = 0.9686 \times 10^6 \text{ Pa},\ T = 1559\text{K and } M = 5.04 \quad . \end{aligned} \tag{109}$$

The ignition and combustion process is modelled with 15 possible reaction paths and 9 species [84]. A computational domain of 4000 axial cells, with a minimum of 2 cells in both the η and ζ directions, is used as an approximation of a one-dimensional flow

domain. The cells were spread out over 760 mm to capture the entire combustion process up to its equilibrium state.

The parameter file and macros used for this test case are,

```
Hydrogen combustion in a constant area duct
1005                          case_id
0.1 300 0.0001                CFL, max_t_steps, tolerance
2 1 1.0                       Xorder, i_supress, p_safety
0.0  0.000125  1.0 800        Xi_0, dXi, X_max, max_x_steps
0.002                         dXi_plot
1                             slice_ident
2  2                          nny, nnz
0 0                           smooth_grid, smooth_iter
3  3  3  3                    bc_N, E, S, W
297.0 297.0 297.0 297.0       Twall_N, E, S, W
1.56283e-1 4551.73 0.0 0.0 1.711086e6 free stream rho, ux, uy, uz, e
0 0                           use_B_spline, bezier_box
2                             np
0.0  0.0  2.0                 0 A
0.0  2.0  2.0                   B
0.0  2.0  0.0                   C
0.0  0.0  0.0                   D
0.76 0.0  2.0                 1 A
0.76 2.0  2.0                   B
0.76 2.0  0.0                   C
0.76 0.0  0.0                   D

#define DEBUG          1
#define EXTRAP_TYPE_X  0
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  1
#define FLUX_SPLIT     0
#define DIMENSION      2
#define VISC           1
#define TURB           0
#define CMUTM          0
#define CHEM           1
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           9
#define N_RE           15
#define BETA_X         0.0
#define BETA_Y         0.0
#define BETA_Z         0.0
#define X_START        0
#define X_END          0
#define Y_BOTTOM       0
#define Y_TOP          0
#define Z_LEFT         0
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7
```

The species mass fractions along the duct, calculated using *sm_3d*$^+$, is compared with the results produced by NASAC in Fig. 34. A good comparison between the

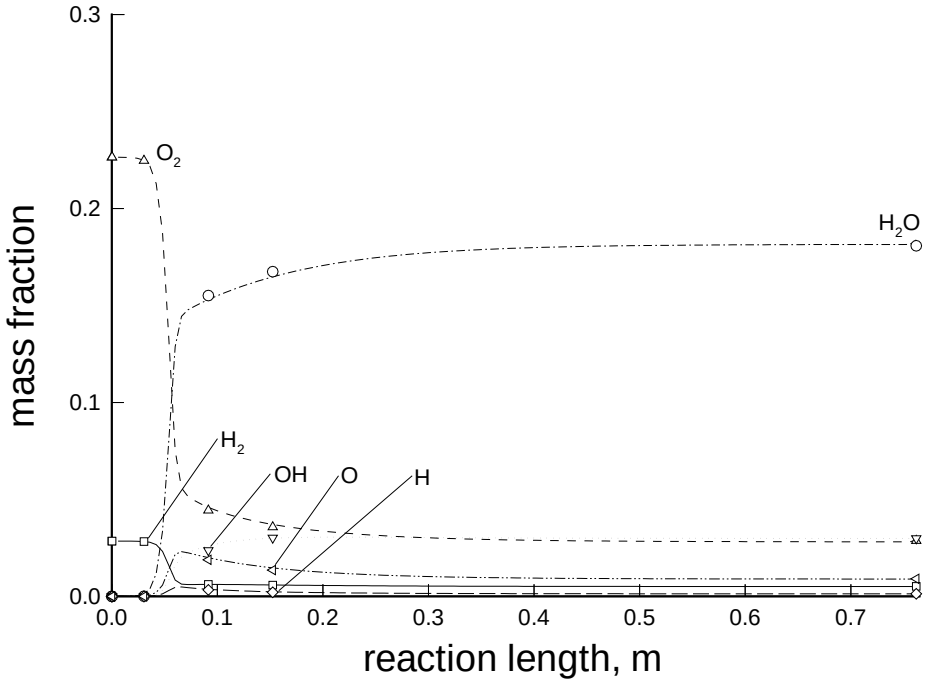

Figure 34: Species mass fractions along a constant area duct resulting from hydrogen combustion in air. The lines represent the results obtained from *sm_3d*$^+$, and the symbols represent results from NASAC.

solutions of the two codes is shown. There is a slight discrepancy, however, in mass fraction just after the ignition zone. This discrepancy is thought to be due to the different schemes used for determining heats of reactions.

The steady state flow variables at the end of the duct are given in Table 4 alongside the inflow conditions for comparison.

Table 4: Inflow and exit, steady state flow variables for the hydrogen combustion test case presented in NASAC [84].

	Inflow Conditions	Exit Steady State Conditions	
		NASAC	*sm_3d*$^+$
u, (m/s)	4551.7	4440.8	4441.6
ρ, kg/m^3	0.15628	0.16018	0.16012
p, kPa	96.8	175.7	175.5
T, K	1559	3016	3014.7
M	5.04	3.78	3.79

6.6 Turbulent Two-Dimensional Flow over a Flap

The sixth test case is from one of the hypersonic, turbulent boundary layer experiments performed by Coleman & Stollery [85]. The experiment being considered consisted of a sharp flat plate with a trailing edge flap as shown in Fig. 35.

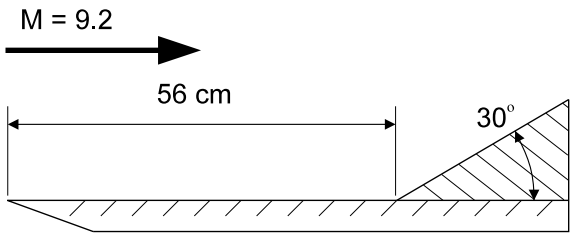

Figure 35: Two-dimensional flat plate and flap used for hypersonic turbulent flow experiments of Coleman & Stollery [85].

The experimental apparatus had a hinged trailing edge flap so that several corner angles could be studied. The current test case is limited to a corner angle of 30^o, which is the largest angle that maintained attached flow in Coleman & Stollerys experiment [85]. The test gas was low temperature air issuing from a hypersonic gun tunnel. The approximated inflow conditions were,

$$\rho = 0.15094 \text{ kg/m}^3,\ u = 1424.87 \text{ m/s},\ e = 4.2648 \times 10^4 \text{ J/kg},$$
$$p = 2.608 \text{ kPa},\ T = 59.4\text{K and } M = 9.22 \quad .$$

The computations were performed assuming a turbulent boundary layer from the leading edge of the plate to the end of the trailing edge flap. The wall temperature was set at a constant 295 K. Due to the strong interaction of the developed boundary layer and the shock emanating from the corner, the computations were performed in stages, using two grids. The first grid was used to simulate the turbulent boundary layer along the flat plate up to the corner. The second grid was used to simulate the flow over the trailing edge flap. This setup allowed axial clustering of the grid at the corner to keep the computations stable. The two grids are shown in Fig. 36.

A clustering parameter of 1.001 was used for cells next to the wall in both grids. This heavy clustering ensured that the y^+ value on the wall never went above 7.3. Fully upwind extrapolation for the *XFaces* was used for this test case because upwind biased extrapolation produced large oscillations in the heat transfer values along the wall.

The parameter file macros used for this test case are,

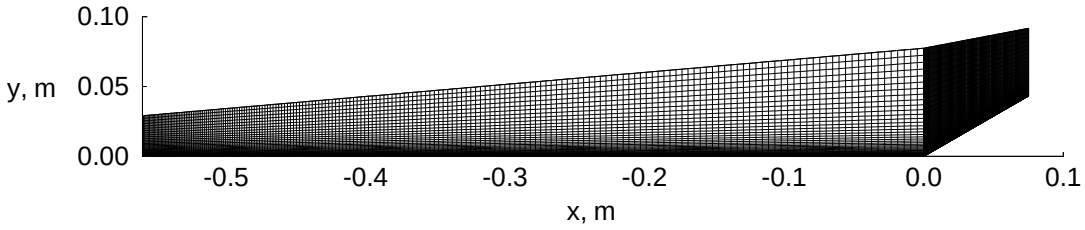

Figure 36: Two-dimensional grids used for the hypersonic turbulent two-dimensional flow over a flap test case.

```
Turbulent Hypersonic Boundary flow over a 30 deg Flap - str
1006                          case_id
0.4 400 0.0001                CFL, max_t_steps, tolerance
2 5 0.75                       Xorder, i_supress, p_safety
0.0  0.0001  1.0 10000        Xi_0, dXi, X_max, max_x_steps
0.005                         dXi_plot
1                             slice_ident
60  2                          nny, nnz
0 0                           smooth_grid, smooth_iter
1  3  5  3                    bc_N, E, S, W
295.0 295.0 295.0 295.0       Twall_N, E, S, W
0.1529 1424.9 0.0 0.0 42648.2 free stream rho, ux, uy, uz, e
0 0                           use_B_spline, bezier_box
2                             np
-0.56  0.0  1.0            0 A
-0.56  0.03 1.0              B
-0.56  0.03 0.0              C
-0.56  0.0  0.0              D
0.0  0.00  1.0             1 A
0.0  0.08  1.0               B
0.0  0.08  0.0               C
0.0  0.00  0.0               D

#define DEBUG          1
#define EXTRAP_TYPE_X  1
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  1
#define FLUX_SPLIT     0
#define DIMENSION      2
#define VISC           1
#define TURB           1
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           1
#define N_RE           1
#define BETA_X         1.1
#define BETA_Y         1.001
#define BETA_Z         0.0
#define X_START        1
#define X_END          0
#define Y_BOTTOM       1
#define Y_TOP          0
#define Z_LEFT         0
```

```
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7

#define  TYPE_of_GAS PERF_AIR_14

-----------------------------------------------------------------
Turb. Hypersonic Boundary flow over a 30 deg Flap - ramp section
1006                          case_id
0.25  2000  0.0001            CFL, max_t_steps, closing tolerance
2 5 0.75                      Xorder, i_supress, p_safety
0.0  0.00001  1.0  100000     Xi_0, dXi, X_max, max_x_steps
0.005                         dXi_plot
1                             slice_ident
60  2                         nny, nnz
0 10                          smooth_grid, smooth_iter
1  3  5  3                    bc_N, E, S, W
295.0 295.0 295.0 295.0       Twall_N, E, S, W
0.1529 1424.9 0.0 0.0 42648.2 free stream rho, ux, uy, uz, e
0 0                           use_B_spline, bezier_box
2                             np
-2.344262e-05  0.00  1.0      0 A
-2.344262e-05  0.08  1.0        B
-2.344262e-05  0.08  0.0        C
-2.344262e-05  0.00  0.0        D
0.075  0.0433  1.0            1 A
0.075  0.0933  1.0              B
0.075  0.0933  0.0              C

#define DEBUG          1
#define EXTRAP_TYPE_X  1
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  1
#define FLUX_SPLIT     0
#define DIMENSION      2
#define VISC           1
#define TURB           1
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           1
#define N_RE           1
#define BETA_X         0.0
#define BETA_Y         1.001
#define BETA_Z         0.0
#define X_START        0
#define X_END          0
#define Y_BOTTOM       1
#define Y_TOP          0
#define Z_LEFT         0
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7

#define  TYPE_of_GAS PERF_AIR_14
```

Figure 37 compares the experimental results of Coleman & Stollery[85] and the

computational results. The surface heat transfer coefficient is nondimensionalized by the coefficient value at a point just upstream of the corner. The pressure coefficient is also nondimensionalized by the value just upstream of the corner. The coefficient of pressure is defined by the expression,

$$C_p = \frac{p}{\rho_\infty u_\infty^2} \quad ,$$

and the heat transfer coefficient is defined by the same expression used for the compression corner case. The calculated pressure coefficients on the flap show reasonable agreement with the experimental values. The computational results approach the expected pressures of an inviscid solution.

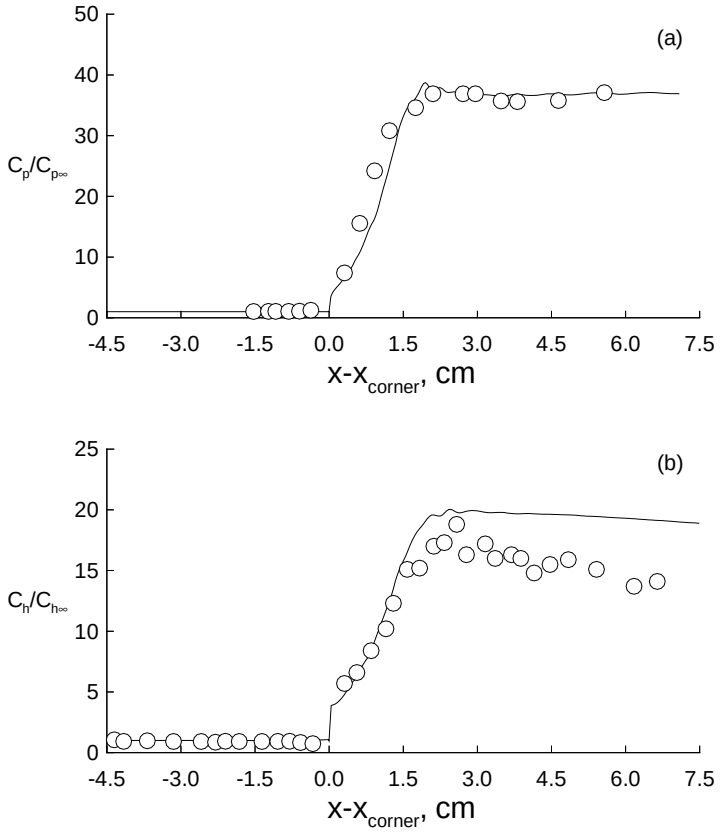

Figure 37: Compression corner flow with 30^o flap angle. The two plots show the experimental data of Coleman & Stollery [85](symbols) compared with the computational results (lines) for (a) surface pressure coefficient and (b) surface heat transfer coefficient.

The computational heat transfer values as predicted using the Baldwin & Lomax turbulence model [5] are slightly higher than those measured in the experiment of Coleman & Stollery [85]. This is consistent with other simulations performed using the same turbulence model [6, 7].

6.7 Viscous Flow over a Cylinder

The implementation of the axisymmetric viscous terms is examined by computing the supersonic, viscous flow over a hollow cylinder. This case is similar to the flat plate boundary layer test case except for the axisymmetric geometry. The flow geometry consists of a hollow cylinder aligned with the x-axis. The cylinder is 1.0 m long and has a radius of 0.005 m. A uniform Mach 2 flow of air is used as the free stream. The air is assumed to be calorically perfect with $\gamma = 1.4$, $R = 287$ J/kg/K and a constant Prandtl number of 0.72. The grid is shaped to capture the leading edge incident shock (LEIS) and is divided into $2000 \times 50 \times 2$ cells (see Fig. 38).

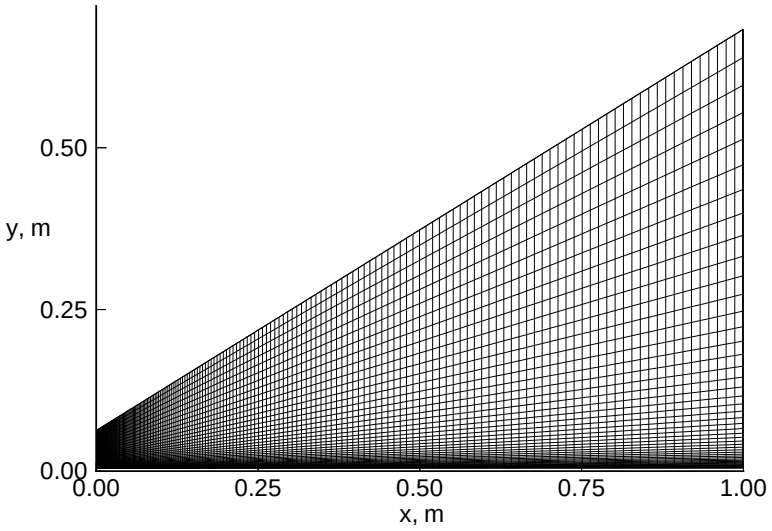

Figure 38: Mesh joining cell centres, that is used for the the boundary layer along a cylinder test case (only every tenth axial cell shown).

The South boundary, which coincides with the cylinder surface, is set to a non-slip boundary condition with a constant temperature of $T_{\text{wall}} = 222.0$ K. The top North boundary is supersonic inflow, and the remaining East and West boundaries are reflective boundaries. The free stream inflow conditions and calculation macros can be obtained from the parameter file and macro listing that follows.

```
Viscous flow along a cylinder
1007                          case_id
0.4   400  0.0001             CFL, max_t_steps, tolerance
2 3 0.75                      Xorder, i_supress, p_safety
0.0  0.0005  1.0  2000        Xi_0, dXi, X_max, max_x_steps
0.005                         dXi_plot
1                             slice_ident
50  2                         nny, nnz
0 10                          smooth_grid, smooth_iter
1  3  5  3                    bc_N, E, S, W
222.0 222.0 222.0 222.0       Twall_N, E, S, W
```

```
0.00404 597.3 0.0 0.0 1.59285e5 free stream rho, ux, uy, uz, e
0 0                             use_B_spline, bezier_box
2                               np
0.0  0.005  2.0                 0 A
0.0  0.065  2.0                   B
0.0  0.065  0.0                   C
0.0  0.005  0.0                   D
1.0  0.005  2.0                 1 A
1.0  0.705  2.0                   B
1.0  0.705  0.0                   C
1.0  0.005  0.0                   D

#define DEBUG          1
#define EXTRAP_TYPE_X  0
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  0
#define FLUX_SPLIT     0
#define DIMENSION      0
#define VISC           1
#define TURB           0
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           1
#define N_RE           1
#define BETA_X         1.01
#define BETA_Y         1.004
#define BETA_Z         0.0
#define X_START        1
#define X_END          0
#define Y_BOTTOM       1
#define Y_TOP          0
#define Z_LEFT         0
#define Z_RIGHT        0
#define NDVMAX         20
#define VOLUME_TOL     1.0e-7

#define  TYPE_of_GAS PERF_AIR_14
```

The weak LEIS is shown in the contour plot of Fig. 39. Figure 40 compares the *sm_3d*$^+$ x-velocity and temperature profiles through the boundary layer at an axial location of $x = 0.916$ m, with the profiles calculated using a spectrally-accurate boundary layer code [75]. The agreement is quite good for such a coarse grid.

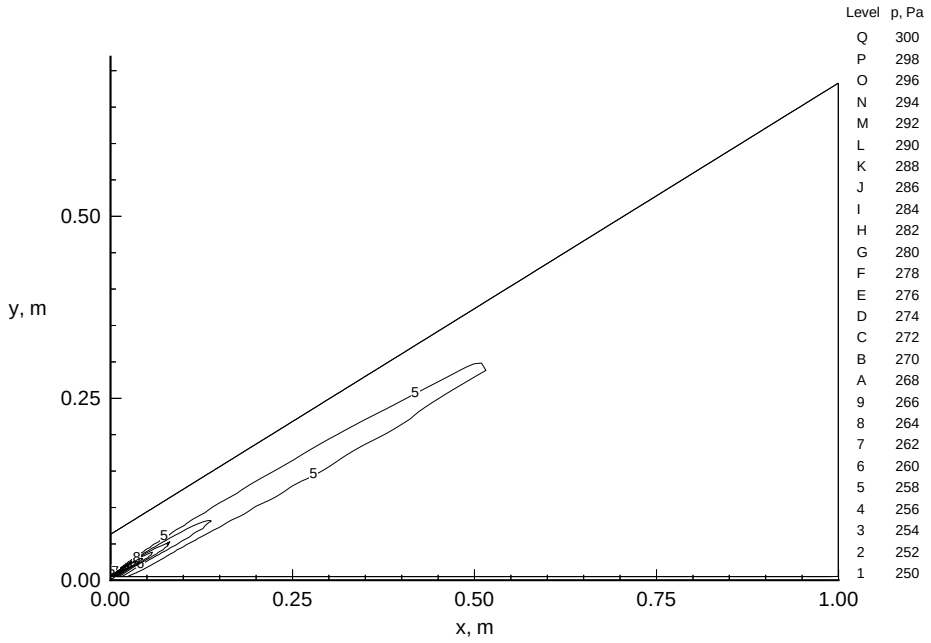

Figure 39: Pressure contours.

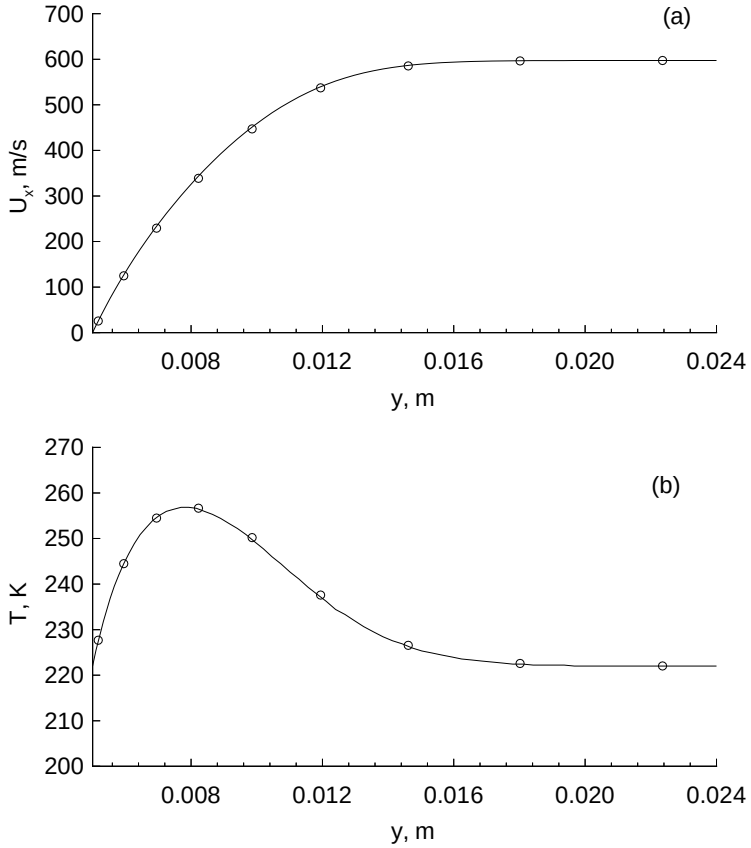

Figure 40: The cross-stream (a) x-velocity profile and (b) temperature profile at an axial location of $x = 0.916$ m. The *sm_3d*$^+$ solution is shown as symbols and a spectral solution [75] is shown as solid lines.

6.8 Optimization of a Scramjet Exhaust Nozzle

For an optimization example, a simple two-dimensional scramjet combustor and exhaust nozzle is considered (see Fig. 41).

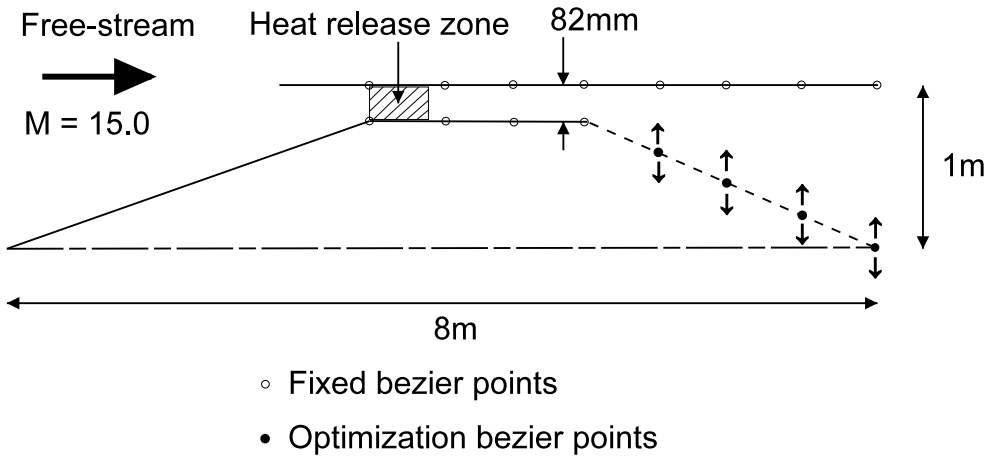

Figure 41: Two-dimensional scramjet for optimization test case.

The inflow to the combustor is the compressed air from the straight inlet compression surfaces. The free-stream flow is Mach 15 air at a static pressure of 1144 Pa and temperature of 218 K. This is roughly equivalent to the ambient conditions at an altitude of 30 km [86].

The resulting combustor entry conditions are,

$$\rho = 0.23495 \text{ kg/m}^3,\ u = 4139.0 \text{ m/s},\ e = 1.0759 \times 10^6 \text{ J/kg},$$
$$p = 101.1 \text{ kPa},\ T = 1500\text{K and } M = 5.3 \ .$$

Heat is uniformly added to the combustor at the rate of 203 MJ/s from the start of the combustor to a point 300mm downstream. This results in an approximate stoichiometric fuel to air mixture if we assume that hydrogen combustion reactions give up 90 MJ/kg of hydrogen fuel when burned to completion at 2000K [21].

The parameter file and macros used for this problem are listed below. Note that the "case id" number triggers the optimization routines.

```
2D Scramjet optimization test case (uniform heat addition)
1008                          case_id
0.3   200  0.0001             CFL, max_t_steps, closing tolerance
2 3 0.75                      Xorder, i_supress, p_safety
0.0  0.00125  1.0  800        Xi_0, dXi, X_max, max_x_steps
0.01                          dXi_plot
1                             slice_ident
40  2                         nny, nnz
0 10                          smooth_grid, smooth_iter
3  3  3  3                    bc_N, E, S, W
```

```
296.0 296.0 296.0 296.0      Twall_N, E, S, W
0.23495 4139.0 0.0 0.0 1.0759e6 free stream rho, ux, uy, uz, e
0 1                          use_B_spline, bezier_box
8                            np
4.923 0.91806  1.0           0 A
4.923 1.0  1.0                 B
4.923 1.0  0.0                 C
4.923 0.91806  0.0             D
5.190 0.91806  1.0           1 A
5.190 1.0  1.0                 B
5.190 1.0  0.0                 C
5.190 0.91806  0.0             D
5.456 0.91806 1.0            2 A
5.456 1.0 1.0                  B
5.456 1.0 0.0                  C
5.456 0.91806 0.0              D
5.723 0.91806 1.0            3 A
5.723 1.0 1.0                  B
5.723 1.0 0.0                  C
5.723 0.91806 0.0              D
6.3535 0.688545 1.0          4 A
6.3535 1.0 1.0                 B
6.3535 1.0 0.0                 C
6.3535 0.688545 0.0            D
6.984 0.45903 1.0            5 A
6.984 1.0 1.0                  B
6.984 1.0 0.0                  C
6.984 0.45903 0.0              D
7.6145 0.229515 1.0          6 A
7.6145 1.0 1.0                 B
7.6145 1.0 0.0                 C
7.6145 0.229515 0.0            D
8.245 0.0 1.0                7 A
8.245 1.0 1.0                  B
8.245 1.0 0.0                  C
8.245 0.0 0.0                  D

#define DEBUG          1
#define EXTRAP_TYPE_X  0
#define INTERP_TYPE_YZ 0
#define CFL_STRINGENT  0
#define FLUX_SPLIT     0
#define DIMENSION      2
#define VISC           0
#define TURB           0
#define CMUTM          0
#define CHEM           0
#define FROZEN         0
#define ADD_HEAT       0
#define N_SP           1
#define N_RE           1
#define BETA_X         0.0
#define BETA_Y         0.0
#define BETA_Z         0.0
#define X_START        0
#define X_END          0
#define Y_BOTTOM       0
#define Y_TOP          0
```

```
#define Z_LEFT          0
#define Z_RIGHT         0
#define NDVMAX          20
#define VOLUME_TOL      1.0e-7

#define  TYPE_of_GAS PERF_AIR_14
```

The optimization problem consists of finding the position of the optimization bezier points (as shown in Fig. 41) that result in the maximum specific impulse for the thrust surface. The specific impulse is defined as,

$$I_{sp} = \frac{F_x}{\dot{m}_{\text{fuel}}\ 9.81} \quad ,$$

where F_x is the axial force developed by the thrust surface, and $\dot{m}_{\text{fuel}}$ is the mass flow of hydrogen fuel. A constraint is placed on the problem, which requires the static pressure at the ramp tip to remain greater than a fixed percentage of the free-stream pressure such that no reverse flow occurs. The objective function that incorporates this constraint can be written as,

$$obj = \begin{cases} (2000.0 - I_{sp})/1000.0 & \text{if } p_{\text{tip}} > 0.67 p_\infty \quad , \\ (2000.0 - I_{sp})/1000.0 + ((\frac{2}{3}\ p_\infty / p_{\text{tip}}) - 1.0) & \text{if } p_{\text{tip}} < 0.67 p_\infty \quad . \end{cases}$$

The optimization bezier points are free to move in the y-direction, while the x-coordinates are constrained to maintain the length of the vehicle. Initially, the four optimization bezier points are distributed linearly between the last point in the combustor and the axis at $y = 0.0$. An initial perturbation of 8 mm is given to the design points to form the initial simplex.

The optimization routine used 104 iterations to converge to a design with an "iteration to iteration" objective function difference of 1.0×10^{-6}. The resulting design is shown in Fig. 42 with contours of constant pressure. The profile of the nozzle is very similar to the classical "bell" shaped nozzles seen in rocket engines [87]. An improvement of 9% in specific impulse was achieved over the initial bezier design. The initial bezier design had an $I_{sp} = 1551$ sec, and the optimized design had an $I_{sp} = 1699$ sec (note that this is the nozzle I_{sp}, not the vehicle I_{sp}). The static pressure at the tip of the thrust surface is 15 times the free-stream pressure and the static pressure at the cowl tip is 6 times the free-stream pressure. To prevent back-flow on the outer part of the cowl, an expansion surface may be required at the end of the cowl. Such is the case for the baseline design used by Baysal *et al.* [64]. They used a different optimization technique and arrived at a similar shaped nozzle with the classical "bell" shape.

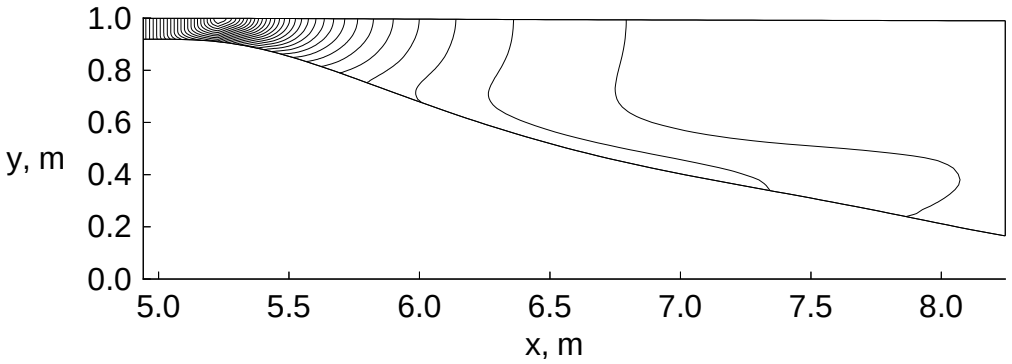

Figure 42: Optimized design for maximum specific impulse with pressure contours.

6.9 Test Case Computation Times and Speeds

The final stages in the development of *sm_3d*$^+$ were carried out on The University of Queensland's Silicon Graphics Origin2000, which had at this time 64 R10000 processors. To give an indication of the speed at which the code runs at on this system, all of the test cases were timed and the results presented in Table 5.

Table 5: Computation times and speeds for all of the test cases

Case Name	Mesh Size	CPU Time (sec)	Solver Speed (μs/cell/step)
Flat Plate	$1000 \times 100 \times 2$	707.5	24.0
Compression Corner	$2000 \times 90 \times 2$	2095.8	22.7
Cone at Angle of Attack	$1000 \times 50 \times 56$	771780	24.8
3-D Scramjet	$500 \times 20 \times 20$	176.5	21.9
Hydrogen Combustion	$8000 \times 2 \times 2$	594.4	29.7
Turbulent Flap - plate	$10000 \times 60 \times 2$	3365.2	23.8
Turbulent Flap - flap	$100000 \times 60 \times 2$	11379.2	25.5
Viscous Cylinder	$2000 \times 50 \times 2$	186.2	16.9
Scramjet Optimization	$800 \times 40 \times 2$	1794.5	

7 Concluding Remarks

This report has described a new version of the original space-marching flow solver *sm_3d*[1]. The new solver, titled *sm_3d*$^+$, is capable of modelling viscous, turbulent, multi-species reacting flow in thermodynamic equilibrium. Using the optimization routines provided with the solver, it can now be used for complete scramjet design studies where there are complex interactions between chemical and viscous effects. This was one of the main objectives for the development of the solver.

The development of *sm_3d* has now reached a level where it can be used to model a whole host of hypersonic flow problems as demonstrated in the Test Case Section of this report. However, there is still room for improvement. Further extensions can be made to the solver in the future by implementing molecular diffusion, thermodynamic non-equilibrium and multiple flow paths. These topics will be the subject of future projects and their reports.

References

[1] Jacobs, P. A., 'A space-marching Euler solver for supersonic flow in a three-dimensional geometry.' Department of Mechanical Engineering Report 2/95, The University of Queensland, Brisbane, Australia, January 1995.

[2] Jacobs, P. A. and Craddock, C. S., 'Simulation and optimisation of three-dimensional flows in scramjet ducts.' *Twelfth International Symposium on Air Breathing Engines.*, AIAA, Washington, DC., September 1995.

[3] Anderson, D. A., Tannehill, J. C., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, New York, 1984.

[4] Craddock, C. S., 'A quasi-one-dimensional space-marching flow solver with finite rate chemical effects,' Department of Mechanical Engineering Report 7/96, The University of Queensland, Brisbane, October 1996.

[5] Baldwin, B. S. and Lomax, H., 'Thin layer approximation and algebraic model for separated turbulent flows,' Paper 78-257, AIAA, 1978.

[6] Horstman, C., 'Prediction of hypersonic shock-wave/turbulent-boundary- layer interaction flows,' AIAA Paper 87-1367, AIAA, 1987.

[7] Vuong, S. and Coakley, T., 'Modeling of turbulence for hypersonic flows with and without separation.' AIAA Paper 87-0286, AIAA, 1987.

[8] Nelder, J. A. and Mead, R., 'A simplex method for function minimization.' *Computer Journal*, Vol. 7, 1965, pp. 338–345.

[9] O'Neill, R., 'Algorithm AS47. Function minimization using a simplex algorithm.' *Applied Statistics*, Vol. 20, 1971, pp. 338–345.

[10] Olsson, D. M. and Nelson, L. S., 'The Nelder-Mead simplex procedure for function minimization.' *Technometrics*, Vol. 17, No. 1, 1975, pp. 45–51.

[11] Srinivas, K., 'An explicit spatial marching algorithm for Navier-Stokes equations.' *Computers and Fluids*, Vol. 21, No. 2, 1992, pp. 291–299.

[12] Korte, J. J., 'An explicit, upwind algorithm for solving the parabolized Navier-Stokes equations.' Ph. D. Dissertation, North Carolina State University, 1989.

[13] Rudmann, S. and Rubin, F., 'Hypersonic viscous flow over slender bodies with sharp leading edges .' *AIAA Journal*, Vol. 6, 1968, pp. 1883–1890.

[14] Tannehill, J., Venkatapathy, E., and Rakich, J., 'Numerical solution of supersonic viscous flow over blunt delta wings.' *AIAA Journal*, Vol. 20, No. 2, 1982, pp. 203–210.

[15] Tannehill, J. C., Anderson, D. A., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer: Second Edition*, Taylor & Francis Publishers, Washington, DC, 1997.

[16] Lighthill, M., 'On boundary layers and upstream influence II. Supersonic flows without separation.' *Proceedings of the Royal Society, Series A, Vol. 217*, 1953.

[17] Lin, T. and Rubin, S., 'A numerical model for supersonic viscous flow over a slender reentry vehicle .' AIAA Paper 79-0205, AIAA, 1979.

[18] Vigneron, Y. C., Rakich, J. V., and Tannehill, J. C., 'Calculation of supersonic viscous flow over delta wings with sharp leading edges.' Paper 78-1137, AIAA, 1978.

[19] Davis, R., Barnett, M., and Rakich, J., 'The calculation of supersonic viscous flows using the parabolized Navier-Stokes equations.' *Computers and Fluids*, Vol. 14, 1986, pp. 197–224.

[20] Shirazi, S. and Truman, C., 'Evaluation of algebraic turbulence models for PNS predictions of supersonic flow past a sphere-cone.' *AIAA Journal*, Vol. 27, No. 5, 1989, pp. 560–568.

[21] Chase, M. W., *JANAF Thermochemical Tables*, American Institute of Physics, New York, N.Y., 1985.

[22] McBride, B., Gordon, S., and Reno, M., 'Thermodynamic data for fifty reference elements.' Technical Paper 3287, NASA, 1993.

[23] Oldenborg, R. and *et al.*, 'Hypersonic combustion kinetics. status report of the rate constant committee, NASP high-speed propulsion technology team.' NASP Technical Memorandum 1107, NASA, 1990.

[24] Grossman, B. and Cinnella, P., 'Flux-split algorithms for flows with non-equilibrium chemistry and vibrational relaxation,' *Journal of Computational Physics*, Vol. 88, No. 1, 1990, pp. 131.

[25] Prabhu, R. and Erickson, W., 'A rapid method for the computation of equilibrium chemical composition of air to 15000K.' Technical Paper 2792 2792, NASA, 1988.

[26] Srinivasan, S., Tannehill, J. C., and Weilmuenster, K. J., 'Simplified curve fits for the thermodynamic properties of equilibrium air.' Reference Publication 1181, NASA, 1987.

[27] Jachimowski, C. J., 'An analytical study of the hydrogen-air reaction mechanism.' Technical Paper 2791, NASA, 1988.

[28] Park, C., Howe, J., Jaffe, R., and Candler, G., 'Review of chemical-kinetic problems of future NASA missions, II: Mars entries.' *AIAA Journal of Thermophysics and Heat Transfer*, Vol. 8, No. 1, 1994, pp. 9–23.

[29] Sutherland, W., 'The viscosity of gases and molecular force.' *Phil. Mag.*, Vol. 5, 1983, pp. 507–531.

[30] Hilsenrath, J., *Tables of Thermodynamic and Transport Properties. National Bureau of Standards. Circular 564; reprinted 1960*, Pergamon, New York, 1955.

[31] Chapman, S. and Cowling, T. G., *The Mathematical Theory of Nonuniform Gases, 3rd Edition.*, Cambridge University Press, London, 1970.

[32] Wilke, C. R., 'A viscosity equation for gas mixtures.' *Journal of Chemical Physics*, Vol. 18, 1950, pp. 517–519.

[33] Hirschfelder, J., Curtiss, C., and Bird, R., *Molecular Theory of Gases and Liquids.*, John Wiley & Sons, New York, 1954.

[34] Neufeld, P., Janzen, A., and Aziz, R., 'Empirical equations to calculate 16 of the transport collision integrals for the Lennard-Jones (12-6) potential.' *The Journal of Chemical Physics*, Vol. 57, No. 3, 1972, pp. 1100–1102.

[35] Svehla, R. A., 'Estimated viscosities and thermal conductivities of gases at high temperatures.' NASA Technical Paper R-132, Lewis Research Center, Washington, DC, 1962.

[36] Danberg, J., van Gulick, P., and Kim, J., 'Turbulence modeling for steady three-dimensional supersonic flow.' Contract Report BRL-CR-553, U.S. Army Ballistic Research Lab., 1986.

[37] Cebeci, T. and Smith, A., *Analysis of Turbulent Boundary Layers.*, Academic Press, New York, 1974.

[38] Hung, C. and MacCormack, R., 'Numerical solution of three-dimensional shock wave and turbulent boundary-layer interaction.' *AIAA Journal*, Vol. 16, No. 10, 1978, pp. 1090–1096.

[39] Degani, D. and Schiff, L., 'Computation of turbulent supersonic flows around pointed bodies having crossflow separation.' *Journal of Computational Physics*, Vol. 66, 1986, pp. 173–196.

[40] van Leer, B., 'Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method.' *Journal of Computational Physics*, Vol. 32, 1979, pp. 101–136.

[41] Anderson, W. K., Thomas, J. L., and van Leer, B., 'A comparison of finite volume flux vector splittings for the Euler equations.' Paper 85-0122, AIAA, 1985.

[42] Sweby, P. K., 'High resolution schemes using flux limiters for hyperbolic conservation laws.' *SIAM J. Numer. Anal.*, Vol. 21, 1984, pp. 995–1010.

[43] Jacobs, P. A., 'Single-block Navier-Stokes integrator.' Interim Report 18, ICASE, 1991.

[44] Liou, M. S. and Steffen, C. J., 'A new flux splitting scheme.' *Journal of Computational Physics*, Vol. 107, No. 1, 1993, pp. 23–39.

[45] Toro, E. F., 'A linearized Riemann solver for the time-dependent Euler equations of gas dynamics.' *Proc. Roy. Soc. London*, Vol. A 434, 1991, pp. 683–693.

[46] Osher, S. and Solomon, F., 'Upwind difference schemes for hyperbolic systems of conservation laws.' *Mathematics of Computation*, Vol. 38, No. 158, 1982, pp. 339–374.

[47] van Leer, B., 'Flux-vector splitting for the Euler equations,' *Lectures Notes in Physics, Volume 170.*, Springer-Verlag, 1982, pp. 507–512.

[48] Steger, J. L. and Warming, R. F., 'Flux vector splitting of the inviscid gas-dynamic equations with applications to finite-difference methods.' *Journal of Computational Physics*, Vol. 40, 1981, pp. 263–293.

[49] Roe, P. L., 'Approximate Riemann solvers, parameter vectors, and difference schemes.' *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[50] Swanson, R. C., Turkel, E., and White, J. A., 'An effective multigrid method for high-speed flows.' *Fifth Copper Mountain Conference on Multigrid Methods*, 1991.

[51] Chakravarthy, S. R., 'Development of upwind schemes for the Euler equations.' Contractor Report 4043, NASA, 1987.

[52] Craddock, C., 'B-spline surfaces for CFD grid generation.' Department of Mechanical Engineering Report 1/95, The University of Queensland, Brisbane, Australia, January 1995.

[53] Rogers, D. F. and Adams, J. A., *Mathematical Elements for Computer Graphics (2nd ed).*, McGraw-Hill, New York, 1990.

[54] Kennon, S. R. and Dulikravich, G. S., 'A posteriori optimization of computational grids.' Paper 85-0483, AIAA, January 1985.

[55] Anderson, D. A., Tannehill, J. C., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, New York, 1984.

[56] Frank, P. and Shubin, G., 'A comparison of optimization-based approaches for a model computational aerodynamic design problem.' *Journal of Computational Physics*, Vol. 98, 1992, pp. 74–89.

[57] Eleshaky, M. and Baysal, O., 'Airfoil shape optimization using sensitivity analysis on viscous flow equations.' *ASME Journal of Fluids Engineering*, Vol. 117, 1993, pp. 75–84.

[58] Kuruvila, G., Ta'asan, S., and Salas, M., 'Airfoil design and optimization by the one-shot method.' AIAA Paper 95-0478, AIAA, 1995.

[59] Foster, N. and Dulikravich, G., 'Three-dimensional aerodynamic shape optimization using genetic and gradient search algorithms.' *Journal of Spacecraft and Rockets*, Vol. 34, No. 1, 1997, pp. 36–42.

[60] Homaifar, A., Lai, H., and McCormick, E., 'System optimization of turbofan engines using genetic algorithms.' *Applied Mathematical Modelling*, Vol. 18, 1994, pp. 72–83.

[61] Korte, J. J. and Hodge, J. S., 'Flow quality of hypersonic wind-tunnel nozzles using computational fluid dynamics,' *AIAA Journal of Spacecraft and Rockets*, Vol. 32, No. 4, 1995, pp. 569–580.

[62] Keeling, S. L., 'A strategy for the optimal design of nozzle contours,' AIAA Paper 93-2720, Calspan Corporation/AEDC Operations Arnold Airforce Base, 1993.

[63] Tolle, R., 'A new optimum design code for hypersonic nozzles, utilizing response surface methodology,' AIAA Paper 97-0519, Wright Laboratory, 1997.

[64] Baysal, O., Eleshaky, M., and Burgreen, G., 'Aerodynamic shape optimization using sensitivity analysis on third-order Euler equations.' *Journal of Aircraft*, Vol. 30, No. 6, 1993, pp. 953–961.

[65] Scales, L., *Introduction to Non-Linear Optimization.*, Macmillan, London, 1985.

[66] Holland, J. H., 'Genetic algorithms.' *Scientific American*, Vol. 267, No. 1, 1992, pp. 44–50.

[67] Goldberg, D. E., *Genetic algorithms in search, optimization and machine learning.*, Addison-Wesley, Reading, Massachusetts, 1989.

[68] DeJong, K., 'Genetic algorithms: a 10 year perspective.' *Proceedings of the First International Conference on Genetic Algorithms.*, Lawrence Erlbaum, New Jersey, 1985.

[69] Spendley, W., Hext, G., and Himsworth, F., 'Sequential application of simplex designs in optimisation and evolutionary operation.' *Technometrics*, Vol. 4, 1962, pp. 441.

[70] Routh, M., Schwartz, P., and Denton, M., 'Performance of the super modified simplex.' *Analytical Chemistry*, Vol. 49, 1977, pp. 1422.

[71] Parker, L. J., Cave, M., and Barnes, R., 'Comparison of simplex algoritnms.' *Analytica Chimica Acta*, Vol. 175, 1985, pp. 231–237.

[72] Holland, J., *Adaptation in Natural and Artificial Systems.*, The University of Michigan Press., 1975.

[73] Michalewicz, Z. and Janikow, C. Z., 'Genetic algorithms for numerical optimization.' *Statistics and Computing*, Vol. 1, 1991, pp. 75–91.

[74] Salomon, R., 'Raising theoretical questions about the utility of genetic algorithms.' *Evolutionary Programming VI, Lecture Notes in Computer Science, 1213*, Springer-Verlag, 1997, pp. 275–284.

[75] Pruett, C. D. and Streett, C. L., 'A spectral collocation method for compressible, non-similar boundary layers.' *International Journal for Numerical Methods in Fluids*, Vol. 13, No. 6, 1991, pp. 713–737.

[76] Holden, M. S. and Moselle, J., 'Theoretical and experimental studies of the shock wave-boundary layer interaction on compression surfaces in hypersonic flow.' Report AF-2410-A-1, CALSPAN, Buffalo, New York, 1978.

[77] Hung, C. M. and MacCormack, R. W., 'Numerical solutions of supersonic and hypersonic laminar compression corner flows.' *AIAA Journal*, Vol. 14, No. 4, 1976, pp. 475–481.

[78] Lawrence, S. L., Tannehill, J. C., and Chaussee, D. S., 'Upwind algorithm for the parabolized Navier-Stokes equations.' *AIAA Journal*, Vol. 27, No. 9, 1989, pp. 1175–1183.

[79] Tracy, R. R., 'Hypersonic flow over a yawed circular cone.' Aeronautical Laboratories Memorandum 69, California Institute of Technology, 1963.

[80] Wendt, M. N. and Jacobs, P. A., 'Modular composite scramjet motor (C.S.M) testing.' Department of Mechanical Engineering Report 2/96, The University of Queensland, Brisbane, Australia, February 1996.

[81] Stalker, R. and Morgan, R., 'The University of Queensland free piston shock tunnel T4 - Initial operation and preliminary calibration.' *Fourth National Space Engineering Symposium, Adelaide, Australia*, IEAust, 1980.

[82] Billig, F. S., 'SCRAM-A supersonic combustion ramjet missile.' AIAA Paper 93-2329, AIAA, 1993.

[83] Archer, R.D. Saarlas, M., *Introduction to Aerospace Propulsion.*, Prentice-Hall, Upper Saddle River, N.J., 1996.

[84] Bittker, D. A. and Scullin, V. J., 'General chemical kinetics computer program for static and flow reactions, with application to combustion and shock-tube kinetics,' NASP Technical Memorandum TND-6586, NASA, Lewis Research Center, 1972.

[85] Coleman, G. T. and Stollery, J. L., 'Heat transfer in hypersonic turbulent separated flow.' Report 72-05, Department of Aeronautics, Imperial College of Science and Technology, London, England, 1972.

[86] National Oceanic and Atmospheric Administration and National Aeronautics and Space Administration and United States Air Force, *U.S. Standard Atmosphere.*, NASA, 1962.

[87] Rao, G., 'Exhaust nozzle contour for maximum thrust.' *Jet Propulsion*, Vol. 28, No. June, 1958, pp. 377–382.

[88] Reid, R., Prausnitz, J., and Poling, B., *The Properties of Gases and Liquids: 4th edition.*, McGraw Hill, New York, 1987.

A Axisymmetric PNS equations

We now consider an axisymmetric form of the PNS equations. A full treatment of deriving the axisymmetric Navier-Stokes equations from the three-dimensional equations is given in [43]. The governing equations for axisymmetric flow presented in this reference are presented below with the PNS assumptions applied for a multi-species gas (see section 2.1.1).

$$\frac{\partial}{\partial t}\int_{\Omega'} \mathbf{U} d\Omega' + \oint_S y(\mathbf{E_i}-\mathbf{E_v})\cdot\hat{\mathbf{n}}_\mathbf{x} dS + \oint_S y(\mathbf{F_i}-\mathbf{F_v})\cdot\hat{\mathbf{n}}_\mathbf{y} dS = \int_{\Omega'} \mathbf{Q} d\Omega' \quad , \tag{110}$$

where

$$\mathbf{U}=\begin{bmatrix} f_i\rho \\ \rho u_x \\ \rho u_y \\ \rho E \end{bmatrix} \quad , \quad \mathbf{E_i} = \begin{bmatrix} f_i\rho u_x \\ \rho u_x^2 + \epsilon p \\ \rho u_y u_x \\ \rho E u_x + p u_x \end{bmatrix} \quad , \quad \mathbf{F_i} = \begin{bmatrix} f_i\rho u_y \\ \rho u_y^2 + p \\ \rho u_z u_y \\ \rho E u_y + p u_y \end{bmatrix} \quad , \tag{111}$$

$$\mathbf{E_v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{F_v} = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u_x\tau_{xy} + u_y\tau_{yy} - q_y \end{bmatrix} \quad . \tag{112}$$

where $(i = 1, 2, ..., N_s)$. The viscous stresses are given by

$$\begin{aligned} \tau_{xx} &= \lambda\left(\frac{\partial u_y}{\partial y} + \frac{u_y}{y}\right) \quad , \\ \tau_{yy} &= 2\,\mu\frac{\partial u_y}{\partial y} + \lambda\left(\frac{\partial u_y}{\partial y} + \frac{u_y}{y}\right) \quad , \\ \tau_{xy} &= \mu\frac{\partial u_x}{\partial y} = \tau_{yx} \quad , \end{aligned} \tag{113}$$

(114)

where μ is the first coefficient of viscosity. As before, this formulation of viscous stresses assumes negligible bulk viscosity. Care must be taken when evaluating the viscous stresses on the axi where $y = 0$. On the axis, all fluxes evaluate to 0 because of the y multiplier in the integral equation (110). The viscous heat fluxes are

$$\begin{aligned} q_x &= 0 \quad \text{and} \\ q_y &= k\frac{\partial T}{\partial y} \quad . \end{aligned}$$

(115)

The effective source term (containing the front and back interface contributions) is

$$\mathbf{Q}=\begin{bmatrix} \dot{\omega_1} \\ \dot{\omega_2} \\ \vdots \\ \dot{\omega_{N_S}} \\ 0 \\ (p-\tau_{\theta\theta})A \\ q \end{bmatrix} \tag{116}$$

where

$$\tau_{\theta\theta} = 2\,\mu\frac{u_y}{y} + \lambda\left(\frac{\partial u_y}{\partial y} + \frac{u_y}{y}\right) \tag{117}$$

The axisymmetric cell volume, Ω', used in the above equations, is the axisymmetric cell volume defined as

$$\Omega' = \frac{A\ .\ y_{\text{cell}}}{2\psi}\quad , \tag{118}$$

where A is the area of the cell which is equivalent to the $ZFace$ area in the three-dimensional formulation, y_{cell} is the distance from the cell centre to the axis, and 2ψ is the circumferential extent of the axisymmetric cell in radians (see Fig. 43).

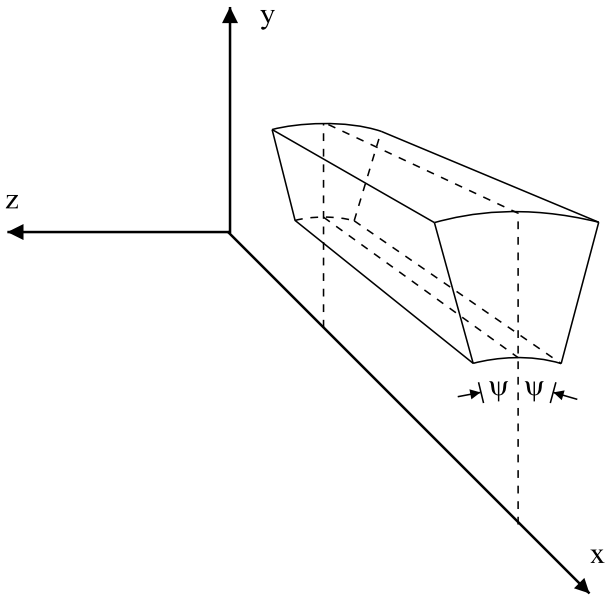

Figure 43: Axisymmetric finite volume cell.

B Lennard-Jones Potentials

Table 6: Lennard-Jones 12-6 Potentials for various gases [88]

	Substance	$\sigma, \mathring{A}$	ϵ/k_B
Air	Air	3.711	78.6
CO	Carbon monoxide	3.690	91.7
CO_2	Carbon dioxide	3.941	195.2
H	Hydrogen	2.070	37.0
H_2	Hydrogen	2.827	59.7
OH	Hydroxide	3.147	79.8
H_2O	Water (gas)	2.800	260.0
H_2O_2	Hydrogen peroxide	4.196	289.3
HO_2	Hydroperoxo	3.068	168.0
NO	Nitric oxide	3.470	119.0
N	Nitrogen	3.798	71.4
N_2	Nitrogen	3.798	71.4
NO_2	Nitrogen oxide	3.798	71.4
HNO	Nitrosyl hydride	3.798	71.4
N_2O	Nitrous oxide	3.828	232.4
O	Oxygen	3.050	106.7
O_2	Oxygen	3.467	106.7

C Finite-Rate Chemistry Models

Table 7: Simple nitrogen and oxygen reaction model (from Oldenborg em et al. [23])

Reaction number	Reaction	Reaction rate variables A_j	N_j	Activation temperature K
1	$O + O + M \longleftrightarrow O_2 + M$	1.14×10^{17}	-1.0	0
2	$O + NO \longleftrightarrow O_2 + N$	3.80×10^{9}	1.0	20820
3	$N_2 + O \longleftrightarrow N + NO$	1.82×10^{14}	0.0	38370
Third body efficiencies for all the termolecular reactions are 1.0				

Table 8: Simple carbon dioxide and oxygen reaction model [28]

Reaction number	Reaction	Reaction rate variables A_j	N_j	Activation temperature K
1	$O_2 + M \longleftrightarrow O + O$	2.0×10^{21}	-1.50	59750
2	$CO_2 + M \longleftrightarrow CO + O + M$	6.9×10^{21}	-1.50	63275
3	$CO_2 + O \longleftrightarrow CO + O_2$	2.1×10^{13}	0.0	27800
Third body efficiencies for all the termolecular reactions are 5.0 for M = O in reaction 1, 2.03 for M = O in reaction 2, and 1.0 for all other M.				

Table 9: Hydrogen and oxygen combustion reaction model by Jachimowski [27]

Reaction number	Reaction	Reaction rate variables A_j	N_j	Activation temperature K
1	$H_2 + O_2 \longleftrightarrow OH + OH$	0.170×10^{14}	0.0	24219
2	$H + O_2 \longleftrightarrow OH + O$	0.142×10^{15}	0.0	8249
3	$OH + H_2 \longleftrightarrow H_2O + H$	0.316×10^{8}	1.8	1524
4	$O + H_2 \longleftrightarrow OH + H$	0.207×10^{15}	0.0	6916
5	$OH + OH \longleftrightarrow H_2O + O$	0.550×10^{14}	0.0	3521
6	$H + OH \longleftrightarrow H_2O + M$	0.221×10^{23}	-2.0	0
7	$H + H \longleftrightarrow H_2 + M$	0.653×10^{18}	-1.0	0
8	$H + O_2 \longleftrightarrow HO_2 + M$	0.320×10^{19}	-1.0	0
9	$HO_2 + OH \longleftrightarrow H_2O + O_2$	0.500×10^{14}	0.0	503
10	$HO_2 + H \longleftrightarrow H_2 + O_2$	0.500×10^{14}	0.0	352
11	$HO_2 + H \longleftrightarrow OH + OH$	0.199×10^{15}	0.0	905
12	$HO_2 + O \longleftrightarrow OH + O_2$	0.500×10^{14}	0.0	503
13	$HO_2 + HO_2 \longleftrightarrow H_2O_2 + O_2$	0.199×10^{13}	0.0	0
14	$HO_2 + H_2 \longleftrightarrow H_2O_2 + H$	0.301×10^{12}	0.0	9406
15	$H_2O_2 + OH \longleftrightarrow HO_2 + H_2O$	0.102×10^{14}	0.0	956
16	$H_2O_2 + H \longleftrightarrow OH + H_2O$	0.500×10^{15}	0.0	5030
17	$H_2O_2 + O \longleftrightarrow OH + HO_2$	0.199×10^{14}	0.0	2968
18	$M + H_2O_2 \longleftrightarrow OH + OH$	0.121×10^{18}	0.0	22886
Third body efficiencies for all the termolecular reactions are 1.0				

Table 10: NASP Chemistry model for hydrogen combustion in air [23]

Reaction number	Reaction	Reaction rate variables A_j	N_j	Activation temperature K
1	$OH + H_2 \longleftrightarrow H + H_2O$	2.16×10^{8}	1.51	1726
2	$H + O_2 \longleftrightarrow O + OH$	1.91×10^{14}	0.0	8273
3	$O + H_2 \longleftrightarrow H + OH$	5.06×10^{4}	2.67	3166
4	$H + HO_2 \longleftrightarrow H_2 + O_2$	2.5×10^{13}	0.0	349
5	$H + HO_2 \longleftrightarrow OH + OH$	1.5×10^{14}	0.0	505
6	$O + HO_2 \longleftrightarrow OH + O_2$	2.0×10^{13}	0.0	0
7	$OH + HO_2 \longleftrightarrow H_2O + O_2$	2.0×10^{13}	0.0	0
8	$H + O_2 + M \longleftrightarrow HO_2 + M$	8.0×10^{17}	-0.8	0
9	$H + OH + M \longleftrightarrow H_2O + M$	8.62×10^{21}	-2.0	0
10	$H + H + M \longleftrightarrow H_2 + M$	7.3×10^{17}	-1.0	0
11	$H + O + M \longleftrightarrow OH + M$	2.6×10^{16}	-0.6	0
12	$O + O + M \longleftrightarrow O_2 + M$	1.14×10^{17}	-1.0	0
13	$OH + OH \longleftrightarrow O + H_2O$	1.5×10^{9}	1.14	0
14	$OH + OH + M \longleftrightarrow H_2O_2 + M$	4.73×10^{11}	1.0	-3206
15	$OH + H_2O_2 \longleftrightarrow H_2O + HO_2$	7.0×10^{12}	0.0	722
16	$O + H_2O_2 \longleftrightarrow OH + HO_2$	2.8×10^{13}	0.0	3220
17	$H + H_2O_2 \longleftrightarrow H_2 + HO_2$	1.7×10^{12}	0.0	1900
18	$H + H_2O_2 \longleftrightarrow H_2O + OH$	1.0×10^{13}	0.0	1800
19	$HO_2 + HO_2 \longleftrightarrow H_2O_2 + O_2$	2.0×10^{12}	0.0	0
20	$CO + OH \longleftrightarrow CO_2 + H$	4.4×10^{6}	1.5	-373
21	$O + NO \longleftrightarrow N + O_2$	3.80×10^{9}	1.0	20820
22	$O + N_2 \longleftrightarrow N + NO$	1.82×10^{14}	0.0	38370
23	$H + NO \longleftrightarrow N + OH$	1.70×10^{14}	0.0	24560
24	$H + NO + M \longleftrightarrow HNO + M$	2.17×10^{15}	0.0	-300
25	$O + NO + M \longleftrightarrow NO_2 + M$	3.31×10^{10}	1.0	-4522
26	$H + HNO \longleftrightarrow H_2 + NO$	1.26×10^{13}	0.0	2000
27	$O + HNO \longleftrightarrow OH + NO$	5.0×10^{11}	0.5	1000
28	$OH + HNO \longleftrightarrow H_2O + NO$	1.26×10^{12}	0.5	1000
29	$H + NO_2 \longleftrightarrow OH + NO$	3.5×10^{14}	0.0	740
30	$O + NO_2 \longleftrightarrow O_2 + NO$	1.0×10^{13}	0.0	300
31	$HO_2 + NO \longleftrightarrow OH + NO_2$	2.09×10^{12}	0.0	-240

Third body efficiencies for all the termolecular reactions are 2.5 for M = H_2, 16.25 for M = H_2O, 3.8 for M = CO_2, and 1.0 for all other M.

D Indexing and Data Storage

Although it has been convenient to formulate the numerical scheme using fractional subscripts for the cell faces, the programming language can only use integral values to address the data arrays. Within the code, cell-centred quantities are simply addressed as $Cell[ix][iy][iz]$ but the cell interfaces and vertices are addressed as shown in Tables 11 and 12.

Table 11: Index translation table for cell faces.

Label	Coded index
$ix + \frac{1}{2}$	$XFace[ix][iy][iz]$
$ix - \frac{1}{2}$	$XFace[ix-1][iy][iz]$
$iy + \frac{1}{2}$	$YFace[iy][iz]$
$iy - \frac{1}{2}$	$YFace[iy-1][iz]$
$iz + \frac{1}{2}$	$ZFace[iy][iz]$
$iz - \frac{1}{2}$	$ZFace[iy][iz-1]$

Table 12: Index translation table for cell vertices.

Label	Coded index
A	$[ix][iy-1][iz-1]$
B	$[ix][iy][iz-1]$
C	$[ix][iy][iz]$
D	$[ix][iy-1][iz]$
A'	$[ix-1][iy-1][iz-1]$
B'	$[ix-1][iy][iz-1]$
C'	$[ix-1][iy][iz]$
D'	$[ix-1][iy-1][iz]$

Relative to a time-marching solver for three-dimensional flow, the amount of data that needs to be stored is relatively small. If $nnx \times nny \times nnz$ cells are required to resolve a particular flow field, the space-marching solver stores cell-centred data for $4 \times nny \times nnz$ cells only. Also, three slices of vertex data, three slices of $XFace$ data and only one slice each of $YFace$ and $ZFace$ data are stored. This is enough data to define the geometry and flow state of the currently *active* slice of the flow field. The two extra upwind slices of cell-centred data are used for high-order reconstruction in the streamwise direction. The extra downwind slice geometry is required to calculate viscous fluxes. Even though viscous fluxes are not required for inviscid calculations, the extra downwind slice data structure is maintained in inviscid calculations since it only requires a little extra memory and does not affect the computation speed. For

viscous solutions, extra data needs to be stored for the secondary cells to calculate the viscous fluxes. Two slices of cell-centred data, three slices of $XFace$ data, three vertex slices and two slices each of $YFace$ and $ZFace$ data are stored for viscous calculations.

Within each slice, enough memory is allocated to cover the *active* flow region (i.e. that region within the four duct boundaries) plus a layer of *ghost* cells around the outside of the boundaries. This ghost-cell layer is two cells deep. It is used to apply the boundary conditions and allow the high-order reconstruction scheme to be used right up to the interfaces on the duct boundaries (see section 2.7.2). *Active* cell-centred data is accessed with the index range

$$iymin \leq iy \leq iymax \ , \quad izmin \leq iz \leq izmax \ ,$$

while cell-vertex data, including the boundary interfaces, is accessed with the range

$$iymin - 1 \leq iy \leq iymax \ , \quad izmin - 1 \leq iz \leq izmax \ .$$

The total amount of inviscid memory allocated (at run time) is $5061 \times (nny + 4) \times (nnz + 4)$ bytes. This includes both the *active*- and *ghost*- cell data. If viscous calculations are required, extra memory must be allocated for the secondary cells. The extra memory required is $5592 \times (nny + 1) \times (nnz + 1)$ bytes.

Although the solver is intended for computing three-dimensional flows, two-dimensional and axisymmetric flows may be studied by collapsing the cross-stream z direction to a minimum of two *active* cells. The walls normal to the z-coordinate direction are then made parallel. The required minimum of two active cells imposes a penalty in both memory and CPU time required to compute two-dimensional flows, but it does simplify the coding within *sm_3d*$^+$.

E Header File

This section lists the first part of the main header file *sm_3d.h*. It contains all of the MACROS, gas types and case identifications numbers which can be varied by the user. The settings shown are for completeness and should not be taken as default values.

```
/* sm_3d.h -- header file for global declarations */


/* Find the other header files. */

#ifndef COMPILER_H
#  include "compiler.h"
#endif

#ifndef GEOM_H
#  include "geom.h"
#endif

#ifndef BEZIER_H
#  include "bezier.h"
#endif

/*---------------------------------------------------------------------*/

/*
 * --------------------
 * CONFIGURATION MACROs
 * --------------------
 */


#define  DEBUG  1
/*
 * Debugging level...
 * 0 = no debugging
 *     Use this setting if the space-marching code is used
 *     inside an optimization loop and is likely to be called
 *     many times.
 * 1 = print message on entry to infrequently used functions
 *     This will be the most common setting.
 * 2 = print message on entry to frequently used functions
 * 3 = dump data every step
 *     This is really tedious!
 *
 */

#define  EXTRAP_TYPE  0
/*
 * Type of higher-order interpolation/reconstruction.
 * INTERP_TYPE = 0 : third-order upwind-biased
 *               1 : fully upwind
 */
```

```
#define  INTERP_TYPE  0
/*
 * Type of higher-order interpolation/reconstruction.
 * INTERP_TYPE = 0 : third-order upwind-biased
 *               1 : fully upwind
 */

#define CFL_STRINGENT 0
/*
 * Use the maximum velocity and the minimum
 * characteristic length of the cell to work out
 * the minimum time step.
 */

#define FLUX_SPLIT   0
/*
 * Type of flux-splitting.
 * FLUX_SPLIT = 0 : 3-stage approximate Riemann solver.
 *              1 : AUSM flux-splitting
 */

#define DIMENSION 3
/*
 * Specifies the dimension of the problem
 * 0 - Axi-symmetric
 * 1 - One-dimensional (not used)
 * 2 - Two-dimensional
 * 3 - Three-dimensional
 */

#define VISC 1
/*
 * Turn the viscous routines on or off
 */

#define TURB 1
/*
 * Switch to include the Baldwin Lomax algebraic turbulence model
 */

#define CMUTM 0
/*
 * Transition to turbulence condition;
 * mu_turb = 0.0 if mu_turb_max < Cmutm * mu_inf; usually 14
 */

#define CHEM 0
/*
 * Turn the chemistry routines on or off
 */

#define FROZEN 0
/*
 * Switch that sets all production rates to 0.0 if chemistry active
 */

#define ADD_HEAT 0
/*
```

```
 * Switch to activate the energy source term if applicable to the
 * case. See sm_src.c
 * ADD_HEAT = 0:  do not add heat in the source-term routine
 *            1:  include the heat production terms as coded for
 *                each particular case.
 */

#define N_SP 11
/*
 * Declaration of maximum number of species in a reaction set
 * That is we can not have more than N_SP species in a reaction set
 */

#define N_RE 20
/*
 * Declaration of maximum number of reactions that the code can
 * deal with. This is actually a memory limitation so make as
 * small as possible
 */

#define BETA_X 3.5
#define BETA_Y 1.1
#define BETA_Z 1.1
/*
 * Clustering factors for each direction. The range of
 * BETA is, 1 < BETA < +inf where the closer BETA is to
 * 1, the greater the clustering.
 */

#define X_START  1
#define X_END    0
#define Y_BOTTOM 0
#define Y_TOP    1
#define Z_LEFT   0
#define Z_RIGHT  1
/*
 * These macros indicate where to cluster the cells.
 * For each set in each direction, a 1 can be placed in
 * either, both or none of the extremes. Placing a 1 in
 * both extremes clusters at both ends and placing a 0
 * in both produces no clustering. Note that if 2D is
 * activated, the macros for the Z direction should both
 * contain 0's.
 */

#define NDVMAX  20
/*
 * Declared vector size for the design variables.
 */

#define VOLUME_TOL  1.0e-7
/*
 * The magnitude of the tetrahedra (negative) volume
 * that will be tolerated.  It seems that, for some
 * difficult (read distorted) geometries, the cells have
 * some negative tetrahedra volumes probably
 * because the grid line cross.
 */
```

```
/*-------------------------------------------------------------*/

/*
 * -------------
 * GAS SELECTION
 * -------------
 */

#define  Rmol 8.3143     /* Universal Gas Constant, J/mol.K */

#define  PERF_AIR_14  0
#define  LOWT_AIR_14  1
#define  PERF_AIR_13  2
#define  PERF_HE_167  3
#define  PERF_AR_167  4
#define  PERF_N2      5
#define  PERF_CO2     6
#define  EQ_AIR_1SP   7
#define  EQ_N2        8
#define  VIBEQ_N2     9
#define  WEIRD_167   10
#define  NON_EQ      11

/*
 * Above are the gas types available.
 * PERF_AIR_14 : Perfect gas, air,            GAMMA = 1.4
 * LOWT_AIR_14 : Perfect gas, air very low T  GAMMA = 1.4
 * PERF_AIR_13 : Perfect gas, air,            GAMMA = 1.32
 * PERF_HE_167 : Perfect gas, Helium,         GAMMA = 1.667
 * PERF_AR_167 : Perfect gas, Argon,          GAMMA = 1.667
 * PERF_N2     : Perfect gas, Nitrogen,       GAMMA = 1.4
 * PERF_CO2    : perfect gas, Carbon dioxide, GAMMA = 1.28
 * EQ_AIR_1SP  : Equilibrium Air, 1-specie, Tannehill EOS
 * EQ_N2       : Nitrogen, equilibrium chemistry
 * VIBEQ_N2    : Nitrogen in vib. eq.
 * WEIRD_167   : Nitrogen, Weird perfect gas for DSMC comparison
 */


/*
 * Pick one ...
 */

#define  TYPE_of_GAS PERF_AIR_14
/* #define  TYPE_of_GAS LOWT_AIR_14  */
/* #define  TYPE_of_GAS PERF_AIR_13  */
/* #define  TYPE_of_GAS PERF_HE_167  */
/* #define  TYPE_of_GAS PERF_AR_167  */
/* #define  TYPE_of_GAS PERF_N2      */
/* #define  TYPE_of_GAS PERF_CO2     */
/* #define  TYPE_of_GAS EQ_AIR_1SP   */
/* #define  TYPE_of_GAS EQ_N2        */
/* #define  TYPE_of_GAS VIBEQ_N2     */
/* #define  TYPE_of_GAS WEIRD_167    */
```

```
/*
 * -----------------
 * Equation of State.
 * -----------------
 */
#define  EOS_PERF     0
#define  EOS_EQUIL    1
#define  EOS_EQ_N2    2
#define  EOS_VIBEQ_N2 3
#define  EOS_NONEQ    4
/*
 * EOS_PERF  : perfect gas, constant gamma, constant Cv
 * EOS_EQUIL : equilibrium air -- Srinivasan & Tannehill
 *             curve fits.  Only one species.
 * EOS_EQ_N2 : vibrational equilibrium nitrogen
 * EOS_VIBEQ_N2 : equilibrium nitrogen -- R. K. Prabhu &
 *             W. D. Erickson (1988)
 *             A rapid method for the computation of
 *             equilibrium chemical composition of air
 *             to 15000 K. NASA Technical Paper 2792
 * EOS_NONEQ : non-equilibrium gas - multiple species
 *
 * GAMMA = ratio of specific heats for a perfect gas
 * C_p   = specific heat at constant pressure (J/kg/K)
 * C_v   = specific heat at constant volume (J/kg/K)
 * GAS_R = gas constant for perfect gas (J/kg/K)
 *
 */
#if (CHEM != 1)

#if (TYPE_of_GAS == PERF_AIR_14)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.4
#   define  PRANDTL    0.72
#   define  GAS_R      287.0
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif


#if (TYPE_of_GAS == LOWT_AIR_14)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.4
#   define  PRANDTL    0.70
#   define  GAS_R      287.0
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif


#if (TYPE_of_GAS == PERF_AIR_13)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.3
#   define  PRANDTL    0.72
#   define  GAS_R      287.0
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif
```

```
#if (TYPE_OF_GAS == PERF_HE_167)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.667
#   define  PRANDTL    0.67
#   define  GAS_R      2077.0
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif


#if (TYPE_OF_GAS == PERF_AR_167)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.667
#   define  PRANDTL    0.67
#   define  GAS_R      208.1
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif


#if (TYPE_of_GAS == PERF_N2)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.4
#   define  PRANDTL    0.70
#   define  GAS_R      296.9
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif


#if (TYPE_of_GAS == PERF_CO2)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.28
#   define  PRANDTL    0.73
#   define  GAS_R     189.0
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif


#if (TYPE_of_GAS == EQ_AIR_1SP)
#   define  EOS_TYPE   EOS_EQUIL
#   define  GAMMA      1.4
#   define  PRANDTL    0.72
#   define  GAS_R      287.0
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif

#if (TYPE_of_GAS == EQ_N2)
#   define  EOS_TYPE   EOS_EQ_N2
#   define  GAMMA      1.4
#   define  PRANDTL    0.72
#   define  GAS_R      296.9
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
```

```
#endif

#if (TYPE_of_GAS == VIBEQ_N2)
#   define  EOS_TYPE   EOS_VIBEQ_N2
#   define  GAMMA      1.4
#   define  PRANDTL    0.72
#   define  GAS_R      296.9
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif


#if (TYPE_OF_GAS == WEIRD_167)
#   define  EOS_TYPE   EOS_PERF
#   define  GAMMA      1.667
#   define  PRANDTL    0.72
#   define  GAS_R      296.2
#   define  C_v     (GAS_R / (GAMMA - 1.0))
#   define  C_p     (C_v + GAS_R)
#endif

#else /* Select non-equilibrium gas */

/*
 * GAMMA, PRANDTL, R, C_v, C_p are all dependent on the
 * mixture composition. Will put small non-zero numbers
 * in to keep compiler happy with divisions
 */
#   define  TYPE_of_GAS NON_EQ
#   define  EOS_TYPE   EOS_NONEQ
#   define  GAMMA      0.0
#   define  PRANDTL    0.0
#   define  GAS_R      0.0
#   define  C_v        1.0e-6
#   define  C_p        1.0e-6
#   define  PMIN       1.0e-12
#   define  TMIN       1.0e-4

#endif

/*-------------------------------------------------------------*/

/* -------------------
 * CASE IDENTIFICATION
 * -------------------
 *
 * General Cases...
 * -------------
 *
 * GENERIC_CASE: Just compute the flow through a duct as specified in
 *               the input parameter file.  The case of Mach 2 flow
 *               onto a 14 degree ramp is an example.
 *
 * THRUST_RAMP : The single-ramp, open-duct scramjet model examined
 *               by Gary Allen in AIAA Paper 91-0227
 *
 * RAYS_MODEL  : Ray Stalker's small scramjet model as tested
 *               by David Mee and Allan Paull
```

```
*
* CRG_CSM     : CRG Modified-composite-engine with/without regions
*               of heat addition.
*
* CRG_ICE     : Initial composite engine model as used in the
*               CRG scramjet study 1993
*
* CRG_WBM     : The steel-plate scramjet module tested by WBM
*               for the CRG scramjet study
*
* M8_NOZZLE   : Mach 4 to 8 box-section nozzle
*
* SMART_RAMP  : The Mach-jet/expansion-ramp combination studied
*               by Michael Smart.
*
* ROUND_JET   : A round Mach jet and a 15 degree ramp (as per
*               SMART_RAMP and THRUST_RAMP).
*
* FREE_CONE   : A cone in free flight.  This case requires a special
*               piece of code for the grid generation.
*
* WINGED_CONE : Same cone plus a couple of stubby wings.
*
* Q_RAMP_1    : Thrust ramp with heat addition in a horizontal line
*               Like Gary Allen's scramjet study.
*
* Q_RAMP_2    : Thrust ramp with heat addition in a vertical line
*               Like Michael Smart's Mach jet study
*
* Q_RAMP_3    : Thrust ramp with heat addition in a round jet
*               Like discrete orifice fuel injection.
*
* SCRAM_2D    : A nominally two-dimensional scramjet for E4351
*               assignment, 1994
*
* SCRAM15     : A nominally two-dimensional scramjet for Mach
*               15 flight. Used as a baseline for optimization.
*
* SC_COMB     : Two-dimensional scramjet with heat added uniformly
*               across the entire combustor.
*
* ISABE_A     : A nominally two-dimensional combustor and thrust
*               nozzle with heat addition uniform across the entire
*               combustor cross-section.
*
* ISABE_B     : A nominally two-dimensional combustor and thrust
*               nozzle with heat addition across a horizontal zone.
*
* ISABE_C     : One quarter of a nominally two-dimensional combustor
*               and thrust nozzle with heat addition in a circular
*               cross-section zone.
*
* ISABE_D     : A nominally two-dimensional combustor and thrust
*               nozzle with heat addition uniform across the entire
*               combustor cross-section.
*
* ISABE_E     : A nominally two-dimensional combustor and thrust
*               nozzle with heat addition across a horizontal zone.
```

```
*
* ISABE_F     : One quarter of a nominally two-dimensional combustor
*               and thrust nozzle with heat addition in a circular
*               cross-section zone.
*
* V_PLATE     : Two dimensional viscous plate flow for testing the
*               Parabolized Navier-Stokes coding
*
* COWL_A      : Two dimensional scramjet combustor/exhaust with an
*               inclined cowl surface.
*
* HUMP        : Two dimensional nozzle for investigating pressure
*               hump on thrust surface.
*
* NOZZLE      : Olga's axisymmetric nozzle design with finite rate
*               chemistry
*
*
*-------------------
* Optimization cases
*-------------------
*
* OPT_RAMP_1   : As for Q_RAMP_1 but with variable ramp angle.
* OPT_RAMP_2   :        Q_RAMP_2
* OPT_RAMP_3   :        Q_RAMP_3
*
* BEZ_RAMP_1   :        Q_RAMP_1 but with a Bezier curve thrust ramp
* BEZ_RAMP_2   :        Q_RAMP_2
* BEZ_RAMP_3   :        Q_RAMP_3
*
* BEZ_NOZ2D    : E4351 Scramjet nozzle with an optimal Bezier ramp
*
* OPT_ISABE_A : A nominally two-dimensional combustor and thrust
*               nozzle with heat addition uniform across the entire
*               combustor cross-section.
*
* OPT_ISABE_B : A nominally two-dimensional combustor and thrust
*               nozzle with heat addition across a horizontal zone.
*
* OPT_ISABE_C : One quarter of a nominally two-dimensional combustor
*               and thrust nozzle with heat addition in a circular
*               cross-section zone.
*
* OPT_ISABE_D : Same as A but for flight style 2D scramjet
*               Higher contraction ratio
*
* OPT_ISABE_E : Same as B but for flight style 2D scramjet
*               Higher contraction ratio
*
* OPT_ISABE_F : Same as C but for flight style 2D scramjet
*               Higher contraction ratio
*
* OPT_COWL_A  : A nominally two-dimensional combustor and thrust
*               nozzle with heat addition uniform across the entire
*               combustor cross-section. The angle of the cowl is
*               the design variable. A drag penalty is imposed on
*               the cowl based on Newtonian pressures.
*
```

```
 * OPT_SQNOZZ  : A square section nozzle that has an adverse
 *               pressure gradient on the walls causing counter-
 *               rotating vortices. Using optimization to minimise
 *               the vortices.
 *
 *-------------------
 * Axisymmetric cases
 *-------------------
 *
 * DRUM_CONIC  : The throat and conical expansion of the M7
 *               Drummond tunnel nozzle (Actual called the
 *               "Small Shock Tunnel Facility at UQ"
 *
 *---------------------------
 * Finite rate chemistry cases
 *---------------------------
 *
 * O2_DISS     : O2 Dissociation
 *
 * CO2_DISS    : CO2 Dissociation
 *
 * H2O2_COMB   : H2 combustion in air
 *
 * SCRAM_CHEM  : Study of SCRAM15 combustor with chemistry.
 *
 *--------------------------------------------
 * Finite rate chemistry cases with optimization
 *--------------------------------------------
 *
 * ISABE_CHEM  : ISABE 2D scramjet with optimization for chemistry
 *
 */

#define   GENERIC_CASE  0
#define   THRUST_RAMP   1

#define   RAYS_MODEL    2
#define   CRG_ICE       3
#define   CRG_WBM       4

#define   M8_NOZZLE     5
#define   SMART_RAMP    6
#define   ROUND_JET     7
#define   FREE_CONE     8
#define   WINGED_CONE   9
#define   Q_RAMP_1      10
#define   Q_RAMP_2      11
#define   Q_RAMP_3      12

#define   SCRAM_2D      14
#define   CRG_CSM       15
#define   SCRAM15       16
#define   SC_INLET      17
#define   SC_COMB       19

#define   RAY_NOZZLE_1  21
#define   RAY_NOZZLE_2  22
#define   ISABE_A       23
```

```
#define   ISABE_B       24
#define   ISABE_C       25
#define   V_PLATE       26
#define   ISABE_D       27
#define   ISABE_E       28
#define   ISABE_F       29
#define   COWL_A        30

#define   HUMP          35

#define   M7_SQUARE     50


/*
 * Axisymmetric Cases start at 71
 */

#define   NOZZLE        71

#define   VISCOUS_CONE  72

#define   DRUM_CONIC    73

/*
 * Optimization cases start at 101 end 200
 */

#define   OPT_RAMP_1    101
#define   OPT_RAMP_2    102
#define   OPT_RAMP_3    103
#define   BEZ_RAMP_1    104
#define   BEZ_RAMP_2    105
#define   BEZ_RAMP_3    106
#define   BEZ_NOZ2D     107

#define   OPT_M8_NOZZ   110

#define   OPT_ISABE_A   123
#define   OPT_ISABE_B   124
#define   OPT_ISABE_C   125
#define   OPT_ISABE_D   126
#define   OPT_ISABE_E   127
#define   OPT_ISABE_F   128
#define   OPT_COWL_A    129
#define   DIV_COMB      130

#define   OPT_SQNOZZ    135

/*
 * Finite rate chemistry cases start at 201
 */

#define   O2_DISS       201
#define   CO2_DISS      202
#define   H2O2_COMB     203
#define   SCRAM_CHEM    216

/*
```

```
 * Finite rate chemistry cases with optimization start at 301 end 400
 */

#define   ISABE_CHEM    301

/*--------------------------------------------------------------------*/
```

The header file continues on with structure definitions and prototype declarations which have been omitted for brevity.