NAS1-00135B

Technical Note 06-474

Date: 5/11/06

To:

Dr. Aaron Auslender
Assistant Branch Head
Hypersonic Airbreathing
Propulsion Branch
Mail Stop 168
NASA Langley Research Ctr.
Hampton VA 23681-0001
Phone (757) 864 6545
Fax (757) 864 6243

cc:
- Paul Ferlemann, TN Files, Swales
- Tom Jentink, Swales

HYPERSONIC VEHICLE ANALYSIS GROUP

Technical Note 06-474

User Manual and Methodology for One Dimensionalization code: Massflow3d version 8.4

Benjamin Meyer

Cover Sheet + 30 Pages

Contract NAS1-00135B
Technical Monitor: Dr.Aaron Auslender
Technical Leader: R. D. Bittner

NOMENCLATURE

A	area
a	speed of sound
$a_1, a_2, \ldots$	McBride-Gordon curve fit polynomials
C_{FG}	gross thrust coefficient
C_p	constant pressure specific heat
C_S	stream thrust coefficient
C_v	constant volume specific heat
d	distance from surface to first node off surface
E	total energy
e	internal energy
F_{ideal}	ideal gross thrust (expansion to P_∞)
F_{ij}	inviscid force on surface in j direction
F_{vac}	ideal vacuum stream thrust (ideal expansion to A_{exit})
F_{vj}	viscous force on surface in j direction
F_1, F_2	Newton-Raphson solution functions
h	enthalpy
$\hat{i}$	i component of unit vector (will have subscript)
J	Newton-Raphson solution Jacobian
$\hat{j}$	j component of unit vector (will have subscript)
$\hat{k}$	k component of unit vector (will have subscript)
k	fluid thermal conductivity
LHS	left hand side of equation
M	Mach number
$\dot{m}$	mass capture
mom	momentum
MW	molecular weight
$\hat{n}$	unit vector normal to area
P	static pressure
P_{pitot}	pitot pressure
Q	heat transfer
R	specific gas constant
$\bar{R}$	universal gas constant
RHS	right hand side of equation
T	static temperature
TP	thrust potential
s	entropy
u	x component of velocity
v	y component of velocity
$\vec{V}$	velocity vector
w	z component of velocity
X	mass fraction
Y	mole fraction
ϕ	fuel equivalence ratio
γ	ratio of constant specific heats
η_{AKE}	adiabatic kinetic energy efficiency
η_c	combustion efficiency
η_{KE}	kinetic energy efficiency
η_m	mixing efficiency
η_r	reaction efficiency
λ	any primitive variable: ρ, u, v, w P, or T
μ	laminar viscosity
ρ	static density
υ	specific volume

SUBSCRIPTS

exit	value at exit of internal nozzle
i	absolute plane indice
n	normal to surface
o	1D quantity (except where noted)
ref	reference (used in distortion and TP)
s	any species or surface vector component (context dependent)
user	value based on user input
T	total (stagnation)
∞	free stream

INTRODUCTION

One dimensional (1D) methods are often employed in the early stages of engine design, or where large databases are desired, since they can be run quickly and cheaply. In most cases 3D CFD methods are eventually employed for some fraction of these cases to verify the simpler 1D methods. In other cases 3D solutions are used to provide inflow planes to 1D codes. This document explains the methodology and usage of the one dimensionalizing code “massflow3d” version 8.4 which allows 3D results to be analyzed in a 1D sense.

Several methods exist for one dimensionalizing a 3D flowfield, such as area weighted averaging, mass flow weighted averaging, and flux conserved models. All three of these approaches are used in this program. The approaches taken and assumptions for each method are explained. Additional outputs include pitot conditions, total conditions, reaction information (η_r, η_m, η_c, ϕ, etc.), wall information (wetted area, forces, moments, heat transfer, etc.), thrust potential, nozzle efficiency, η_{KE} and much more discussed in the subsequent

sections. The code combines and expands on the capabilities of several existing fragmented codes and provides a single tool for usage. This document provides a clear explanation of the methods used in the code and can be provided as documentation regarding inquires into such. A user manual is also provided detailing the codes numerous options.

AREA WEIGHTED AVERAGING

Of the area weighted, mass weighted, and flux conserved approaches, the area weighted approach is probably the least accurate in representing the 3D flowfield one dimensionally. This is because the primitive variables (rho, u, v, w, P, and T) are represented by a simple area average over the plane as shown by Eq. 1.

$$\lambda_o = \frac{\int_A \lambda \, dA}{\int_A dA} \qquad (1)$$

where λ_o is solved for all the primitive variables. All subsequent 1D calculations such as enthalpy, mass flow rate, energy, velocity, etc. are then done with these primitive variables. The one dimensional mass fraction is the only quantity calculated using mass weighted average, which is discussed in the next section, since using any other approach would be inappropriate. Since the size of the grid cells are dictated by the user, the area weighted approach can have regions of flow with large areas but low mass flow rates. These areas are then over represented in the one dimensionalization. This can lead to a large difference between flux values such as mass flow rate, energy, etc. between the 1D and 3D flowfield. The following approach, mass weighted averaging, attempts to address this shortcoming.

MASS WEIGHTED AVERAGING

The mass weighted average approach to one dimensionalizing the flow determines the primitive variables by using a mass weighted average over the plane as shown by Eq. 2.

$$\lambda_o = \frac{\int_A \lambda(\rho \vec{V} \bullet \hat{n}) dA}{\int_A (\rho \vec{V} \bullet \hat{n}) dA} \qquad (2)$$

where λ_o is solved for all the primitive variables. Note the denominator represents the mass flow through the plane. As with the area weighted average, all subsequent quantities are calculated from the primitive variables. In most cases, this approach will lead to a better representation of the 3D flowfield since the flow physics drive the one dimensionalization rather than the size of the grid cells. However, this method still does not ensure that mass, momentum, and energy are conserved between the 1D and 3D flowfield. Thus, if an entire duct is analyzed, the mass flow rate could dramatically change between the entrance and exit. This can lead to unexpected results for thrust, total pressure, entropy, etc. The method is attractive due to the simplicity and stability in obtaining the primitive variables with Eq. 2.

1D FLUX CONSERVED METHODOLOGY

The flux conserved method ensures that the 1D mass, momentum (in all three directions), and energy is consistent with the 3D flow field. This is particularly important if the results are passed to a 1D code for analysis. A flux conserved method employed by previous versions of massflow3d assumed that all one dimensionalized planes were planar in y-z. While this assumption is suitable for many cases such as combustors where the grids are created planar, there are also many cases where it is not practical to enforce this assumption. Figure 1 shows a typical y-z plane and a non y-z planar plane. While this is a simple box and would not require a non-planar grid, this figure is shown merely to illustrate the difference between the gridding examples.

In order to allow the flux conserved method to handle arbitrary planes, as shown in Fig. 1, the one dimensionalization was rederived. The derivations starts with the mass, momentum, and energy equations shown as Eqs. 3-7.

$$\dot{m} = \int_A \rho(\vec{V} \bullet \hat{n}) dA = \rho_o A(u_o \hat{i}_o + v_o \hat{j}_o + w_o \hat{k}_o) \qquad (3)$$

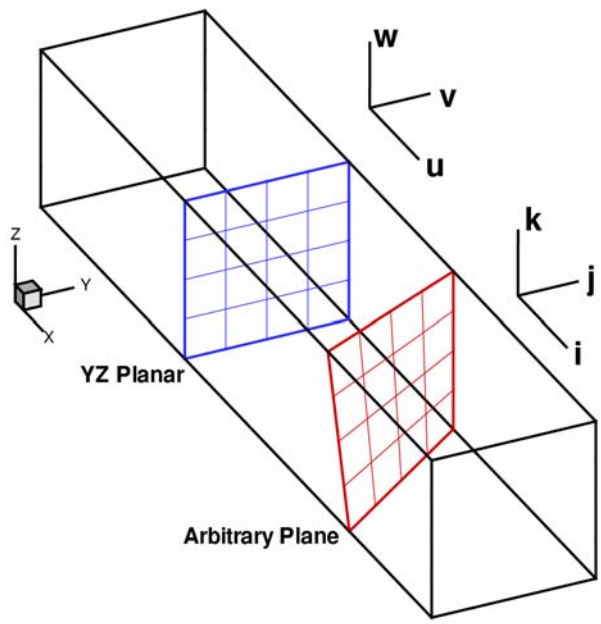

Figure 1. Possible user plane orientations.

$$mom_x = \int_A (\rho(\vec{V} \bullet \hat{n})u + P)dA = \dot{m}u_o + P_o A\hat{i}_o \quad (4)$$

$$mom_y = \int_A (\rho(\vec{V} \bullet \hat{n})v + P)dA = \dot{m}v_o + P_o A\hat{j}_o \quad (5)$$

$$mom_z = \int_A (\rho(\vec{V} \bullet \hat{n})w + P)dA = \dot{m}w_o + P_o A\hat{k}_o \quad (6)$$

$$E = \int_A (\rho e_i + P)\left|\vec{V}\right|dA \quad (7)$$

$$= \rho_o \vec{V}_o A\left(h_o + \frac{u_o^2 + v_o^2 + w_o^2}{2}\right)$$

$$E = \dot{m}\left(h_o + \frac{u_o^2 + v_o^2 + w_o^2}{2}\right) \quad (8)$$

The term on the LHS in Eqs. 3-7 defines the term. The second term (between the equal signs, or on RHS in the case of Eq. 7) of each equation gives the term for finding the variable on the LHS for the entire plane. For example, to find a given plane's $\dot{m}$, the values at each node of the plane (for all blocks at that plane) are integrated (summed in coding terms) across the plane's area. The RHS of Eqs. 3-6 and 8 show the one dimensionalized representation of each LHS term. The arbitrary planes are included here by keeping the P*A term at the end of each momentum equation. For the 1D part of each equation, the "o" subscript indicates the 1D quantity of that variable. Since the LHS of each equation is known by simply integrating across the plane, the RHS 1D quantities in Eqs. 3-6 and 8 leave the 9 unknowns: $\hat{i}_o$, $\hat{j}_o$, $\hat{k}_o$, ρ_o, u_o, v_o, w_o, P_o, and T_o. The $\hat{i}_o$, $\hat{j}_o$, and $\hat{k}_o$ unknowns represent the 1D components of the normal vector for the entire plane. While getting the normal components of an individual cell is a matter of straightforward mathematics, getting a 1D planar value of each component could be done in several ways. A simple arithmetic average was chosen for this application. Once the average i, j, and k components have been found, they are normalized by the magnitude to force them to be unit vectors. If a plane is truly planar in space then this should be mathematically the same as using the four corners of the plane to find the area vector. However, if the plane is contoured (i.e. all cells do not lie in exactly the same plane in space) then this method will find the normal vector that applies to the majority of the plane.

After finding $\hat{i}_o$, $\hat{j}_o$, and $\hat{k}_o$ the number of unknowns is reduced to 6. Adding the equation of state, shown in Eq. 9, gives 6 unknown and 6 equations.

$$P_o = \rho_o R_o T_o \quad (9)$$

where R_o is determined from Eq. 10.

$$R_o = \frac{\bar{R}}{MW_o} \quad (10)$$

and MW_o is found from Eq. 11.

$$MW_o = \frac{1}{\sum_s \left(\frac{X_{o_s}}{MW_s}\right)} \quad (11)$$

where $X_{o,s}$ is the 1D mass fraction of species s found from the mass weighted integration across the plane shown in Eq. 2 where λ is X. This is done for each species.

Solving the x, y, and z momentum equations (Eqs. 4-6) for u_o, v_o, and w_o, respectively, and substituting in Eq. 9 for P_o yields each in terms of the unknowns ρ_o and T_o as seen in the following equations (remember R_o is known from the previous equations):

$$u_o = \frac{mom_x - \rho_o R_o T_o A \hat{i}_o}{\dot{m}} \quad (12)$$

$$v_o = \frac{mom_y - \rho_o R_o T_o A \hat{j}_o}{\dot{m}} \quad (13)$$

$$w_o = \frac{mom_z - \rho_o R_o T_o A \hat{k}_o}{\dot{m}} \quad (14)$$

Now, moving all the terms of Eq. 3 to one side and calling it function 1 (F_1) yields the following equation in terms of unknowns ρ_o and T_o (remember from Eqs. 12-14 that u_o, v_o, and w_o are known in terms of ρ_o and T_o).

$$F_1 = 0 = \rho_o A(u_o \hat{i}_o + v_o \hat{j}_o + w_o \hat{k}_o) - \dot{m} \quad (15)$$

Doing the same thing for Eq. 8, and naming it F_2 yields Eq. 16.

$$F_2 = 0 = \dot{m}\left(h_o + \frac{u_o^2 + v_o^2 + w_o^2}{2}\right) - E \quad (16)$$

There are now 2 unknowns, ρ_o and T_o, and 2 equations, F_1 and F_2. In massflow3d, these equations are solved with the Newton-Raphson method. This requires the partial derivative of each function with respect to each variable. After some algebra, and remembering that u_o, v_o, and w_o are functions of ρ_o and T_o, the following results are obtained.

$$\frac{\partial F_1}{\partial \rho_o} = A\left(u_o \hat{i}_o + v_o \hat{j}_o + w_o \hat{k}_o - \frac{\rho_o R_o T_o T A_o}{\dot{m}}\right) \quad (17)$$

$$\frac{\partial F_1}{\partial T_o} = \frac{-\rho_o^2 A_o^2 R_o}{\dot{m}} \quad (18)$$

$$\frac{\partial F_2}{\partial \rho_o} = -R_o T_o A(u_o \hat{i}_o + v_o \hat{j}_o + w_o \hat{k}_o) \quad (19)$$

$$\frac{\partial F_2}{\partial T_o} = \dot{m} C_p - \rho_o R_o A(u_o \hat{i}_o + v_o \hat{j}_o + w_o \hat{k}_o) \quad (20)$$

where C_p is a function of temperature and comes from the term $\partial h_o/\partial T$. C_p can be readily calculated from McBride-Gordon curve fits.[1] The solution procedure is iterative, with each successive value of ρ_o and T_o found using the following equation.

$$\begin{bmatrix} \frac{\partial F_1}{\partial \rho_o} & \frac{\partial F_1}{\partial T_o} \\ \frac{\partial F_2}{\partial \rho_o} & \frac{\partial F_2}{\partial T_o} \end{bmatrix} \begin{bmatrix} \Delta\rho_o \\ \Delta T_o \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \quad (21)$$

With the first matrix defined as the Jacobian and represented with a J, solving for $\Delta\rho_o$ and ΔT_o yields:

$$\begin{bmatrix} \rho_{new} \\ T_{new} \end{bmatrix} = \begin{bmatrix} \rho_{old} \\ T_{old} \end{bmatrix} - J^{-1} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \quad (22)$$

With the converged values of ρ_o and T_o, P_o is found from Eq. 9. Finally, the values of u_o, v_o, and w_o are found from Eqs. 12-14. Thus, all 1D primitive values are known.

CODE METHODOLOGY AND EQUATIONS

The previous section outlined the heart of the code, which is how to obtain the 1D primitive variables from the 3D flowfield for whichever method of one dimensionalization is chosen. Figure 2 shows a flow chart of a simplified representation of the entire code. This provides an outline that will be used to detail the equations used so that the method of determining all outputs are known. The headings will correspond to steps in the code so that the user can follow along with the flow chart.

MASSFLOW3D MAIN SCRIPT

The main massflow3d script is shown in the left box. It involves system functions not handled by traditional FORTRAN programs such as searching and removing files. Upon running, the script reads the user's input deck, checks for errors (invalid inputs), calculates block information (such as dummy lines, sizes, etc.) and then writes the information to an internal input deck. The main FORTRAN program is run with the internal input deck as an input. When the FORTRAN program is complete, the script preplots all output dat files (if the user has the preplotting flag on) and then cleans up any temporary files from the directory.

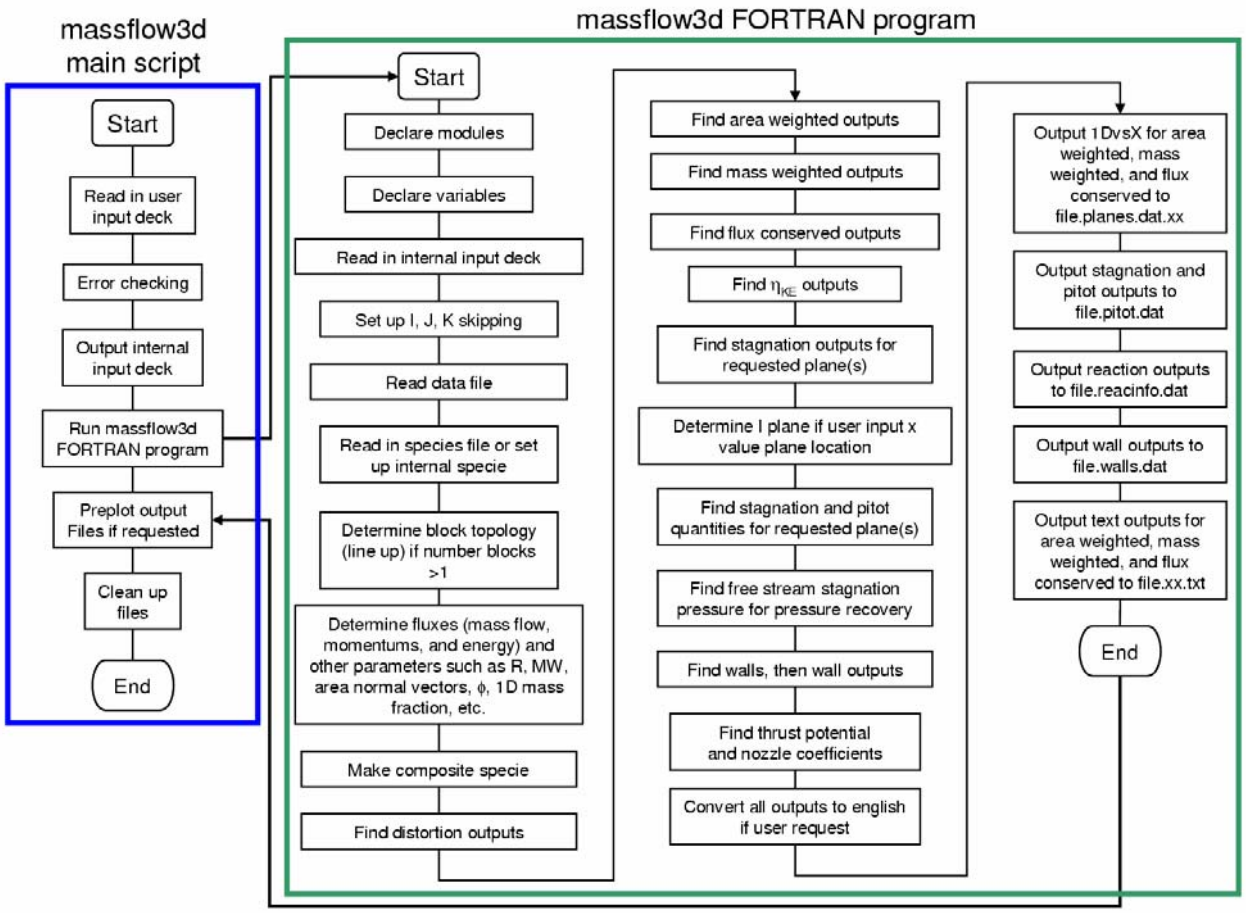

Figure 2. Simplistic flow chart of code.

DECLARE MODULES -- READ INPUT DATA

The massflow3d FORTRAN program, called by the main script, begins by declaring all modules and variables. Next, the internal input deck is read in and more error checking is done. If the user specified I, J, or K skipping, then the memory for reading in the input data is allocated appropriately. Then, the input data is read in with the user's specified skipping.

READ IN SPECIES FILE

The species curve fits either use an internal set of curves or are found by reading in the user specified file, depending on the user's input (see user manual species file section of input deck description). In either case, the species data is used in SHIP[2] format. A sample species is shown in Fig. 3.

```
c*********************************************************************
c  Species                  Mol Wt.
   N2                       28.01348
c  Sutherland Laminar Viscosity (Derived from NASA RP 1311).
   0.155742E-05  169.64
c  Sutherland Thermal Conductivity (Derived from NASA RP 1311).
   0.273381E-02  280.35
c  McBride Thermodynamic Coefficients, 200-6000 K, NASA TM 4513.
 2.95257626E+00 1.39690057E-03-4.92631691E-07 7.86010367E-11-4.60755321E-15
-9.23948645E+02 5.87189252E+00 3.53100528E+00-1.23660987E-04-5.02999437E-07
 2.43530612E-09-1.40881235E-12-1.04697628E+03 2.96747468E+00 0.00000000E+00
c*********************************************************************
```

Figure 3. species SHIP format for N2.

The dummy lines shown with text are required at the line numbers where they are shown but are not discussed here. First, the species name and molecular weight are required. This is followed by the Sutherland coefficients for the calculations of laminar viscosity and thermal conductivity. However, the coefficients are not the standard μ_r, T_r (where the r subscripts here represent a reference quantity), and S. In fact, it is clear only two coefficients are present. The standard Sutherland formulation[3] for a quantity such as μ or k is shown below (where λ is used for the quantity of interest).

$$\lambda = \lambda_r \left(\frac{T}{T_r}\right)^{\frac{3}{2}} \left(\frac{T_r + S}{T + S}\right) \tag{23}$$

The SHIP format rearranges the equation to the form:

$$\lambda = \lambda_r \frac{T_r + S}{T_r^{\frac{3}{2}}} \left(\frac{T^{\frac{3}{2}}}{T + S}\right) = \Phi \left(\frac{T^{\frac{3}{2}}}{T + S}\right) \tag{24}$$

This formulation requires only two inputs, Φ and S. The final section is the standard curve fit polynomials and are arranged in the following order

$$\begin{bmatrix} a_1^H & a_2^H & a_3^H & a_4^H & a_5^H \\ a_6^H & a_7^H & a_1^L & a_2^L & a_3^L \\ a_4^L & a_5^L & a_6^L & a_7^L & h_f^o \end{bmatrix} \tag{25}$$

where $h_f^o / \bar{R}$ is the enthalpy of formation over $\bar{R}$, and the L and H superscripts represent the low and high temperatures ranges. The low temperature range is used for temperatures between 200 and 1000 K, while the high temperature range is valid from 1000 to 6000 K. These polynomials are used for thermally perfect calculations of C_p, enthalpy, and entropy as follows:

$$C_p = R(a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4) \tag{26}$$

$$h = R\left(a_1 T + \frac{a_2 T^2}{2} + \frac{a_3 T^3}{3} + \frac{a_4 T^4}{4} + \frac{a_5 T^5}{5} + a_6\right) \tag{27}$$

$$s = R\left(a_1 \ln(T) + a_2 T + \frac{a_3 T^2}{2} + \frac{a_4 T^3}{3} + \frac{a_5 T^4}{4} + a_7\right) \tag{28}$$

DETERMINE BLOCK TOPOLOGY

If more than one block is entered, the program automatically determines how all blocks line up relative to one another. The method is outlined here so that a user may understand how the program attempts to determine the block topology in case there is a problem with a particular case. If there is no override file present (see “other features” section near the end of the input manual), then the program determines the block topology automatically. This is done by first finding the average x value of every I plane for every block. Every plane is then searched to find the I plane with the smallest average x value. This is dubbed “absolute” plane number 1. Once this plane is found, every point on its perimeter (nodes along that plane’s minimum and maximum J and K boundaries) are checked against the perimeter of every I plane of every other block. If at least a given number of points match on the perimeter to within the code’s tolerance (code default set to at least 2 matches) the program defines that I plane for that block to be in the same absolute plane as the original block being searched against. When this occurs, the newly found block’s perimeter is also searched for other blocks I planes that match its absolute plane. This continues until there are no perimeter matches. This defines all I planes for all the blocks that lie in the first absolute plane. The program then searches all the remaining I planes of all blocks to find the minimum x value, calls it absolute plane 2, and the process starts over again. This continues until every plane of every block has been used.

The final step in the absolute plane determination is to remove overlapping planes. A sample 2D plane is shown in Fig. 8 of the input manual section. From this figure it is clear the ending plane of the top left zone overlaps with the beginning of the top right zone. Since these planes have the same perimeter points, they would be put in the same absolute plane; however, this would double book keep this region of the flowfield. To remove this possibility, the program searches all interior nodes of any overlapping blocks. If the nodes of a block’s I maximum plane are coincident with the nodes of another block’s I minimum plane (default set to at least 4 coincident points in plane), then the planes are assumed coincident and either the minimum or maximum plane’s absolute plane number is set to zero so that it is not used. A setting in the massflow3d FORTRAN program determines if the minimum (the default setting) or maximum plane is removed. The bottom zone’s absolute plane at this region is unaffected by these operations since it has no interior matching points and the corresponding plane is not a minimum or maximum plane of the block. Note that, as stated, only the interior points are searched when checking for coincidence. This is done because of cases similar to Fig. 4. This example shows the symmetry plane of a 3D backward facing step solution. Note that block 1 ends where blocks 2 and 3 begin. If every point in block 1’s I maximum plane were searched for coincidence against all other block’s I minimum planes in the same absolute plane (which is both blocks 2 and 3), then both blocks 2 and 3’s I minimum planes would be removed (if minimum planes were set for removal) due to point A.

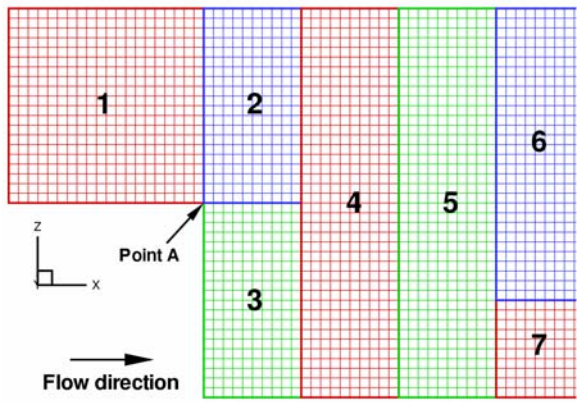

Figure 4. Finding coincident planes.

While it is clear that block 2's planes should be removed, block 3's should not since it is not coincident with block 1's I maximum plane. By searching only the interior nodes, point A does not get examined, resulting in only block 2's I minimum plane getting removed. This is the desired behavior.

DETERMINE FLUXES AND OTHER PLANAR INTEGRATED PARAMETERS

After all planes making up each absolute plane have been determined, the flux values at each absolute plane can be determined by integrating across each absolute plane sequentially. First the area is found at each cell assuming a general quadrilateral shape. Since the program operates at each node, rather than the cell centers, the method of finding the area depends on whether an interior, perimeter, or corner node is being examined. Figure 5 shows several nodes in a plane section at a corner along with how the area is determined for each type of node. Although all nodes are shown orthogonal to each other, this is not required. The equations used in the code can handle any quadrilateral. This allows skewed cells to be calculated correctly.

To find the area of a given interior node (labeled point 5 in Fig. 5) the points surrounding it are found, labeled 1-9. The points A-D are found by taking the average of each quadrants values, e.g. point A is found from the average x, y, and z values of points 1, 2, 4, and 5. Point B is found from the average x, y, and z values of points 2, 3, 5, and 6. Similarly for points C and D. Once points A-D have been found, the sides (a, b, c, and d) and

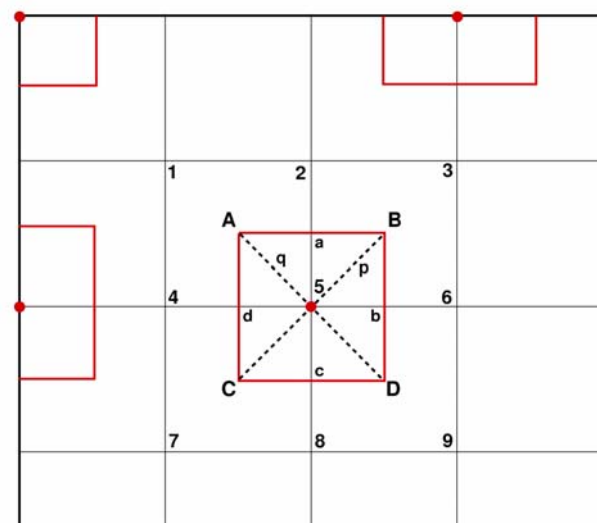

Figure 5. Finding area around any node.

diagonals (p and q) are found. Finally, the area of the general quadrilateral is found from Eq. 29.[4]

$$A = \frac{1}{4}\sqrt{4p^2q^2 - (b^2 + d^2 - a^2 - c^2)^2} \qquad (29)$$

A similar approach is taken at the other points on the boundaries or in the corner with the same mathematics if one simply notes that the points collapse. For example, if the same numbering approach is applied to the unnumbered node above point 3, then point 1=4, 2=5, and 3=6 due to the boundary. A similar approach can be used for the other boundary nodes and the corner nodes. The area of each node in the absolute plane is summed to find the absolute plane's total area.

With the diagonals of the node known, taking their cross product and finding the unit vector of the resulting vector yields the area normal vector shown in Eq. 30.

$$\hat{n} = \frac{\vec{p} \times \vec{q}}{|\vec{p} \times \vec{q}|} = \hat{i}_o + \hat{j}_o + \hat{k}_o \qquad (30)$$

After finding every node's $\hat{i}_o$, $\hat{j}_o$, and $\hat{k}_o$ value, the average values of $\hat{i}_o$, $\hat{j}_o$, and $\hat{k}_o$ are found for the plane, then normalized by their magnitude to create a 1D absolute plane normal vector. Since the direction of J and K is specified by the user which could result in a cross product facing upstream or downstream, the area normal vector is always forced to face downstream. This applies to the individual cells as well as the absolute plane value.

To find the 1D mass fraction of each species in the absolute plane, the mass fractions are integrated across the absolute plane using a mass weighted average as shown in Eq. 2 where λ is defined as the mass fraction of species s (X_s). If the user inputs mole fractions instead of mass fractions, then prior to using Eq. 2 each cell's molecular weight and mass fraction must be found using Eqs. 31 and 32, respectively,

$$MW = \sum Y_s MW_s = \frac{1}{\sum\left(\frac{X_s}{MW_s}\right)} \quad (31)$$

$$Y_s = X_s \frac{MW}{MW_s} \quad (32)$$

Once the 1D mass fractions are known, then Eqs. 31 and 32 are used again, but this time on a 1D level (instead of node level) to find MW_o (1D MW) and Y_{os} (1D mole fraction of species s).

Since the code is integrating over each absolute plane at this point, there are many other parameters that can also be calculated. The mass capture is found from the LHS and middle terms in Eq. 3. The momentum in the x, y, and z directions are found from Eq. 4-6. However, while Eqs. 4-6 are used to obtain the 1D quantities, the user has the option of requesting the momentums with a (P-P_∞)*A term, rather than just P*A. Thus, two separate momentums (for each direction) are tracked, one for the 1D analysis and another based on the user's input of P_∞. While the equations could have been developed with a P-P_∞ term in them, they were not in order to simplify the analysis. Since the P_∞ term is a constant, it falls out of the equations and does not affect the one dimensionalization. The 1D energy can be found by integrating Eq. 33 (derived from Eq. 7) over the area

$$E = \dot{m}\left(h + \frac{u^2 + v^2 + w^2}{2}\right) \quad (33)$$

Since the energy and mass flow are both flux conserved quantities, the flux conserved total enthalpy is then found from Eq. 34.

$$h_T = \frac{E}{\dot{m}} \quad (34)$$

The average x value of each absolute plane is found with a simple arithmetic average of every node in the plane. The average value is used in the output files for plotting against 1D properties.

The following equations are summed for each cell across the absolute plane to get the moments about the user input x, y, and z axes, respectively.

$$Moment_x = mom_z(y - y_o) - mom_y(z - z_o) \quad (35)$$

$$Moment_y = mom_x(z - z_o) - mom_z(x - x_o) \quad (36)$$

$$Moment_z = mom_y(x - x_o) - mom_x(y - y_o) \quad (37)$$

where x_o, y_o, and z_o are the user input axes location. Note from the equations above, the positive direction for each moment is determined using the right hand rule for each direction. Note that if the user inputs a non zero P_∞ term, than the momentums used in Eqs. 35-37 use a (P-P_∞)*A term, instead of just P*A.

Finally, the fuel equivalence ratio is calculated for every cell as well as the absolute plane's total ϕ. The following simple model is used to obtain the stoichiometric fuel to oxygen ratio.

$$2H_2 + O_2 \rightarrow 2H_2O$$

From this equation the fuel to oxygen ratio is:

$$\left.\frac{F}{O}\right)_{stoic} = \frac{mass_H}{mass_O} = \frac{2(2.01588)}{1(31.9988)} = 0.126 \quad (38)$$

Using the stoichiometric fuel to oxygen ratio, the fuel equivalence ratio is found from Eq. 39.

$$\phi = \frac{\left.\frac{F}{O}\right)_{actual}}{\left.\frac{F}{O}\right)_{stoic}} = \frac{mass_H}{mass_O(0.126)} \quad (39)$$

where $mass_H$ and $mass_O$ represent the mass for all molecules present in all species. For example, for $mass_H$, the H from the species, H, H_2, OH, H_2O, etc. must be accounted for. Since regions of the flowfield may have very large amounts of H with little to no O, such as near injectors, a limit of 88.888 is placed on ϕ. All 8s were chosen so that it was clear the number was being enforced. This limit ensures the code does not overflow at extremely low values of O. Since the total mass of H and O needs to be found for ϕ, the absolute plane's water based mixing, reaction, and combustion efficiencies can also be found from Eqs. 40-42,[5] respectively, by also finding the total mass of H_2O.

$$\eta_m = \frac{\text{mass flow of locally least available reactant}}{\text{mass flow of globally least available reactant}} = \frac{\text{LLAR}}{\text{GLAR}}$$

$$\eta_m = \frac{\sum \dot{m} \min\left[X_{\text{all H}}, \frac{F}{O}\right)_{\text{stoic}} X_{\text{all O}}\right]}{\min\left[\sum \dot{m} X_{\text{all H}}, \sum \dot{m} \frac{F}{O}\right)_{\text{stoic}} X_{\text{all O}}\right]} \quad (40)$$

$$\eta_r = \frac{\sum \dot{m} X_{H_2O}}{\frac{MW_{H_2O}}{MW_{H_2}} \bullet \text{LLAR}} \quad (41)$$

$$\eta_c = \frac{\sum \dot{m} X_{H_2O}}{\frac{MW_{H_2O}}{MW_{H_2}} \bullet \text{GLAR}} = \eta_m \bullet \eta_r \quad (42)$$

These definitions are preferred over others since they allow the mixing and reaction components of the combustion efficiency to be independently examined. This allows the user to intelligently implement design changes to increase the lagging efficiency.

A parameter defined here as the local combustion efficiency can be found by examining η_c on a cell by cell basis. For an individual cell, η_m is 1 by default since LLAR = GLAR. This implies from the RHS of Eq. 42 that $\eta_c = \eta_r$. Thus, $\eta_{c\ local}$, shown in Eq. 43, shows the fraction of H_2O actually created versus what would have been possible given ideal combustion conditions.

$$\eta_{c\text{-local}} = \frac{\dot{m} X_{H_2O} \frac{MW_{H_2}}{MW_{H_2O}}}{\dot{m} \min\left[X_{\text{all H}}, \frac{F}{O}\right)_{\text{stoic}} X_{\text{all O}}\right]} \quad (43)$$

MAKE COMPOSITE SPECIES

With the 1D mass fractions of each species and 1D molecular weight known, a composite[6] species can be generated for each absolute plane that is thermodynamically equivalent to the original mixture. This allows all subsequent calculations in the code to be done more rapidly since only one call is needed to the thermodynamic functions of C_p, h, and s. Eq. 44, taken directly from reference 5, is done for each polynomial coefficient i shown in Eq. 25 except for the entropy coefficient, a_7, which uses Eq. 45.

$$a_{i\ \text{composite}} = \sum_{s=1}^{\text{number species}} X_s a_i \quad (44)$$

$$a_{7\ \text{composite}} = \sum_{s=1}^{\text{number species}} X_s (a_7 - \text{LN}(X_s)) \quad (45)$$

Note that the entropy coefficient, a_7, now includes the mass fraction term so that term should not be added to the entropy term in subsequent calculations.

FIND DISTORTION OUTPUT

Three distortion parameters[7] are output for each of the area weighted, mass weighted, and flux conserved approaches. The distortion area, momentum, and divergent (off axis) velocity efficiencies are shown in Eqs. 46-48, respectively.

$$\eta_a = \frac{\int \rho u dA}{\rho_{ref} u_{ref} A} \quad (46)$$

$$\eta_f = \frac{\int \rho u^2 dA}{\rho_{ref} u_{ref}^2 A} \quad (47)$$

$$\eta_{dv} = 1 + \frac{\int \rho u (v^2 + w^2) dA}{(\rho_{ref} u_{ref}) u_{ref}^2 A} \quad (48)$$

Note that these distortion equations were developed for planar flow fields. This is evident from the use of the u velocity as the main velocity component and the inclusion of v and w in the off axis velocity term. Therefore, care should be taken when using these parameters when the reported plane unit vector (output in the text output file) is not sufficiently pointed in the x direction (where “sufficiently” is dependent on the users acceptable level of error).

In the above equations, u_{ref} and ρ_{ref} are unknowns. u_{ref} is found from Eq. 49.

$$u_{ref} = \sqrt{\frac{\int (\rho u) u^2 dA}{\int \rho u dA}} \quad (49)$$

ρ_{ref} is found by first finding P_{ref} and h_{ref} as shown in Eqs. 50 and 51, respectively.

$$P_{ref} = \frac{\int P dA}{\int dA} \tag{50}$$

$$h_{ref} = \frac{\int \rho u h dA}{\int \rho u dA} \tag{51}$$

With the value of h_{ref} known, a simple iteration is used to determine the value of T_{ref} required for h_{ref}. With T_{ref} and P_{ref} known, the equation of state is used to find ρ_{ref}.

FIND AREA WEIGHTED, MASS WEIGHTED, AND FLUX CONSERVED OUTPUTS

The primitive variables for each 1D approach are found using the methods outlined at the beginning of this document. Once the primitive variables are known, the subsequent calculations are trivial. For the flux conserved approach the mass capture, momentums, and energy are already known since they are required to solve the 1D equations and the 1D results are, by definition of the approach, equal to the calculated values. For the mass and area weighted approaches, the flux quantities are calculated using Eqs. 3-6 and 8 but using the 1D values obtained for each approach. The P*A term uses $(P-P_\infty)$*A where appropriate as discussed previously.

The moment for the flux conserved output is the true moment felt by the flowfield. The incremental moments are found at each node using Eqs. 35-37 and summed over the absolute plane. However, the moments for the area and mass weighted outputs are done differently. The 1D primitive values are applied to every node in the domain and then the same approach used in the flux conserved method is applied. Thus, the moment is that caused by the 1D results applied at the plane's center of area. This indirectly allows that user to see how different the 1D primitive variable results are between the flux conserved and weighted approaches by examining how different the resulting moments are at any given plane.

The remaining discussion in this section applies to all three 1D approaches. Since the 1D temperature is known, C_p, h, and s are found from Eqs. 26-28. C_v, γ, speed of sound, and the Mach number are found from the standard expressions shown in Eqs. 52-55.

$$C_v = C_p - R \tag{52}$$

$$\gamma = \frac{C_p}{C_v} = 1 + \frac{R}{C_v} = \frac{C_p}{C_p - R} \tag{53}$$

$$a = \sqrt{\gamma R T} \tag{54}$$

$$M = \frac{u^2 + v^2 + w^2}{a} \tag{55}$$

The Mach number shown in Eq. 55 is based on the total velocity. Additional Mach numbers based on just u, v, and w are also calculated.

Next the calorically and thermally perfect 1D stagnation properties are found. The calorically perfect properties assume constant specific heats and are found from the following equations:

$$P_T = P\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}} \tag{56}$$

$$T_T = T\left(1 + \frac{\gamma - 1}{2} M^2\right) \tag{57}$$

The stagnation density, ρ_T, is found using the equation of state with total conditions.

The thermally perfect stagnation conditions are only slightly more complicated. First, the thermally perfect total temperature is found with a simple iteration loop to find the total temperature that satisfies the previously found total enthalpy using the enthalpy definition of Eq. 27. To find the thermally perfect total pressure, we begin with the Tds equation (Eq. 58) (sometimes also called the combined first and second law of thermodynamics[8]). The following analysis, of which Eqs. 58-62 were taken from reference 9, yields the thermally perfect total pressure.

$$T ds = dh - \upsilon dP \tag{58}$$

Using the definition of specific volume, the equation of state, noting that the thermally perfect definition of h is $\int C_p dT$, and integrating, gives the following equation.

$$ds = C_p \frac{dT}{T} - R \frac{dP}{P} \tag{59}$$

Integration between the static and total states yields:

$$s_T - s = \int_T^{T_T} \frac{C_p}{T} dT - R\left(LN\frac{P_T}{P}\right) \tag{60}$$

If the stagnation process is assumed to be isentropic, then $s = s_T$ and the LHS drops to zero.

$$R\left(LN\frac{P_T}{P}\right) = \int_T^{T_T} \frac{C_p}{T} dT \tag{61}$$

Rearranging yields the thermally perfect total pressure.

$$\frac{P_T}{P} = e^{\left(\frac{1}{R}\int_T^{T_T} \frac{C_p}{T} dT\right)} \tag{62}$$

where e is Napier's constant (~2.71828). The thermally perfect stagnation density, ρ_T, is found using the equation of state with the thermally perfect total conditions. Noting that $\int (C_p/T)dT$ is entropy, s, which is clear by examining Eqs. 26 and 28, provides a simple way to evaluate Eq. 62.

The laminar viscosity and thermal conductivity for each species is found from Eq. 24. If more than one species is present, then Wilke's Law[3], shown in Eq. 63, is used to find the mixture value.

$$\mu_{mixture} \approx \sum_{s=1}^{\text{number species}} \frac{X_s \mu_s}{\sum_{n=1}^{\text{number species}} X_n \Phi_{s,n}} \tag{63}$$

where:

$$\Phi_{s,n} = \frac{\left(1 + \sqrt{\frac{\mu_s}{\mu_n}} \sqrt[4]{\frac{MW_n}{MW_s}}\right)^2}{\sqrt{8 + 8\frac{MW_s}{MW_n}}} \tag{64}$$

If more than one absolute plane is input then the entropy increase relative to the first plane is found from Eq. 65 which is found from Eq. 60 but using the 1D states of absolute planes 1 and i instead of the static and total conditions.

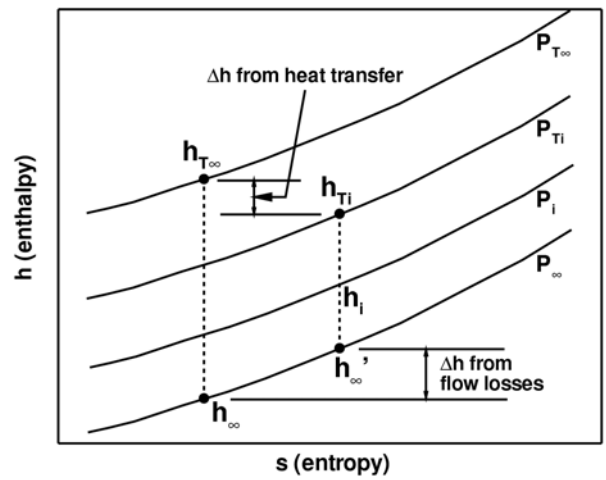

Figure 6. h-s diagram to find η_{KE}.

$$\Delta s_{1 \to i} = s_i - s_1 - R\ LN\left(\frac{P_i}{P_1}\right) \tag{65}$$

where the 1 subscript represents the first plane of the input block, and i represents the current plane. The value of s is found from Eq. 28.

FIND η_{KE} OUTPUTS

If a free stream pressure greater than zero is input on the "free stream values" line near the end of the input deck, then the input P_∞, T_∞, and V_∞ are used to find the kinetic energy efficiency values. Figure 6 shows a h-s diagram used to find η_{KE}. The free stream values are shown with an ∞ subscript and the plane at which the efficiency is being evaluated is denoted with an i subscript.

η_{KE} is defined in Eq. 66.[10]

$$\eta_{KE} = \frac{h_{T_i} - h_\infty'}{h_{T_\infty} - h_\infty} \tag{66}$$

Each of these terms is shown as a point in Fig. 6. The first quantities found are h_i and h_∞ which are found using T_i (while the i subscript has been kept for consistency, T_i is actually just T_o, the 1D temperature at plane i) and T_∞, respectively, in Eq. 27. Then, $h_{T\infty}$ is found along a constant entropy line using Eq. 67 (found by combining Eqs. 33 and 34).

$$h_{T\infty} = h_\infty + \frac{V_\infty^2}{2} \tag{67}$$

Finally, h_∞' (read as free stream enthalpy prime) is found using the following method. Starting with the isentropic relation:

$$\frac{T_\infty'}{T_i} = \left(\frac{P_\infty}{P_i}\right)^{\frac{\gamma-1}{\gamma}}$$

If constant specific heats are assumed, then the value of C_p between the free stream and plane i states is the same. Multiplying the LHS by C_p/C_p yields enthalpies on the LHS. Solving for h_∞' yields Eq. 68.

$$h_\infty' = h_i\left(\frac{P_\infty}{P_i}\right)^{\frac{\gamma-1}{\gamma}} \tag{68}$$

Note that the numerator and denominator of Eq. 66 is a total enthalpy minus a static enthalpy. From Eq. 67 it is clear that this subtraction results in the kinetic energy. Further, Eq. 66 examines the ratio of the kinetic energy at the current station to the free stream, hence the name kinetic energy efficiency. Note that the numerator does not simply include the kinetic energy (since the second term is h_∞' instead of h_i) but also includes the enthalpy used to increase the pressure from P_∞ to P_i since a pressure increase is a desirable result for typical inlets. As Fig. 6 shows, with the kinetic energy and some of the static enthalpy change accounted for, only other changes in the static enthalpy remain. They are the heat transfer and flow losses which are inefficiencies and should not be included in the numerator of Eq. 66. With all terms found, η_{KE} is easily found. If the adiabatic kinetic energy efficiency, η_{AKE}, is desired, the heat transfer terms can be left in the numerator of Eq. 66 resulting in Eq. 69.

$$\eta_{AKE} = \frac{h_{T_\infty} - h_\infty'}{h_{T_\infty} - h_\infty} \tag{69}$$

DETERMINE I PLANE IF USER INPUT X VALUE PLANE LOCATION

If the user inputs an x/i value of x, then all the input values for the locations of plane specific output files are requested as x locations. At this point, the code iterates through the average x value of each absolute plane and finds the closest value to the input x value for each output file. The output file data is then output at this absolute plane.

FIND STAGNATION AND PITOT QUANTITIES FOR REQUESTED PLANE(S)

The stagnation and pitot quantities calculated in this section of the code are found at every node in the user requested absolute plane. The static values of ρ, u, v, w, P, T, and any mass or mole fractions are simply the input values at the node. The values of MW, R, C_p, C_v, γ, a, A, $\vec{V}$, P_T, T_T, ρ_T, h_T, M, at the node are found using the same equations previously discussed.

There is an additional term found called the user velocity used in the calculation of the pitot conditions. This ensures the code is finding the correct pitot pressure for the orientation of any pitot rakes the user wishes to compare the results to. The input deck determines which velocity components will be used to find the user velocity (u, v, w, some combination of them, or the velocity normal to the I face of the cell). This is described in more detail in the input deck section of the user manual. The user Mach number is found using the user velocity. The first pitot pressure found uses the empirical relationship of Eq. 70.

$$P_{pitot} = 0.93\rho\left|\vec{V}_{user}\right|^2 \tag{70}$$

The true pitot pressure is found using the Rayleigh pitot tube formula,[11] shown in Eq. 71, if the user Mach number is greater than or equal to one. If the user Mach number is less than 1, then the pitot pressure is simply the total pressure. In this case, the thermally perfect total pressure is used to obtain as accurate a pitot pressure as possible.

$$P_{pitot} = \frac{1-\gamma+2\gamma M_{user}^2}{\gamma+1}\left(\frac{(\gamma+1)M_{user}^2}{4\gamma M_{user}^2 - 2(\gamma-1)}\right)^{\frac{\gamma}{(\gamma-1)}} \tag{71}$$

Both pitot pressure equations are used so that the user can periodically make comparisons between the two to validate the use of Eq. 70's pitot pressure.

FIND FREE STREAM STAGNATION PRESSURE FOR PRESSURE RECOVERY

The free stream stagnation pressure is found using the same method as that for the 1D stagnation pressure but the free stream values of P_∞, T_∞, and V_∞ are used as the static quantities. Both the calorically and thermally perfect stagnation pressures are found. These terms are used as the denominator for a pressure recovery to free stream output.

FIND WALLS, THEN WALL OUTPUTS

For the wall output, the code first determines which cells on every I, J, and K minimum and maximum surface are walls. For any given cell, the magnitude of the velocity at the corners is found. If the velocity magnitude of all cell's corners is less than a user input maximum velocity, then the cell is tagged a wall and the subsequent calculations follow.

First, the area, normal vectors, surface vectors, and normal distance (d) to the first cell off the surface are found. Unlike the area calculations of the 1D analysis, the area is not centered around the node, rather, the actual cell corners are used to determine the area with the primitive variables found at the cell center with a simple arithmetic average. The inviscid forces in the x, y, and z directions are found from Eqs. 72-74, respectively.

$$F_{ix} = (P - P_\infty)A\hat{i}_n \tag{72}$$

$$F_{iy} = (P - P_\infty)A\hat{j}_n \tag{73}$$

$$F_{iz} = (P - P_\infty)A\hat{k}_n \tag{74}$$

To obtain the viscous force (skin friction), the laminar viscosity at the cell is found using Eqs. 24 and 63, then the skin friction is found using Eqs. 75-77.

$$F_{vx} = \mu A\hat{i}_s \frac{\left|\vec{V}_{\substack{\text{one node}\\ \text{off surface}}}\right|}{d} \tag{75}$$

$$F_{vy} = \mu A\hat{j}_s \frac{\left|\vec{V}_{\substack{\text{one node}\\ \text{off surface}}}\right|}{d} \tag{76}$$

$$F_{vz} = \mu A\hat{k}_s \frac{\left|\vec{V}_{\substack{\text{one node}\\ \text{off surface}}}\right|}{d} \tag{77}$$

The inviscid moment, calculated from the inviscid forces, is found using Eqs. 78-80.

$$\text{Moment}_x = F_{iz}(y - y_o) - F_{iy}(z - z_o) \tag{78}$$

$$\text{Moment}_y = F_{ix}(z - z_o) - F_{iz}(x - x_o) \tag{79}$$

$$\text{Moment}_z = F_{iy}(x - x_o) - F_{ix}(y - y_o) \tag{80}$$

The viscous moment is calculated using the same equations but using F_v instead of F_i. The total forces and moments are the simple addition of the inviscid and viscous forces and moments.

The current version of the code does not read in turbulent quantities such as turbulent kinetic energy or dissipation. Therefore, the turbulent viscosity, which should be included in the skin friction calculation, can not be found and only the laminar viscosity is used.

The heat transfer is found by first finding k using Eqs. 24 and 63 with k substituted for μ. The heat transfer is then given by Eq. 81.

$$Q = kA\frac{\left(T_{\substack{\text{one node}\\ \text{off surface}}} - T_{\text{surface}}\right)}{d} \tag{81}$$

The heat flux is found by dividing the heat transfer by the area. The values of all of the above parameters are summed for each absolute plane for output. All of the above parameters, except heat flux, also have a "running" total that gives the parameter's total up to and including the current plane.

FIND THRUST POTENTIAL AND NOZZLE COEFFICIENTS

If the user inputs an A_{exit} greater than 0 in the input deck, the following thrust potential and nozzle efficiency calculations are performed using the user's input A_{exit}, P_{ref}, C_S, and C_{FG} for the area weighted, mass weighted, and flux conserved 1D values. There are two TP calculations, expansion to A_{exit} and expansion to P_{ref}. Both are 1D analyses and as such use the 1D values wherever appropriate. As with the distortion parameters, this analysis was developed for 1D analysis and as such the planes are assume y-z planar such that the 1D velocity vector is normal to the y-z plane.

The A_{exit} expansion[12] uses the current absolute plane as the inflow plane (denoted with an i in the following analysis). The entropy at the inflow plane is found using Eq. 82.

$$s_i = s - R\,LN(P) \tag{82}$$

where s is found from Eq. 28. The total enthalpy, h_{Ti}, is also required but was already found in the previous 1D analysis. The area expansion thrust potential seeks to find the ideal thrust (hence the "potential") that can be obtained at the exit given the current flow state at plane

i. Thus, the total enthalpy and entropy at plane i are assumed to equal those at the exit (i.e. no losses). The species are also assumed frozen from plane i to the exit plane. Equating the inlet and exit entropy and total enthalpy yields the following two equations written in the F_1 and F_2 format used earlier in the 1D analysis for a Newton-Raphson solution. The two unknowns are P_{exit} and T_{exit}.

$$F_1 = 0 = s_{exit} - R\,LN(P_{exit}) - s_i \tag{83}$$

$$F_2 = 0 = h_{T,\,exit} + \frac{1}{2}\left(\frac{\dot{m}_i R}{A_{exit}}\right)^2 \frac{T_{exit}^2}{P_{exit}^2} - h_{T,\,i} \tag{84}$$

where $h_{T,exit}$ and s_{exit} are found using T_{exit} in Eqs. 27 and 28, respectively. The middle term on the RHS of Eq. 84 comes from the kinetic energy term of the total enthalpy by using $\dot{m} = \rho u A$ and the equation of state.

The partial derivatives of these function with respect to the unknowns are shown in Eqs. 85-88.

$$\frac{\partial F_1}{\partial T_{exit}} = \frac{a_1}{T_{exit}} + a_2 + a_3 T_{exit} + a_4 T_{exit}^2 + a_5 T_{exit}^3 \tag{85}$$

where the RHS of Eq. 85 is ds/dT and was found from differentiating Eq. 28. The polynomial coefficients are for the composite species.

$$\frac{\partial F_1}{\partial P_{exit}} = \frac{-R}{P_{exit}} \tag{86}$$

$$\frac{\partial F_2}{\partial T_{exit}} = C_p + \left(\frac{\dot{m}_i R}{A_{exit} P_{exit}}\right)^2 T_{exit} \tag{87}$$

$$\frac{\partial F_2}{\partial P_{exit}} = \frac{-1}{P_{exit}^3}\left(\frac{\dot{m}_i R T_{exit}}{A_{exit}}\right)^2 \tag{88}$$

With these equations, P_{exit} and T_{exit} are found using equations similar to Eqs. 21 and 22. With P_{exit} and T_{exit} known, ρ_{exit} is found from the equation of state. Then, using Eq. 89 u_{exit} can be found.

$$\dot{m} = \rho_{exit} u_{exit} A_{exit} \rightarrow u_{exit} = \frac{\dot{m} R T_{exit}}{P_{exit} A_{exit}} \tag{89}$$

Finally, the TP of the expansion to a given exit area is found from the difference in the stream thrust between the exit and inlet conditions, given by Eq. 90.

$$TP = \dot{m} u_{exit} + (P_{exit} - P_{ref}) A_{exit} - (\dot{m} u_i + (P_i - P_{ref}) A_i) \tag{90}$$

The second TP found is an expansion to the input P_{ref} (intended as an expansion to the free stream pressure). The first step is to find T_{exit} using Eq. 83. Since P_{exit} is known ($P_{exit} = P_{ref}$ in this case) a simple iterative loop can be used to find T_{exit}. With P_{exit} and T_{exit} both known, Eq. 84 can be solved for A_{exit} directly. In this case, A_{exit} is the required exit area that would be needed to ideally expand to P_{ref}. Note that the actual area would have to be larger due to flow losses. The thrust potential is then calculated using Eqs. 89 and 90, where the P_{exit}-P_{ref} term will be zero since $P_{exit} = P_{ref}$.

The nozzle coefficients[13] C_{FG} and C_S are related to the TP because they deal with the stream thrust. C_{FG} and C_S are defined in Eqs. 91 and 92.

$$C_{FG} = \frac{F_{exit}}{F_{ideal}} \tag{91}$$

$$C_S = \frac{F_{exit} + P_\infty A_{exit}}{F_{vac}} \tag{92}$$

C_{FG} represents how efficiently the flow was expanded by comparing the stream thrust to the maximum obtainable thrust available through an ideal expansion to free stream pressure. C_S represents how efficiently the flow was expanded by comparing the stream thrust to the maximum obtainable thrust available through an ideal expansion to a given exit area regardless of the free stream pressure. Thus, C_S is primarily tied to the flow losses, whereas C_{FG} is driven by how near the exit pressure is to the free stream pressure.

C_{FG} is the actual nozzle gross thrust divided by the ideal gross thrust, F_{ideal}. C_S is the actual nozzle vacuum thrust divided by the ideal vacuum stream thrust, F_{vac}. F_{vac} is the ideal expansion to A_{exit} and therefore was found in the first TP calculation, while F_{ideal} is the ideal expansion to P_∞ and was therefore found in the second TP calculation. Thus, the only unknowns in both equations are F_{exit}, C_S and C_{FG}. As the user manual states, because the equations are dependent, the user must input one and only one value of C_S or C_{FG}, otherwise Eqs. 91 and 92 will be over constrained. Since all other parameters are known, if C_S is input, then Eq.

91 is solved for F_{exit}; otherwise, with C_{FG} input, Eq. 92 is solved for F_{exit}. With F_{exit} known, the other equation is easily solved.

CONVERT ALL OUTPUTS TO ENGLISH IF REQUESTED

If the user requested english output units, the code establishes conversion factors for each unit of measure. For example, a length conversion, a force conversion, a speed conversion, etc. These conversions are applied to each parameter that will be output.

OUTPUT FILES

All files that are output, as well as the contents of each are discussed in the user manual at the end of this document.

CODE VALIDATION

Previous versions of the code that contained the main outputs such as stream thrust, mass capture, stagnation quantities, etc. have been in use for a few years and have been validated through comparisons to other codes when the same output variables existed for both codes. These previous versions have also been validated through their successful use in analyzing many previous problems. The current version of the code was also compared against an undocumented code by Ferlemann that had some of the capabilities of the current code and by far allowed the most comprehensive validation of parameters than any other code.

As new versions, such as 8.4, are released, the primary method of validation is to ensure that any modifications or code additions do not adversely affect any outputs known to be correct. This is done through the use of "golden files". A directory of test cases in the main directory uses all three types of inputs (1D, 2D, 3D) with varying input deck flags. For example, some cases have species while others do not, some input temperature and some use the equation of state temperature option, some request english output and other use metric, etc. By exercising nearly all of the codes options, errors can be quickly found by comparing the new version's output files (nearly 50 files) against every one of their golden files. If no changes are expected but are seen, the code is modified till the errors are corrected. If a new feature is introduced that should change the golden files, they are updated accordingly. While using this golden file approach does not ensure no errors exist or are introduced, it greatly cuts down on their potential.

SUMMARY

A code has been written that intends to satisfy all 1D post processing needs for the group. This code can be used instead of several fragmented tools used previously. Further, improvements to previous codes, such as automatic determination of the number of blocks, sizes of blocks, and block topology were made through the use of scripts that will greatly reduce the time required for the user to set up an input deck.

A flow chart is provided for each major section of the code. The methods and equations used for every section of the code have been laid out in great detail so that users, or recipients of the code's results obtained from users, can understand the origin of all parameters. An exhaustive user manual detailing setting up, modifying, and running the code is provided on the following pages. Each term in the input deck is examined providing the user a list of acceptable inputs as well as the meaning of each. Finally, a list of all file outputs as well as every variable output in each file is provided.

ACKNOWLEDGMENTS

Thanks to Paul Ferlemann and Tom Jentink for running previous versions (to help flush out bugs) as well as contributing ideas for improvements to the current version.

REFERENCES

[1]McBride, B.J., Gordon, S., and Reno, M.A., "Coefficients for Calculating Thermodynamic and Transport Properties of Individual Species," NASA TM-4513, October 1993.

[2]Ferlemann, P., "Description of SHIP 3.13 Output Files." HNAG Report 98-098 (NAS1-96013), NASA Langley Research Center, March, 1998.

[3]White, F., "Viscous Fluid Flow," McGraw-Hill, Boston, MA, 1991.

[4]Beyer, W.H., “CRC Standard Mathematical Tables,” CRC Press Inc., Boca Raton, FL, 1981.

[5]Rogers, R.C., Weidner, E.H., and Bittner, R.D., “Quantification of Scramjet Mixing in the Hypervelocity Flow of a Pulse Facility,” AIAA Paper 94-2518, June 1994.

[6]Ferlemann, P.G., “A Solution Methodology and Computer Program to Efficiently Model Thermodynamic and Transport Coefficients of Mixtures,” 37th JANNAF CS/PSHS/APS Joint Meetings, November 2000.

[7]Vitt, P., Ferlemann, P.G., and Bittner, R.D., “Distortion Database and Analysis for the Scramjet Combustor Configurations Tested Under GWP 54 (U),” HNAG Report 95-1-056, January 1996.

[8]Shames, I. H., “Mechanics of Fluids,” McGraw-Hill, New York, NY, 1982.

[9]Witte, D., “Computer Code for Determination of Thermally Perfect Gas Properties,” NASA Technical Paper 3447, September, 1994.

[10]Waltrup, P.J., Billig, F.S., and Stockbridge, R.D., “A Procedure for Optimizing the Design of Scramjet Engines,” Journal of Spacecraft and Rockets, Vol. 16, No. 3, May/June 1979, pp. 163-171.

[11]Anderson, J. D., “Introduction to Flight,” McGraw-Hill, New York, NY, 1989.

[12]Riggins, D.W., McClinton, C.R., and Vitt, P.H., “Thrust Losses in Hypersonic Engines Part 1: Methodology,” Journal of Propulsion and Power, Vol. 13, No. 2, March/April 1997, pp. 281-287.

[13]Anderson, G.Y., McClinton, C.R., and Weidner, J.P., "Chapter V - Scramjet Performance," Scramjet Propulsion, edited by E.T. Curran and S.N.B. Murthy, Vol. 189, Progress in Astronautics and Aeronautics, AIAA, 2001.

MASSFLOW3D v8.4 USER MANUAL

GETTING STARTED

The user will get a gzipped tar file called *massflow3d.tar.gz*. This can be opened with the command "tar -xvzf *massflow3d.tar.gz*". This will create a directory called massflow3d. The directory contains the following files:

README

miscellaneous minor information

RELEASE_NOTES

List of changes since last release.

TN-06-474

This document.

make

This is not a true make file but it has the same function in that it compiles the code automatically for the user. However, the script provides more options than are available in a true make file.

massflow3d

The main script. This script serves as a wrapper for the main massflow3d_prog.f FORTRAN file and allows many functions that are difficult if not impossible to do with FORTRAN commands such as system file management.

massflow3d.m3d

Sample input deck.

massflow3d_prog

linux compiled binary of main program massflow3d_prog.f

massflow3d_prog.f

FORTRAN 90 style main program

oldversions

Directory of previous versions

speciedata.ship

File with 49 species in SHIP format (file from composite[1] development).

testcases

Directory of test cases used for development. When version changes are made (minor or major) all the test cases are run and each case's output files are compared to "golden" files (output files from previous version) to ensure no unexpected changes occur. This reduces the chance that a new version will introduce unexpected errors. There are currently nearly 50 golden files. Each test case also provides the user with more input decks to examine for guidance in setting up problems. This directory makes up over 90% of the package size. It can be safely removed by the user if they wish. However, the checkcases flag in the *massflow3d* script will have to be set to 0.

SETTINGS

A few user specific settings and preferences must be set by the user upon initial receipt of the code. These settings only need to be changed once when setting up the code. In massflow3d there is a section at the top called "USER SHOULD SET THESE". The workdir variable should be set to the directory where massflow3d is located.

For the following options 0=off and 1=on. If the command "preplot" executes preplot on the user's system, then the autopreplot option can be turned on (default off) to automatically preplot all output Tecplot[2] dat files. Turning on the rmdatfiles option (default off) automatically removes all successfully preplotted dat files leaving only the plt file. Note this option only works if the autopreplot option is also on. If the dat file is not successfully preplotted, it is not removed. Turning on the debugclean option (default on) removes all debug files if the code is run with debug off, i.e. if debug is not on and the code is run, any debug files from previous runs will be removed. This provides a way to automatically clean the directory of debug files when debug is no longer needed. It is recommended that this option be left on unless the user wishes to keep the debug files.

---

1. Ferlemann, P.G., "A Solution Methodology and Computer Program to Efficiently Model Thermodynamic and Transport Coefficients of Mixtures," 37th JANNAF CS/PSHS/APS Joint Meetings, November 2000.
2. Amtec, TECPLOT v10.0, www.tecplot.com, 2006.

RUNNING MAKE FILE

If the precompiled binary does not work for your system or you wish to make changes to the code's logic or settings you need to compile a new binary. Note that altering the settings discussed in the "settings" section above do not require a recompile because those are in a script file. Script files do not get compiled, they are simply interpreted at run time. If the compiled version works on your system or you do not wish to make any changes you may skip this section.

After making any desired changes to the code, edit the *make* file. The checkcases flag (default off) should be turned off unless the user desires comparisons to the test cases; although, this is primarily used for development. The f90checking flag (default off) is also used for development and should not be used since run times are longer when error checking is used. The compile flag (default on) should be on to compile the code. The compiler flag sets the compiler used to compile the code. The default setting is "pg" for the Portland Group compiler available on the local clusters. If this compiler is not available, a free compiler such as g95 can be obtained and used. If there is a default compiler on the system that works with the command "f90" then the compiler option "default" may be used. However, this provides no flags for optimization or forced double precision since the compiler is unknown and therefore so are its options. Its use is only recommended as a last resort.

Once these values have been set the user can simply type "make" on the command line and the make script will report if compilation was successful or not. The user will need to add the path to massflow3d to their start up shell or set up an alias so that the code can be executed from any directory.

If the user alters the code and wishes to compare the output to the golden files for validation, the compiler and checkcases flag must both be set to 1. It is also recommended the f90checking also be set to 1, but this is not a requirement. The user then simply executes "make" from the command line in the massflow3d directory. This checks all test cases and displays all differences using xdiff (note, this assumes xdiff is on the user's system, otherwise, the user should modify the make script to use diff. If the user does not have xdiff, it is highly recommended due to its superior file comparing compared to the system's diff command). If the golden files are slightly different, such as machine round off differences, or expected differences appear due to the code modifications, the golden files can be replaced by the current output files by simply adding a 1 as a command line option to the make argument, thus, the user would execute: make 1. The user will be prompted to answer yes or no to overwrite the golden files. Upon answering yes, the code will be run again for all test cases and all golden files will be replaced by the current output files.

```
massflow3d inputdeck for version 8.X
-----------------------------------------------------------------------
data-file        i,j,k skip
inputfile.dat    1 1 1
-----------------------------------------------------------------------
x/i, text-loc, text-amf, 1DvsX-amf, total/pitot, reac, wall, t/p-V-vec
i       1           001        001       0        0     0       n
-----------------------------------------------------------------------
Variable numbers,  keep<--|-->optional
 X  Y  Z rho U  V  W  P  T H  H2 H2O N2 O  OH O2
 1  2  3  4  5  6  7  8  9
-----------------------------------------------------------------------
Species: mass/mole fraction for spec read in (dummy if nspec=0)
mass
-----------------------------------------------------------------------
Multiplying factor to convert x, y, and z to meters
0.0254
-----------------------------------------------------------------------
Multiplying width for 2-D problems (same units as input file units)
1.0
-----------------------------------------------------------------------
P(Pa) for gauge pressure in momentum forces and moments
(use -1 to use free stream value near bottom)
-1
-----------------------------------------------------------------------
Coordinates to take moments about (meters)
0.0 0.0 0.0
-----------------------------------------------------------------------
units for all output files (0=m=M for Metric,e=E=1 for English)
0
-----------------------------------------------------------------------
species file (SHIP format) or can use air, pgas, pgasair, or all
all
-----------------------------------------------------------------------
MW (kg/kmol) and Cp (J/kg-K) (only used if above species is 'pgas')
28.96518        1008
-----------------------------------------------------------------------
Free stream values for eta_ke (make pressure <=0 to skip etake)
P(Pa)     Temp(K)      Vel (m/s)  Mfractions
 0.0         0.0          0.0
-----------------------------------------------------------------------
TP: Aexit(m^2)-Pinf(Pa), Cs,  Cfg (A: 0=off, P: -1=use Pinf 1 line up)
     0.0       -1        0.98   0
-----------------------------------------------------------------------
```

Figure 7. Sample input deck provided.

INPUT DECK

Figure 7 shows the sample input deck provided in the massflow3d directory. Each section will be examined sequentially by showing the acceptable inputs and explaining each option. The first line of the input deck tells the code which version it expects to be run with. This line should never be altered. Changing this line will cause the code to not run since it will believe that the input deck and code are incompatible. This line is used so that when new versions of the code are released that require an input deck upgrade, old input decks are not accidentally used that could give erroneous results. The input deck is free format so inputs may be spaced out however the user wishes. However, the recommended set up is to set values under their titles to reduce confusion. The inputs follow:

Input name	Acceptable inputs
data-file	any file name

The data file must be an ASCII column formatted text file. Typically, a Tecplot point format is used. More than one block may be input. The words zone and block are used interchangeably in this document. While the file need not be an actual Tecplot file, it must have at least one line before each zone which states the I, J, and K sizes of each zone (even if they are 1). The K must also have a "," after it so that the massflow3d script can automatically determine the block sizes. Note that this is the default output style of Tecplot v10 or higher, so no modifications will be necessary in the majority of cases. Thus, the minimum information that could be provided on a line before a zone is (Note the "," after the K statement):

I=1,J=2,K=3,

Where the values of I, J, and K are dictated by the block size. Any other lines may also be present but they are unused and treated as dummy lines.

While the values of J and K may be either direction, the I and x directions must both be in the axial direction. If the problem is 2D then K must =1. If either of these conditions is not the case, there is an existing code (written by the author) that can change the directions of any of the dimensions to the users desired orientation. The u, v, and w velocity vectors should also correspond to x, y, and z; however, if this is not the case only a simple input deck change is needed (discussed in the input deck section: variable numbers). If u, v, and w and x, y, and z are not aligned as required and the input deck change is not done, then the output of the momentums and forces will be reported in the incorrect directions. The one dimensionalizations, such as pressure, temperature, etc. will still be done correctly.

The rho, u, v, w, P, and T variables discussed later must be in metric units of (kg/m^3, m/s, Pa, and K). the x, y, and z variables can be in any units. See the variable numbers input section for more details.

Input name	Acceptable inputs
i,j,k skip	integer >=1 (3 inputs with spaces between)

The skip factors allow the I, J, and K indices to be independently skipped. Due to the large memory requirements that some outputs can have since they are output at every point that is input (as is possible with the pitot/total calculations, for example, discussed later) the code can attempt to allocate too much memory. Using skipping reduces the input by advancing the I, J, or K read ahead by the input integer specified by the user. Thus, a skipping of 2 for I would read in grid points I=1, 3, 5, 7, etc. while a value of 3 would read in I=1, 4, 7, 10, etc. Note that even small integers such as 2 can greatly reduce the amount of memory. If all skipping factors are set to 2 the required memory for reading the data file drops by 8. If such a large drop is not necessary and grid quality nearer the original data file is desired the user could set just I=2 and leave J=K=1, or alternatively leave I=1 and set J=K=2 if axial grid quality is more important. The user should keep in mind that if the solution was discretized for parallel processing (and is still in that topology) that the dimensions of some of the blocks may be small already and further reduction could lead to a dramatically different flowfield being input to the code. If a global skipping is not sufficient to meet the user's needs, a code such as Tecplot should be used to reduce the dimensions of only those blocks the user wishes to reduce.

Since this code deals with contained (bounded) flows, ensuring that the cross sectional area remains intact is important. Thus, if the user inputs an integer that would end the read of any dimension prior to the end of that dimension's max dimension, the max dimension is read in anyway. For example, in Fig. 1 it is clear that the dimension of the box is J=5, K=5. If the user inputs J skipping of 3 then J=1, 4, and 5 would be read in where the 5 is forced to ensure the outer boundary is used. If the user inputs such a large value that only the minimum and maximum boundaries are used, a warning message is output. Again, this could occur if a discretized grid is used which contains one or more small zones.

Input name	Acceptable inputs
x/i	x or i

The x/i input tells the code whether certain inputs that follow on the same line describe x locations or i plane locations. These inputs are discussed in the sections below.

Input name	Acceptable inputs
text-loc	IF x/i = i THEN integer >=0 IF x/i i = x THEN any real number

The text-loc input determines the location of the text file output (the output file is detailed in the output files sections). A value of 0 turns off all text file outputs (area weighted, mass weighted, and flux conserved) for either value of x/i. If x/i = x then the axial location used for the text file output is the plane nearest the input value. The value input should be the same units as the data file's grid units. This applies for all x values on this input deck line. If the user desires a value of x = 0, 0 can not be input because this would turn off the outputs, rather, a very small value such as 1E-10 should be used. The closest plane at x = 0 will then be used. If x/i = i then the plane output is simply that of the integer input. Although, if there are multiple blocks with overlapping planes, then the situation is a bit more complex. Figure 8 shows a 2D problem with three zones. Note that one of the zones runs the full length of the domain, while the other two do not. Since the first plane of the top right zone starts at the same location that the last plane of the top left zone ends, there is an overlap of one plane. This can be further complicated if there are multiple blocks connected and/or overlapping. Thus, the concept of the "absolute plane" is introduced. The absolute plane removes the overlapping plane in its accounting. Thus, for Fig. 8's topology, the bottom zone's planes correlate one to one with the absolute plane number. However, if the user desires the information in the absolute plane made up partly by the top right zone's plane 8, they would have to note that this is absolute plane number 18, NOT a plane number determined by adding the dimension of the top left (11) and the desired plane (8) which would equal 19. While determining the absolute plane is fairly simple in most cases, complex topologies could result in the wrong output plane being output. If the user is unable to determine the absolute plane number of the plane they wish to output, using the x option of the x/i input may be a simpler alternative.

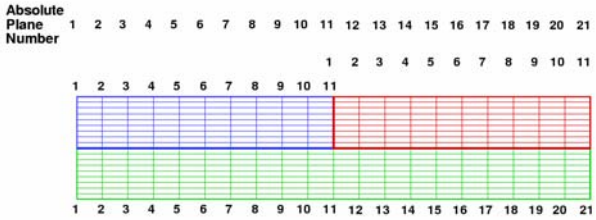

Figure 8. Absolute plane terminology

Input name	Acceptable inputs
text-amf	000, 001, 010, 011, 100, 101, 110, or 111

The text-amf flag is a bitwise representation of on/off bits (0=off, 1=on) for output of the area weighted (a), mass weighted (m), and flux conserved (f) text files. The input should be a single integer of 3 digits (no spaces). While the input can be anywhere on the line, placing it directly under the "amf" portion of the text-amf string clearly indicates which outputs are on and which are off. The axial location (or plane) at which these text files are output is given by the value of the text-loc input discussed in the previous section.

Input name	Acceptable inputs
1DvsX-amf	000, 001, 010, 011, 100, 101, 110, or 111

The 1DvsX-amf flag is a bitwise representation of on/off bits (0=off, 1=on) for output of the area weighted (a), mass weighted (m), and flux conserved (f) one dimensional quantities versus axial distance (X). The input should be a single integer of 3 digits (no spaces). While the input can be anywhere on the line, placing it directly under the "amf" portion of the 1DvsX-amf string clearly indicates which outputs are on and which are off. Since these output files automatically output the full axial range, this input is sufficient to determine the output of the 1DvsX files.

Input name	Acceptable inputs
total/pitot	"all" or any integer or real

The total/pitot input determines the axial location for the output of the stagnation quantities and pitot quantities for every node in an absolute plane. If the user specifies "all" then the total/pitot conditions are output at every point in the domain. This can easily become too memory intensive for large 3D problems since dozens of quantities are output for every node in the domain. If there is not enough system memory to use the "all" setting, then the user must either use skipping or output desired planes one at a time using the following other input options. If x/i = x, then any non zero value will output the total/pitot information at the nearest absolute plane. An input of 0 turns off the total/pitot output. If x/i = i, then any positive integer will output the total/pitot

information at the input absolute plane. Any integer <=0 turns off the total/pitot output.

Input name	Acceptable inputs
reac	“all” or any integer or real

The reac input determines the axial location for the output of quantities related to reaction solutions (detailed in the output files section later). Note that the reaction output file is only output if there is at least one H molecule present in the flowfield. The acceptable inputs and behavior is the same as the total/pitot input (previous section).

Input name	Acceptable inputs
wall	any real

The wall input determines the output for the wall information. Any value <=0 turns off the wall output. Any other value turns on the wall output and uses the input value as the maximum allowable velocity a node can have and still be considered a wall. This is used because the wall cells will not have velocity magnitudes of exactly zero due to numerical round off. A recommended input value is 1 if wall output is desired. This implies any cell which has an absolute velocity less than or equal to 1 is a wall. Only the minimum and maximum I, J, and K boundaries of each zone are checked since a wall can not be in the middle of the flowfield.

Using the debug feature (discussed in the “other features” section) can help diagnose problems with the wall determination because it outputs an additional file named “walls_dumped_by_debug.dat”. This file contains all the I, J, and K minimum and maximum boundaries. The area is output for each cell. Cells not considered walls by the program are assigned an area of zero. If the file is read into Tecplot and value blanking is used for all cells with areas equal to zero, then only the surfaces determined to be walls will remain. This provides a quick way to determine if the program is correctly finding the walls. The normal and surface unit vectors used for the inviscid and viscous forces, respectively, for each non zero area cell are also output and can be viewed with the velocity vector feature of Tecplot.

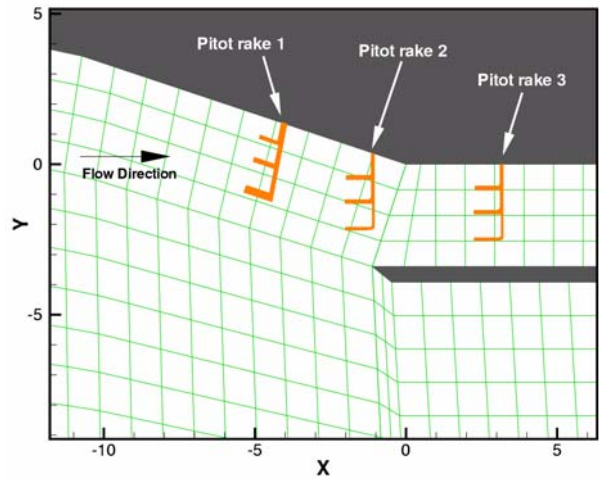

Figure 9. total/pitot velocity vector input description

Input name	Acceptable inputs
t/p-V-vec	u,v,w,uv,uw,vw,uvw,n

The t/p-V-vec input stands for total/pitot-Velocity-vector. It determines what velocity vector should be used for the pitot calculations output in the total/pitot output file. Figure 9 shows how this feature is useful. A fictitious 2D forebody and grid are shown. If a pitot rake were mounted as shown for pitot rake 1, then the user should input a t/p-V-vec = n where n stands for normal to make a valid comparison to data obtained from this rake. The “n” input uses the velocity vector normal to the I cell face. For pitot rake 2, the normal vector of the cell does not align with the pitot rake. In this case, a t/p-V-vec = u should be used so that only the u component of the velocity is used for the total/pitot calculations. Finally for pitot rake 3, an input of u or n would produce the same result since the cell area normal vector aligns with the u (i.e. x) direction. Note that only the pitot pressures use this user specified velocity. Terms such total pressure and total temperature still use the actual velocity made up of all velocity components.

Input name	Acceptable inputs
X,Y,rho,U,V,P	integer >0
Z,W,T	integer >=0
H,H2,H2O, etc.	integer >=0, 99, or no input

The variable number inputs tell the program the columns in which each input (X, Y, Z, etc.) is located in

the data file. Using this type of input allows the user to have the variables in any order in the data file without the need to rearrange them to satisfy an input requirement. It also allows for extra variables, such as Mach number, to be present in the data file and still allow the program to run. If for example there was a Mach number variable in the data file in column 9, but the rest of the variables were the same as shown in the input deck in Fig. 7 (except the variables after 9 were shifted one column to the right due to the presence of the M variable), then the input line would appear as follows:

```
-------------------------------------
Variable numbers,keep<--|--> optional
X Y Z rho U V W P T      H H2 H2O N2 O OH O2
1 2 3  4  5 6 7 8 10
-------------------------------------
```

Note that there is no 9 present because it is Mach number and the program does not request it to be read in.

The variables X, Y, rho (density), U, V, and P all require integers >0 providing the location of those variables in the input deck. For 3D problems Z and W require the same. However, for 2D problems, if Z and W are not present in the input deck, or the user does not wish to read them in, one or both values can be set to 0. Any variable set to 0 is not read in.

The rho, u, v, w, P, and T (temperature) variables must be in metric units of (kg/m^3, m/s, Pa, and K). The x, y, and z variables can be in any units since there is a multiplying factor further down in the input deck to convert the grid to meters.

If the u, v, and w variables do not correspond to the x, y, and z directions as required to correctly output the momentums and moments (as stated previously) then the user need only trick the program into reading v as w and w as v by switching their variable numbers in the input.

T can also be set to zero if it is not present in the data file or the user does not wish to read it in. In this case, the equation of state (P=rho*R*T) is used to calculate T at every node.

The species names and numbers listed at the end of the line need only be present if species mass or mole fractions are being read in. There is no required order or number of species that need to be listed. In Fig. 7, the species names are shown but there are no numbers listed, thus, the species are not used. The species names could have been removed entirely (indicated by the "-->optional" arrow on the line above the species names) or the values could have been set to zero for each species. Thus, the following lines are all equivalent ways of telling the program there are no species present.

```
-------------------------------------
Variable numbers,keep<--|--> optional
X Y Z rho U V W P T      H H2 H2O N2 O OH O2
1 2 3  4  5 6 7 8 9
-------------------------------------

-------------------------------------
Variable numbers,keep<--|--> optional
X Y Z rho U V W P T      H H2 H2O N2 O OH O2
1 2 3  4  5 6 7 8 9      0 0  0   0  0 0  0
-------------------------------------

-------------------------------------
Variable numbers,keep<--|--> optional
X Y Z rho U V W P T
1 2 3  4  5 6 7 8 9
-------------------------------------
```

The species names and numbers should be changed as needed for each problem. Note that any species indicated must be present with <u>exactly</u> the same name as specified in the species file discussed later. Care should be taken when entering species with an O that a zero is not typed instead. The program will exit and tell the user why if it can not match an entered species with any in the species file. A sample variable number section is shown below for species input that include N2, H2, and O2. Note that the other species have been removed and the new species names and number have been input. Also note that as with the "keep" inputs the order is irrelevant.

```
-------------------------------------
Variable numbers,keep<--|--> optional
X Y Z rho U V W P T      H2 N2 O2
1 2 3  4  5 6 7 8 9      12 10 11
-------------------------------------
```

If a species number is input as 99, then the program calculates the species fraction by summing the rest of the species and subtracting them from 1, i.e. it forces the sum of the species fractions to be 1. Note that only one species can be assigned a value of 99 since the program can not uniquely determine the mass fraction of multiple species by subtracting more than one species fraction from 1. The 99 input has 2 uses. First, it can be used when all but one species are available in the data file. For example, if in the above sample variable number line N2 is not output from the CFD program in order to reduce the data file size, the input would appear as follows:

```
-------------------------------------
Variable numbers,keep<--|--> optional
X Y Z rho U V W P T      H2 N2 O2
1 2 3  4  5 6 7 8 9      12 99 11
-------------------------------------
```

The input species must always sum to 1 to within the programs tolerance (default tolerance of 0.01). If they do not, the program will exit. The user can get around this if the sum is, for their use, close enough to one by adding an extra species (such as "Air" or some other benign species appropriate to the problem) and setting its input number to 99 if species 99 has not been used. If species 99 has already been used, the user will have to read in the data file into a program such as Tecplot and manually generate an extra species fraction. This should not be necessary for the majority of problems since the species should always sum to nearly one and the 0.01 tolerance is fairly liberal.

The second use of the species number 99 feature is for problems where a species fraction is not output because there is only one species. The use of 99 is not necessary if the species is air, a perfect gas, or is the first species present in the species data file (discussed later). However, suppose a flowfield is made up of entirely N2. The user can simply list N2 as the only species and set its number to 99. Its mass fraction then becomes one because there are no other species present in the input deck (or if they are present their values can be set to zero or they can be removed). If the user forgets to tell the program to use N2, then the first species listed in the species data file will be used (which is air for the default file).

Input name	Acceptable inputs
Species: mass/mole fractions	"mass" or "mole"

The mass/mole input tells the program if the species fraction inputs discussed in the previous section are mass or mole fractions. Any other inputs will cause the program to exit.

Input name	Acceptable inputs
Multiplying factor	any real

The multiplying factor input converts the user's input grid to meters. The input value is multiplied by the x, y, and z value of the data file and should result in units of meters. If the grid is already in meters a value of 1.0 should be input. If, for example, the user's grid is in inches (as is common) then 0.0254 should be entered.

Input name	Acceptable inputs
Multiplying width	any real

The multiplying width input is only used if K=1 for all zones, i.e. the input is a 2D problem. The width should be input in the same units as the input grid. It will be multiplied by the multiplying factor to convert it to meters. Thus, if the input units were inches, but a width of one meter is desired, the multiplying width should be input as 39.37.

Input name	Acceptable inputs
P(Pa) for gauge pressure in momentum forces and moments	any real

The gauge pressure input is used for the calculation of the stream thrust (momentum) and moment calculations. The term is the P_∞ in the generic stream thrust equation. A value >=0 uses the input value as P_∞, while a value less than zero (-1 suggested for consistency) uses the value of P_∞ near the bottom of the input deck on the freestream η_{KE} line. This allows the user to only enter P_∞ once in the input deck to reduce the chance or error in inputting the value.

Input name	Acceptable inputs
Coordinates to take moments about (meters)	any real (3 values with spaces between)

The coordinates input is the x, y, and z centers of gravity for the moment calculations. Any values can be input but all three value must be present. The inputs must also be in meters. Note this is different from the multiplying width where the units of the input grid's units were required. The input deck clearly states meters are requested so the user is not confused. If the center of gravities are not known in meters, the user will have to convert the existing units prior to entering them.

Input name	Acceptable inputs
units for all output files	e,E,m,M,0 or 1

The units input determines if all output files will be in english or metric units. As stated in the input deck, 0 = m = M for metric and 1 = e = E for english. All output

file parameters' units are clearly labeled so there is no confusion of the units output.

Input name	Acceptable inputs
species file	air, pgas, pgasair, all or a valid file name

The species file input tells the program where to find the species information for the current case. The program must use a species to find the fluid's thermodynamic quantities even if no species are being read in. The only valid non-file name inputs are "air", "pgasair", "pgas", and "all". If air is input, an internal thermally perfect air species (previously generated by composite) is used and no species file is necessary. If pgasair is input, an internal calorically perfect air species is used and no species file is necessary. If pgas is input, a calorically perfect gas is used with properties given by the inputs of the next input deck line. Any other input (except "all") is assume to be a valid filename with one or more species listed in the SHIP format. The SHIP format is described in reference 2 of the methodology section.

If "all" is specified the provided species file "speciedata.ship", from the composite program development, providing the following 49 species (in order) is used: Air, N2, O2, H2, H2O, O, H, OH, Ar, C, CH, CH3OCH3, CH3OH, CH4, CN, CO, CO2, C2, C2H2, C2H4, C2H5OH, C2H6, C2N2, C3H6, C3H8, C3H8O, C4H10, C4H10, C6H6, HCN, HCO, HNO, HNO2, HO2, H2O2, He, N, NH, NH3, NO, NO2, N2O, N2O4, O3, Si, SiH4, SiO, SiO2, and Si2.

If the user wishes to use a species not listed above, for example a new composite species, they can either add the species to the end of the speciedata.ship file with a unique name and then use that name in the variable names input line, or they can put the new species in a file in the same directory as the input deck and enter that file name as the species file input. If no species fractions are input, then the program will use the first species listed in the species file (which may be the only species in this case). The use of composite is suggested if the user is making new species since it automatically outputs species in the SHIP format.

Input name	Acceptable inputs
MW (kg/kmol) and Cp (J/kg-K)	any real

The MW and C_p inputs are only used if the species file input is specified as "pgas". If this is the case, then the MW and C_p values are used to determine the calorically perfect thermodynamic quantities. Note the value must be input in the units specified by the input deck.

Input name	Acceptable inputs
Free stream values for eta_ke P, T, Vel, Mfractions	any real

The free stream values for eta_{KE} inputs are used to determine the free stream enthalpy and other parameters required to find the kinetic energy efficiency (defined in the methodology section). As the input deck states, the inputs are pressure in Pa, temperature in K, velocity magnitude in m/s, and the mass or mole fractions (whatever was specified earlier in the input deck) of each free stream species. If the P term is set to <=0 then the eta_{KE} calculations are not done. In this case the T, Vel, and Mfractions terms are dummy inputs. If the number of species listed in the variable numbers line is zero, then the Mfractions need not be listed. However, if the number of species is >=1, then the mass fractions of the free stream species must be listed in the same order as they are listed in the species variable number line for all non zero species. This includes any species numbered 99.

Input name	Acceptable inputs
TP: Aexit, Pinf, Cs, Cfg	any real

The TP (Thrust Potential) inputs are used to obtain thrust potential and nozzle efficiency parameters. The parameters are exit area, free stream pressure, C_S, and C_{FG}. If A_{exit} is set to 0, then none of the TP or nozzle efficiency calculations are done and the rest of the values are dummy inputs. Otherwise, an exit area in m^2 must be input. Pinf, input in Pa, determines the free stream pressure for use in the $(P-P_\infty)*A_{exit}$ calculation for the ideal expansion to the input area, as well as the free stream pressure to expand to for the ideal gross thrust. C_S and C_{FG} are nozzle coefficient inputs. One and only one must be provided. The other is calculated, since they are dependently related, from the input coefficient. All of the TP equations and terms are explained in detail in the methodology section of this document.

OTHER FEATURES

While the input deck has the majority of the input options, there are a few other features which are used less often but can be useful.

If an extra line is added to the bottom of the input deck with only the word “debug” (no quotes) the code enters debug mode. This can help a user diagnose any problems they may be having. First, the internal input deck (which is created from the user input deck) is not removed upon run completion and is echoed to the screen. Second, an extra file called “debug_output.txt” is created in the directory where more informative messages are printed that are not printed to the screen. The most useful of this extra information is probably the automatic plane line up information which is output when more than one block is read in. This tells the user the program determined automatic plane line up and can provide useful information if the 1D output does not appear as expected. For the case shown earlier in Fig. 8, the plane line up would appear as follows:

```
========================
Block and planes line up
========================
apl|  1|  2|
------------
  1|  1|   |
  2|  2|   |
  3|  3|   |
  4|  4|   |
  5|  5|   |
  6|  6|   |
  7|  7|   |
  8|  8|   |
  9|  9|   |
 10| 10|   |
 11| 11|   |
 12|   |  2|
 13|   |  3|
 14|   |  4|
 15|   |  5|
 16|   |  6|
 17|   |  7|
 18|   |  8|
 19|   |  9|
 20|   | 10|
 21|   | 11|
```

The first column shows the absolute plane number. The number of columns after the first is determined by the number of blocks. Since there are two blocks for this case, there are two additional columns, representing each block, which are numbered at the top. Each of the rows for these two columns then shows where each I plane of that block falls relative to the absolute plane. Note that for block 2, there is no plane 1 listed. Since the program works at the nodes (and not cell centers), including the final plane of block 1 as well as the first plane of block 2 would double book keep these nodes since they are overlapping (as shown in Fig. 8). While the default is to remove the plane of the overlapping block which has the I=1 plane, this can be changed in the massflow3d_prog.f file so that the maximum plane is removed and the first plane is instead used for the one dimensionalization. This flag is called “overlapremove” and is currently set to “min” in the massflow3d_prog.f main program file. The value may alternatively be set to “max”. If this is done then the absolute plane 11 line from the above example would appear as follows:

```
11|   |  1|
```

while all other lines would remain the same. Note that since this flag is in the FORTRAN code, changing it requires a recompile. There should be no reason to need this change but the option is provided nonetheless.

The final feature of using debug is the extra wall output information. This is discussed in detail in the wall input deck section.

The other non input deck option is to override the automatic plane line up that the program uses. This can be useful in the case where the program incorrectly lines up the planes. While this should not happen, providing this option allows a work around to be used until the source of the program’s difficulties can be fixed. If any user has a case where the automatic plane line up is not working, they should notify the author so the problem can be fixed. The override is done by simply putting a file in the directory called “opl” (override plane lineup). This file has a line for each block which specifies the block number followed by the starting absolute plane for the first plane of each block. Each subsequent plane of each block is assumed to increase one for one with the absolute plane. For the example above, the opl file would be as follows:

```
1     1
2    11
```

The file must have the same number of lines as there are blocks or it will not be used. In this case the user will be notified that an opl file is present but it is not being used. Note that blank lines count as lines. The user can use this to their advantage if they wish to temporarily disable the opl file by simply inserting a dummy line anywhere in the file.

If debug is on, the debug_output.txt file will have the plane line up in opl format as well as the output shown previously. This allows the user to quickly generate an

opl file for large problems that are repeatedly run, allowing much faster run times since the program does not have to determine the plane line up every time it is executed.

If the user uses skipping in the I plane then the opl file must take this into account. The same opl file can not be used for input decks where I skipping is not the same. The values of the J and K skipping are irrelevant.

Blocks in the input file may be excluded from the calculations by adding a file named "massflow3d_zonesremove" to the current directory. This file should simply list the block numbers to be excluded with one block on each line.

RUNNING THE CODE

Assuming the user has created a path or an alias for the *massflow3d* script file, the program is run by simply typing:

massflow3d inputdeck.m3d

While the input deck is not required to have a m3d extension, doing so has some advantages. First, it is clear that the file is an input deck for massflow3d. Secondly, if there is only one m3d file in the working directory, then the user need not type its name, the program will automatically find the only m3d file and assume it is the input deck. In this case, the program can be run by simply typing:

massflow3d

In this case if more than one m3d file is present the user will be notified of such, presented with a list of all m3d files, and prompted to enter the input deck they wish to run. The script can be killed with ctrl-C if the user does not wish to run any of the input decks.

OUTPUT FILES AND DESCRIPTIONS

The following output file descriptions simply list each file and the output. While a brief description is provided, detailed explanations of output variables and how they were obtained is done in the methodology section. All output files, except the text files, are output in Tecplot format. The units are also clearly labeled in every variable name and are dependent on the user's english or metric preference so they will not be reproduced here. Rather, the goal of this section is to identify which variables are output in which files.

The names of the output files are determined based on the input data file. The output files are named based on the base name of the data file up to the final ".". The following file names assume the data file name was file.dat

TEXT FILES

The variable outputs of the text file outputs are identical for the area weighted, mass weighted, and flux conserved approaches. The text file outputs an easily readable list of one dimensional parameters at a given plane. The text output file names are file.text.aw, file.text.mw, and file.text.fc where aw, mw, and fc represent the area weighted, mass weighted, and flux conserved approaches, respectively. Since the outputs are descriptive, explanations are provided for a few outputs following the table where the output is not standard. Table 1 shows the text file outputs. Note that the area and mass weighted moments are not the true plane's moments because they were found by applying the 1D results over the entire plane, and then summing the results. More details concerning this can be found in the code methodology section "Find area weighted, mass weighted, and flux conserved outputs".

Table 1. file.text.XX output

Average x value of plane
Absolute plane number
Plane unit vector x
Plane unit vector y
Plane unit vector z
Mass flow rate
Stream thrust in x dir
Stream thrust in y dir
Stream thrust in z dir
Ambient pressure input
Total stream thrust
Total area
Moment about user x axis
Moment about user y axis
Moment about user z axis
Total Energy Flux (b0)
Total Enthalpy (b0)

Table 1. file.text.XX output

Static Enthalpy (b0)
Total Energy Flux (b298K)
Total Enthalpy (b298K)
Static Enthalpy (b298K)
Pressure
Total Pressure (c.p.)
Total Pressure (t.p.)
Temperature
Total Temp (c.p.)
Total Temp (t.p.)
Density
Total Density (c.p.)
Total Density (t.p.)
u
v
w
Velocity
Mach (using u)
Mach (using v)
Mach (using w)
Mach (using Vel)
Re/x
Laminar viscosity
Thermal conductivity
gamma
Molecular Weight
R (Gas Constant)
C_p
C_v
eta_KE
eta_KE adiabatic
+Entropy from plane 1
Pt recovery (to pl 1,cp)
Pt recovery (to pl 1,tp)
Pt recovery (to fs,cp)
Pt recovery (to fs,tp)
Distortion, eta_a
Distortion, eta_f
Distortion, eta_dv
Mass fraction X

Table 1. file.text.XX output

Mole fraction X
Mixing efficiency
Reaction efficiency
Combustion efficiency
Fuel equivalence ratio
Thrust potential (ref A)
Thrust potential (ref P)
Exit A req for Pref TP

In the text file output eta_ke is the kinetic energy efficiency, gamma is the ratio of specific heats, b(0) and b(298K) are the values using a reference (base) temperature of 0 or 298 K, and c.p. and t.p. are calorically and thermally perfect, respectively. Note that not all these variables may be output depending on the characteristics of the flow field. For example, if a free stream pressure >0 is not input, the etake variables nor the total pressure recovery to free stream can be found and are not output.

PLANES FILES

As with the text files, the planes files have the same variables output for the aw, mw, and fc outputs. These files plot various one dimensional parameters given for each absolute plane and its corresponding average x value. The planes output file names are file.planes.aw.dat, file.planes.mw.dat, and file.planes.fc.dat. The output is shown in Table 2. Note that the area and mass weighted moments are not the true plane's moments because they were found by applying the 1D results over the entire plane, and then summing the results. More details concerning this can be found in the code methodology section "Find area weighted, mass weighted, and flux conserved outputs".

Table 2. file.planes.XX.dat output

x	Average x value at absolute plane
i	Absolute plane
mdot	Mass flow rate
streamthx	Stream thrust in x dir
streamthy	Stream thrust in y dir
streamthz	Stream thrust in z dir
streamthtot	Total stream thrust
area	Area of plane
Momentx	Moment about user x axis

Table 2. file.planes.XX.dat output

Momenty	Moment about user y axis
Momentz	Moment about user z axis
E	Energy
ht	Total enthalpy
hs	Static Enthalpy
P	Static pressure
Pt(cp)	Calorically perfect total pressure
Pt(tp)	Thermally perfect total pressure
T	Static temperature
Tt(cp)	Calorically perfect total temperature
Tt(tp)	Thermally perfect total temperature
rho	Static Density
rhot(cp)	Calorically perfect total density
rhot(tp)	Thermally perfect total density
u	u component of velocity
v	v component of velocity
w	w component of velocity
Vel	Velocity magnitude
M_u	Mach number using u
M_v	Mach number using v
M_w	Mach number using w
M	Mach number using Vel
Re/x	Unit Reynolds number
mu_l	Laminar viscosity
k	Fluid thermal conductivity
MW	Molecular Weight
R	Specific gas constant
C_p	Specific heat (constant pressure)
C_v	Specific heat (constant volume)
deltaentropy	Entropy increase from first plane
etake	Kinetic energy efficiency
etakea	Adiabatic kinetic energy efficiency

Table 2. file.planes.XX.dat output

dist-eta_a	Distortion parameter η_a
dist-eta_f	Distortion parameter η_f
dist-eta_d_v	Distortion parameter η_{dv}
Ptrfs(cp)	Total pressure recovery = 1D total pressure at plane over 1D total pressure at free stream, both calorically perfect
Ptrfs(tp)	Total pressure recovery = 1D total pressure at plane over 1D total pressure at free stream, both thermally perfect
Ptrp1(cp)	Total pressure recovery = 1D total pressure at plane over 1D total pressure at plane 1, both calorically perfect
Ptrp1(tp)	Total pressure recovery = 1D total pressure at plane over 1D total pressure at plane 1, both thermally perfect
Massfrac-X	Mass fraction of species X. (listed for every species)
Molefrac-X	Mole fraction of species X. (listed for every species)
eta_m	Mixing efficiency
eta_r	Reaction efficiency
eta_c	Combustion efficiency
phi	Fuel equivalence ratio
TP(ref-area)	Thrust potential for expansion to known exit area
TP(ref-P)	Thrust potential for expansion to free stream P
Exit-area-req-for -TP-refP	Exit area required for TP expansion to free stream P
C_S	Nozzle stream thrust coefficient
C_F_G	Nozzle gross thrust coefficient

In the planes file output, the enthalpies are only provided in the bases for which they are calculated. Thus, for a thermally perfect calculation where McBride curve fits are used, the enthalpies are provided at a 298K base, while for a perfect gas computation, the base is 0K.

TOTAL/PITOT FILE

The total/pitot output file is named file.pitot.dat. This file outputs variables at every node of the data file for the requested plane(s). The output is structured with multiple Tecplot zones with one zone for every plane of every block grouped in absolute plane order. Unlike the previous files, there is only one file since the variables output in this file are not dependent on the one dimensionalization method. The output variables are shown in Table 3.

Table 3. file.pitot.dat output

x	X location of node
y	Y location of node
z	Z location of node
P	Static pressure
Ptcp	Calorically perfect total pressure
Pttp	Thermally perfect total pressure
Pp93ru2	Pitot pressure found using empirical equation 0.93*rho*vel^2
Pprayl	Pitot pressure found with Rayleigh formula
T	Static temperature
Ttcp	Calorically perfect total temperature
Tttp	Thermally perfect total temperature
rho	Static density
rhotcp	Calorically perfect total density
rhottp	Thermally perfect total density
vel	Velocity magnitude
user-vel	User specified velocity for pitot calculations
M	Mach number using vel
user-M	Mach number using user-vel
Ptrp1(cp)	Total pressure recovery = 1D total pressure at node over 1D total pressure at plane 1, both calorically perfect
Ptrp1(tp)	Total pressure recovery = 1D total pressure at node over 1D total pressure at plane 1, both thermally perfect

Table 3. file.pitot.dat output

Ptrfs(cp)	Total pressure recovery = 1D total pressure at node over 1D total pressure at free stream, both calorically perfect
Ptrfs(tp)	Total pressure recovery = 1D total pressure at node over 1D total pressure at free stream, both thermally perfect

REACTION INFO FILE

The reacinfo file outputs quantities associated with combustion solutions. This file outputs variables at every node of the data file. The output is structured with multiple Tecplot zones with one zone for every plane of every block grouped in absolute plane order. The file is only created if there is at least one H molecule present in the data file. The output file is named file.reacinfo.dat and its contents are shown in Table 4.

Table 4. file.reacinfo.dat output

x	X location of node
y	Y location of node
z	Z location of node
local-phi	Fuel equivalence ratio of node
local-eta-c	Local combustion efficiency which measures the ratio of the nodes conversion of fuel to H2O to what was possible based on fuel levels
Massfrac-X	Mass fraction of species X (listed for all species)
Molefrac-X	Mole fraction of species X (listed for all species)

WALL FILE

The wall file outputs variables associated with any walls that are present in the data file. The determination of what is a wall is detailed in the wall section of the input deck discussion. As with the 1DvsX output files, the results are given for each absolute plane and its corresponding average x value. If a plane rather than a block is input (such that I=1) and a wall file is requested, an axial length is needed to calculated the area. In this case, the length used is the multiplying factor entered previously in the input deck. This implies that it is one

unit distance of the input grid units. For example, if the input grid units are inches, then the user must have specified 0.0254 for the multiplying factor. Thus, the length is 0.0254 meters (which is 1").

The perimeter calculation is the perimeter of the wetted surfaces of the absolute plane. Thus, if the plane is not y-z planar, then the true perimeter of the walls at a given x location will not be output because the slanted surfaces are longer than a straight line between opposite walls. Thus, care should be taken if using this parameter with a non planar grid.

The output variables for the wall file are shown in Table 5. The “running” connotation means that the variable has been summed to include all absolute planes up to, and including, the plane being plotted.

Table 5. file.walls.dat

x	Average x value at absolute plane
i	Absolute plane
perimeter	wetted perimeter of plane
wetted-area	Wetted area at absolute plane
rwetted-area	Running wetted area
F_x-inv	Inviscid force in the x direction
F_y-inv	Inviscid force in the y direction
F_z-inv	Inviscid force in the z direction
rF_x-inv	Running inviscid force in the x direction
rF_y-inv	Running inviscid force in the y direction
rF_z-inv	Running inviscid force in the z direction
F_x-vis	Viscous force in the x direction
F_y-vis	Viscous force in the y direction
F_z-vis	Viscous force in the z direction
rF_x-vis	Running viscous force in the x direction
rF_y-vis	Running viscous force in the y direction
rF_z-vis	Running viscous force in the z direction
F_x-tot	Total force in x direction
F_y-tot	Total force in y direction
F_z-tot	Total force in x direction

Table 5. file.walls.dat

rF_x-tot	Running total force in x direction
rF_y-tot	Running total force in y direction
rF_z-tot	Running total force in x direction
M_x-inv	Moment about user x axis due to inviscid forces
M_y-inv	Moment about user y axis due to inviscid forces
M_z-inv	Moment about user z axis due to inviscid forces
rM_x-inv	Running moment about user x axis due to inviscid forces
rM_y-inv	Running moment about user y axis due to inviscid forces
rM_z-inv	Running moment about user z axis due to inviscid forces
M_x-vis	Moment about user x axis due to viscous forces
M_y-vis	Moment about user y axis due to viscous forces
M_z-vis	Moment about user z axis due to viscous forces
rM_x-vis	Running moment about user x axis due to viscous forces
rM_y-vis	Running moment about user y axis due to viscous forces
rM_z-vis	Running moment about user z axis due to viscous forces
M_x-tot	Total moment about user x axis
M_y-tot	Total moment about user y axis
M_z-tot	Total moment about user z axis
rM_x-tot	Running total moment about user x axis
rM_y-tot	Running total moment about user y axis
rM_z-tot	Running total moment about user z axis
Heat-Transfer	Heat transfer
rHeat-Transfer	Running heat transfer
Heat-Flux	Heat flux at absolute plane