Numerical Simulation of Unsteady Hypersonic Chemically Reacting Flow

by

David E. Taflin

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

1995

Approved by _____
(Chairman of Supervisory Committee)

Program Authorized
to Offer Degree Department of Aeronautics and Astronautics

Date December 11, 1995

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature David E Jaffe

Date 12/12/95

University of Washington

Abstract

Numerical Simulation of Unsteady Hypersonic Chemically Reacting Flow

by David E. Taflin

Chairman of Supervisory Committee: Professor D. Scott Eberhardt
Dept. of Aeronautics and Astronautics

The goal of this research is to develop and evaluate a new algorithm for the numerical solution of the axisymmetric Navier-Stokes equations for unsteady hypersonic flow with chemically reacting gas species. The LU-SGS algorithm with a diagonal approximation to the chemical source Jacobian is extended to encompass a logarithmic form of the species conservation equations. It is then further extended to allow time-accurate calculations without inversion of block diagonal matrices. The algorithm is combined with second-order upwind differencing for convective fluxes and central differencing for viscous fluxes to produce an algorithm which is second-order accurate in space and time. This new algorithm is then applied to both steady and unsteady experimental results by employing both an eight-species, nine-reaction model and a nine-species, nineteen-reaction model for the combustion of hydrogen and oxygen. Excellent agreement is achieved in both cases, confirming the accuracy of the algorithm. The efficiency of the algorithm is then compared with results of other researchers. Its computational expense per iteration is shown to be nearly linear with the number of chemical species, where the expense of other algorithms varies with the cube of the number of species. This advantage is reduced by a slower convergence rate per iteration. Its memory requirements are also shown to be linear with the number of species, where those of other algorithms vary with the square of the number of species. As a result of these properties, the new algorithm is shown to be competitive with other algorithms in terms of computational expense, and vastly superior in terms of memory requirements.

Table of Contents

List of Figures

Page
Page

List of Tables

Page

Nomenclature

A	Convective flux Jacobian matrix $= \frac{\partial F}{\partial Q}$
$A^{\pm}$	Eigenvalue-split flux Jacobian matrix
$A_{f,j}$	Reaction coefficient for reaction j
B	Convective flux Jacobian matrix $= \frac{\partial G}{\partial Q}$
$B_{f,j}$	Reaction coefficient for reaction j
$C_{f,j}$	Reaction coefficient for reaction j
a	Frozen speed of sound
c_i	Species mass fraction
c_p	Specific heat at constant pressure
c_{p_i}	Species specific heat at constant pressure
c_v	Specific heat at constant volume
e	Total energy per unit volume
F	Vector of convective fluxes in the x or ξ direction
F_v	Vector of viscous fluxes in the x or ξ direction
G	Vector of convective fluxes in the y or η direction
G_v	Vector of viscous fluxes in the y or η direction
g_i^0	Species Gibbs free energy
H	Vector of axisymmetric source terms
h_i	Species enthalpy per unit mass
h_i^0	Species heat of formation
J	Jacobian of coordinate transformation
K_c	Equilibrium constant
k	Thermal conductivity
$k_{b,j}$	Backward reaction constant for reaction j
$k_{f,j}$	Forward reaction constant for reaction j

L	Characteristic length
M_i	Molecular weight of species i
nl	The number of atomic elements
nr	The number of chemical reactions
ns	The number of chemical species
p	Static pressure
Q	Vector of conserved variables
q	Heat conduction
q_i	Element of conserved variable vector
R	Specific gas constant
$\hat{R}$	Matrix of right eigenvectors of $\hat{A}$
$\hat{R}^{-1}$	Matrix of left eigenvectors of $\hat{A}$
Re	Reynolds number based on sound speed $= \frac{\rho_\infty a_\infty L}{\mu_\infty}$
R_i	Species specific gas constant
r_A	Spectral radius of A
r_B	Spectral radius of B
$r_{b,j}$	Backward rate of reaction j
$r_{f,j}$	Forward rate of reaction j
s_i	Species specific entropy
T	Static temperature
U	Contravariant velocity in ξ direction
V	Contravariant velocity in η direction
t	Time
u	Flow speed in x direction
v	Flow speed in y direction
W	Vector of source terms due to chemical reactions
w_i	Species production due to chemical reactions
X_i	Species mole fraction
$[X_i]$	Species molar concentration

x	x-coordinate
y	y-coordinate
Z	Chemical source Jacobian matrix
z_i	Coefficients of curve fits for species enthalpy and specific heat

Greek Symbols

α	Vector used to calculate TVD dissipation operator $= \hat{R}^{-1}\Delta Q$
γ	Ratio of specific heats
Δ	Forward difference
δ	Central difference
ε	Internal Energy
η	Curvilinear coordinate normal to the body surface
Λ	Diagonal matrix of the eigenvalues of A
λ	Second coefficient of viscosity
μ	Dynamical viscosity
μ_i	Species dynamical viscosity
ν	Kinematic viscosity
π_i	Logarithmic species i
ξ	Curvilinear coordinate along the body surface
ρ	Density
ρ_i	Density of species i
ρ_{i^*}	Density of atomic species i
τ	Time in generalized curvilinear coordinates
τ_{xx}	Viscous flux in x direction
τ_{xy}	Transverse viscous flux
τ_{yy}	Viscous flux in y direction
Φ	TVD dissipation operator
Ψ	Eigenvalue smoothing function

Other Symbols

∇	Backward difference
$[X_i]$	Species molar concentration
$\mu(x)$	Roe-averaged value of variable x
$\lvert\ \rvert$	Absolute value

Subscripts and Superscripts

$(\ )_e$	Derivative with respect to e
$(\ )_{i,j}$	Variable at grid indices (i,j)
$(\ )_\eta$	Derivative with respect to η
$(\ )_{\pi i}$	Derivative with respect to π_i
$(\ )_{\rho i}$	Derivative with respect to the density of species i
$(\ )_{\rho i^*}$	Derivative with respect to the density of atomic species i
$(\ )_\tau$	Derivative with respect to τ
$(\ )_\xi$	Derivative with respect to ξ
$(\ )_\infty$	Free stream value
$(\ )^n$	Present time level
$(\ )^{n+1}$	New time level
$\hat{(\ )}$	Midpoint averaged value
$\dot{(\ )}$	Time derivative

ACKNOWLEDGMENTS

The author wishes to thank Professor Scott Eberhardt for guidance, insight, encouragement, and help in preparing this document; Brian Leverson for tirelessly maintaining the computer hardware used in this work; labmates Ashok Bangalore, Chen Chuck, Linda Hedges, Oggie Jones, Byoungsoo Kim, Wei Lin Li, Pete McQuade, Honam Ok, Steve Pluntze, Bogdan Udrea, Lora Wirth, and Bill Wood for making the experience enjoyable; Moeljo Soetrisno, Scott Imlay, and the rest of the AMTEC crew for moral and technical support, as well as the use of their fine product, TECplot. Special thanks to my parents, Charles and Marlys Taflin, and to my wife, Cathy, for patience, encouragement, and advice.

To Cathy

Chapter 1: Introduction

Recent interest in hypersonic vehicles, as well as continuing development of reentry vehicles and of the Ram Accelerator, have created the need for efficient numerical methods to predict the effects of chemical reactions in high speed flows. Due to their superior stability, many implicit algorithms have been developed to solve for steady-state flows. These algorithms have generally not been time-accurate, so explicit algorithms have been largely relied on for the calculation of unsteady flows. Since explicit algorithms introduce significant time-step restrictions, however, it is desirable to develop implicit algorithms capable of calculating unsteady flows.

1.1 Background

There has been much recent interest in high-speed vehicle development. Much of this effort, of course, has involved the Space Shuttle and other reentry vehicles, such as the aeroassisted orbit transfer vehicle (AOTV). Other efforts have been in the area of single-stage-to-orbit (SSTO) vehicles and other supersonic/hypersonic vehicles with air-breathing propulsion plants. These vehicle concepts introduce at least two areas where chemically-reacting flow may exist. First, at high supersonic speeds, oxygen begins to dissociate, producing atomic oxygen and other radical species. These radicals may significantly affect the characteristics of the flow, especially the temperature distribution near the stagnation point on the forward portion of the vehicle, which has obvious thermal protection design implications.

Secondly, non-conventional air-breathing engine designs are required for flight beyond about Mach 5.[1] Perhaps the most highly favored engine concept is the supersonic combustion ramjet, or scramjet.[2] In this type of engine, air is compressed by a shock or system of shocks prior to entering the engine inlet, but remains supersonic relative to the vehicle. Fuel is the injected, and combustion takes place. Due to the extremely short residence time of the fuel/air mixture, however, this combustion process continues beyond the engine outlet, and the combustion products expand against the after portion of the vehicle, providing forward thrust. The combustion process involves many chemical species, and many reactions which take place at time scales similar to the fluid dynamic time scales. Thus, it is necessary to model these chemical reactions as finite-rate processes, as opposed to equilibrium processes. In addition, the mixing of the fuel with the air may involve Kelvin-Helmholtz or other unsteady phenomena, which must be studied in order to optimize the mixing process.[3-5]

Another concept which has characteristics similar to the scramjet is the Ram Accelerator, being developed at the University of Washington and elsewhere.[6-17]

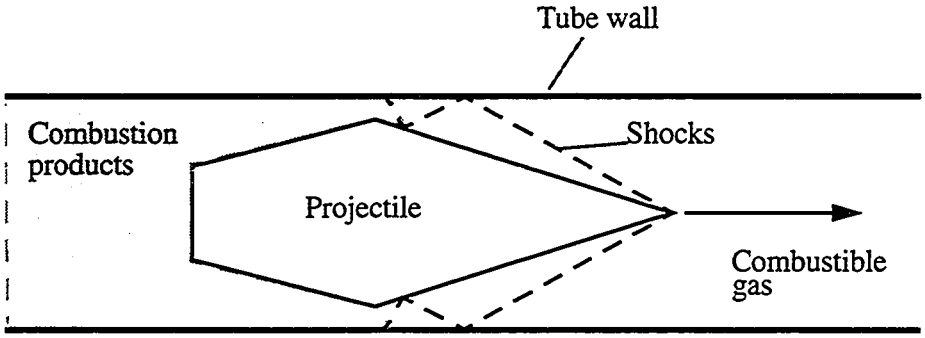

Figure 1-1: A Schematic of a Ram Accelerator

Figure 1-1 shows a schematic of this device, in which a projectile is fired at supersonic speeds into a tube filled with a combustible mixture of gases. When the projectile is flying faster than the detonation speed of the gas mixture, the shock wave pattern which forms around the projectile ignites the gas mixture. The resulting combustion increases the pressure on the aft portion of the projectile, accelerating it up to several kilometers per second. Intended applications of this concept include launching projectiles into low Earth orbit.

To maximize the performance of the Ram Accelerator at various projectile speeds, the gas mixture is altered at different stations in the tube, separated by thin diaphragms. The projectile's transition from one gas mixture to the next is an interesting unsteady problem, as is the start-up of combustion when the projectile first enters the combustible mixture.

Another fluid flow problem which involves unsteady chemically-reacting flow is the Pulse Detonation Engine (PDE),[18] rumored to be used in the latest reconnaissance aircraft, the Aurora. Figure 1-2 shows a schematic of a PDE taken from Bussing *et al.*[19] In this engine concept, a detonable gas mixture is admitted to a detonation chamber via an inlet and a mixer. In the mixer, fuel, and possibly an oxidizer, are mixed with the air. Distribution manifolds are designed to ensure thorough mixing. The detonation chamber is then (usually) sealed at the forward end. A detonation is initiated at the forward end of the chamber, and thrust is generated as the combustion products exit the after end of the chamber through a nozzle. This type of engine offers good performance from zero to high supersonic speeds. It can achieve higher efficiency than gas turbine engines due to its approximation of the constant volume Humphrey cycle, compared to the Brayton cycle seen in gas turbine engines.[18] It also contains few moving parts, in stark contrast to gas turbine engines.

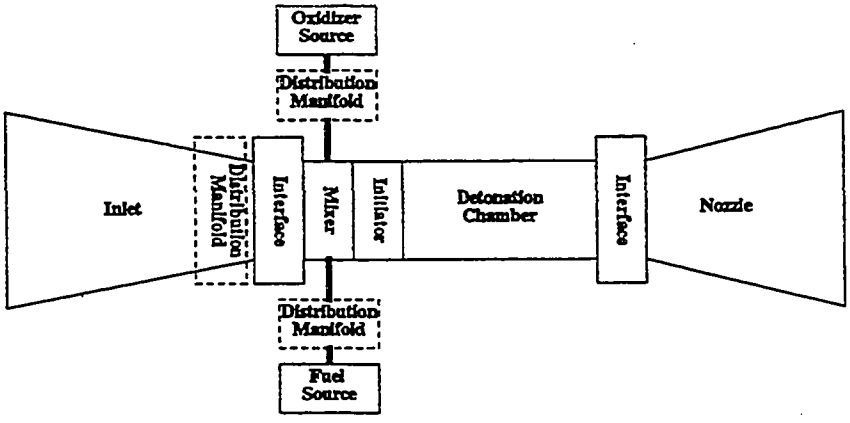

Figure 1-2: A Schematic of a Pulse Detonation Engine

Another possible area involving high-speed unsteady flow with chemical reactions is the shock wave reactor concept, being studied at the University of Washington.[20-22] The shock wave reactor is intended to replace conventional methods for the production of ethylene, an important chemical precursor to many commercial products. It involves the supersonic flow of a pre-heated gas mixture through a diverging nozzle, where a normal shock occurs which suddenly heats the gas, forming ethylene. The gas is then rapidly cooled, and the ethylene is extracted. Much remains to be learned about the details of the flow through this device.

1.1.1 The Ram Accelerator

Experimental work on the Ram Accelerator has been ongoing for several years at the University of Washington.[6-11] The work has included photographing the projectile in flight to determine the location of the combustion processes, as well mapping in detail the pressure distribution along the wall of the tube. These detailed measurements provide an excellent opportunity to validate computational fluid dynamics codes intended for the calculation of hypersonic, chemically-reacting flow.

Computational studies of the Ram Accelerator have been performed by Yungster,[12,13] Chuck,[14,15] Oran *et al.*,[17] Soetrisno *et al.*,[23] and others. Yungster performed steady simulations of inviscid flow, while Chuck simulated viscous flow. Both used a point implicit method (to be discussed in §3.2). Chuck, in addition, used a variant first proposed by Eberhardt which eliminates much of the storage requirements of the point-implicit method and reduces the computer time required for steady-state calculations. This method was further implemented by Eberhardt and Imlay[24] in an implicit algorithm which showed dramatic savings in computational expense. This method has been further applied to Ram Accelerator configurations by Soetrisno *et al.*,[23] and a modification to it has been introduced by Candler.[25,26] Eberhardt's algorithm, which involves a diagonal approximation to the chemical source Jacobian, appears to offer significant reductions in both storage requirements and computational expense for steady flow, and its use in unsteady calculations will be examined in this work.

1.1.2 The Pulse Detonation Engine

Much of the work being done on PDE's is either proprietary or covered by other restrictions.[19] It is clear, however, that this is an active area of current research, both numerical and experimental. Experimental work has recently been outlined by Ting *et al.*,[27] and Hinkey *et al.*[28] Computational work has been performed by Bussing *et al.*,[19] Sterling *et al.*,[29] and Lynch *et al.*[30] Bussing *et al.* and Sterling *et al.* used a one-dimensional algorithm to study the unsteady behavior of the gas within the detonation chamber. Lynch *et al.* performed axisymmetric unsteady calculations for hydrogen mixtures using an implicit algorithm which allowed larger time steps than would have been allowed by a point-implicit approach. This approach, however, required inversion of block-banded Jacobian matrices, the expense of which prevented them from using this technique for larger reaction models, for which they resorted to the point-implicit scheme.

1.1.3 Lehr's Experiments

An interesting set of experiments involving unsteady phenomena is the work performed by Lehr.[31] Lehr fired both sphere-cylinder and cone-cylinder projectiles into premixed stoichiometric hydrogen-air and hydrogen-oxygen gas mixtures. Depending on the speed of the projectile relative to the detonation velocity of the gas, different shock and combustion patterns emerged. At speeds above the detonation velocity in hydrogen-air, a coupled shock-deflagration formed near the centerline of the body. Further away from the centerline, at the point where the velocity normal to the shock fell below the detonation velocity, the combustion front separated from the shock. The induction zone (the distance between the shock and the combustion front) was then used to determine the ignition delay time. Figure 1-3 depicts this case.

At projectile speeds just below the detonation velocity, Lehr observed a remarkable unsteady phenomenon. The interaction of alternating compression and expansion waves near the stagnation line with the shock and combustion processes resulted in a very regular "ringing" behind the shock. McVey and Toong[32] proposed a model to explain this phenomenon. This model will be outlined in Chapter 4.

Wilson,[33] and Wilson and Sussman,[34] developed a numerical algorithm which was able to reproduce Lehr's unsteady results. It employed a new logarithmic form of the species conservation equations first proposed by Sussman and Wilson.[35] The advantage of this form will be discussed in §2.6. Numerical reproduction of this unsteady experiment gives an excellent validation for unsteady CFD algorithms.

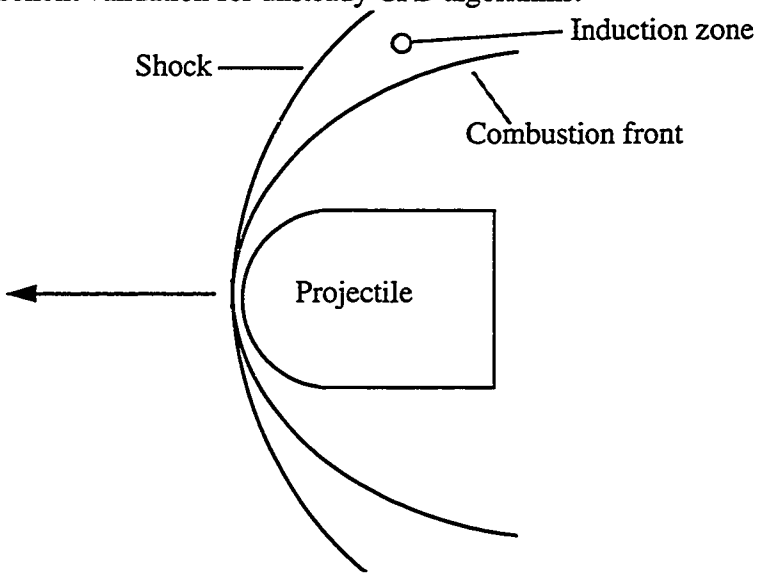

Figure 1-3: A Schematic of Lehr's Superdetonative Hydrogen-Air Case

1.2 Present Work

The advantages of implicit, time-accurate methods for the calculation of chemically-reacting flows have been demonstrated by the work of Sterling *et al.*,[29] Shuen *et al.*,[36] Yungster and Radhakrishnan,[37] and Ju.[38] All of these approaches, however, involve at a minimum the inversion of a block diagonal matrix for each time step. The blocks of this matrix are composed of Jacobians of both the convective flux terms and the chemical source terms, and are of the same order as the number of chemical species. The computational expense of this inversion varies with the cube of the number of species. Further, as the number of chemical species becomes large, the storage requirement can become prohibitive. These factors motivate the development of algorithms which avoid the use of full diagonal blocks.

Eberhardt's algorithm eliminates storage and inversion of the block diagonal, but destroys time accuracy. It is possible, however, to reintroduce time accuracy through the same sub-iterative approach used by Shuen *et al.* The present work is intended to develop and examine the effectiveness of this modification. Further, since the logarithmic form of the species conservation equations of Sussman and Wilson appears to be a highly useful technique, Eberhardt's algorithm is extended to the use of these equations.

The resulting algorithm reduces the amount of computational grid required for flow problems involving shock-induced combustion, thereby significantly improving an algorithm which is already highly efficient. It is fully implicit, yet requires no matrix inversions. Its memory requirements and computational expense per iteration scale linearly with the number of chemical species, where for other algorithms (aside from Eberhardt and Imlay's original algorithm) they scale with the square and the cube of the number of species, respectively. Thus, the algorithm gives increasing benefit over other algorithms as the number of species increases.

The algorithm is validated against the steady and unsteady experimental results of Lehr using both an eight-species, nine-step reaction model and a nine-species, nineteen-step reaction model for hydrogen-air combustion. It is also compared to computational results from Yungster and Rabinowitz[39] for the computation of a combusting methane-air mixture using their twenty-species, fifty-two-step reaction model. The algorithm is found to give highly accurate results for complex flow fields. Its performance is measured against other published results, and it is found to be competitive in terms of CPU time, and superior in terms of memory requirements.

Chapter 2: Governing Equations

The first step in numerical solutions is to define the equations which will be solved. In this chapter, non-dimensional variables will be introduced, followed by a transformation of the governing equations to generalized curvilinear coordinates. The resulting equations will then be shown in detail. Chemical reactions will be discussed, and finally, the logarithmic form of the species conservation laws will be introduced.

2.1 Nondimensional Variables

Nondimensional variables are formed by dividing the dimensional variables of the problem by certain variables or combinations of variables which have the same dimensions as the variables of interest, and which typically characterize the physical problem in some way. For example, the flow speed may be nondimensionalized by dividing it by the free stream flow speed or by the free stream speed of sound. One advantage of nondimensional variables is that most are of order one, thus reducing round-off error. In some cases, it becomes easier to specify free stream and boundary conditions when nondimensional variables are used. In particular, when the flow speeds are nondimensionalized by the free stream sound speed, specifying the free stream flow speed reduces to specifying the free stream Mach number, a significant advantage. The use of nondimensional variables may also allow results to be more readily interpreted and generalized to similar flows.

The following definitions define the nondimensional variables used in this work. Dimensional variables are indicated with a tilde (~), and free stream (dimensional) quantities have the infinity (∞) subscript.

$$
\begin{aligned}
x &= \frac{\tilde{x}}{L} \qquad y = \frac{\tilde{y}}{L} \qquad t = \tilde{t}\frac{a_\infty}{L} \\
\rho_i &= \frac{\tilde{\rho}_i}{\rho_\infty} \qquad u = \frac{\tilde{u}}{a_\infty} \qquad v = \frac{\tilde{v}}{a_\infty} \\
e &= \frac{\tilde{e}}{\rho_\infty a_\infty^2} \qquad p = \frac{\tilde{p}}{\rho_\infty a_\infty^2} \qquad T = \frac{\tilde{T}}{\gamma_\infty T_\infty} \\
R &= \frac{\tilde{R}}{R_\infty} \qquad c_p = \frac{\tilde{c}_p}{R_\infty} \qquad c_v = \frac{\tilde{c}_v}{R_\infty} \\
\mu &= \frac{\tilde{\mu}}{\mu_\infty} \qquad k = \frac{\tilde{k}}{k_\infty} \qquad h_i^0 = \frac{\tilde{h}_i^0}{a_\infty^2} \\
w_i &= \frac{L}{\rho_\infty a_\infty}\tilde{w}_i
\end{aligned}
\tag{2-1}
$$

In the above equations, x and y are the spatial coordinates — x for the axial direction, and y for the radial direction. t represents time. ρ is density, ρ_i are species densities, and u and v are the flow speeds in the x- and y-directions, respectively. e is the total energy per unit volume, p is the static pressure, and T is the static temperature of the gas mixture. R is the gas constant, and c_p and c_v are the gas specific heats at constant pressure and constant volume, respectively. μ is the coefficient of dynamic viscosity, k is the coefficient of thermal conductivity, h_i^0 is the species specific heat of formation per unit mass, and w_i represents the production of species i due to chemical reactions. L is a length scale factor, normally chosen to be the length of the body about which the flow solution is to be calculated. γ is the ratio of specific heats, and a represents sound speed.

2.2 Coordinate Transformation

The axisymmetric Navier-Stokes equations can be expressed as a system of partial differential equations, in the following form:

$$\frac{\partial yQ'}{\partial t} + \frac{\partial yF'}{\partial x} + \frac{\partial yG'}{\partial y} = \frac{\partial yF'_v}{\partial x} + \frac{\partial yG'_v}{\partial y} + H' + yW' \qquad (2\text{-}2)$$

Q' represents the vector of conserved variables, mass, momentum, and energy, and F' and G' represent the convective fluxes of these quantities. F'_v and G'_v are the fluxes due to molecular transport, H' represents source terms due to axisymmetry, and W' represents source terms due to chemical reactions. These terms will be examined in detail in §2.3.

Numerical solution of the Navier-Stokes equations in the form of equation (2-2) requires that the problem be solved on a uniform mesh, making solutions about arbitrary bodies difficult (see Figure 2-1).[40]

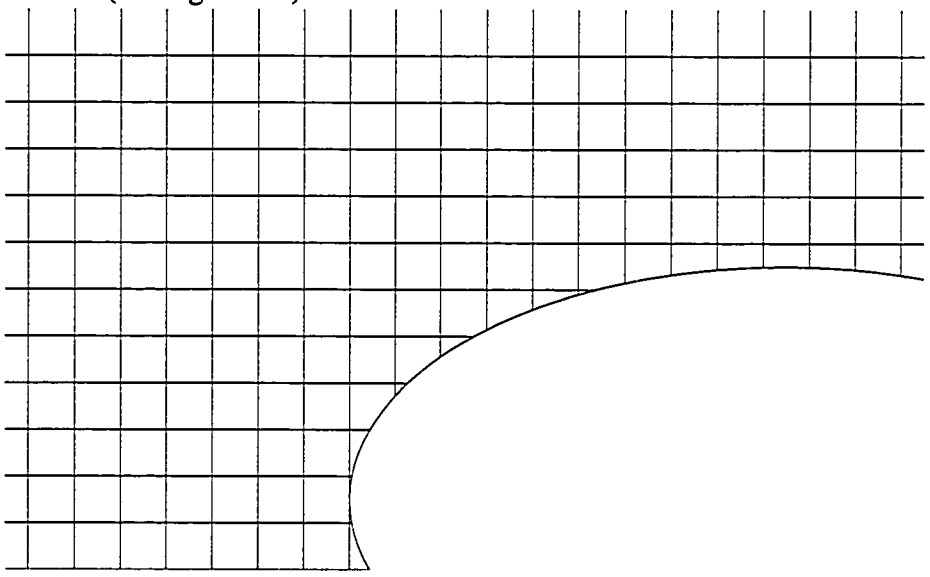

Figure 2-1:A Cartesian Computational Grid

Although this type of approach is sometimes used, it requires careful treatment of the boundaries, and makes the solution of problems involving boundary layers impractical. It

is desirable, therefore, to use a grid which can be conformed to the shape of the body, or the boundaries of the physical domain. Figure 2-2 shows an example of a body-fitted grid.

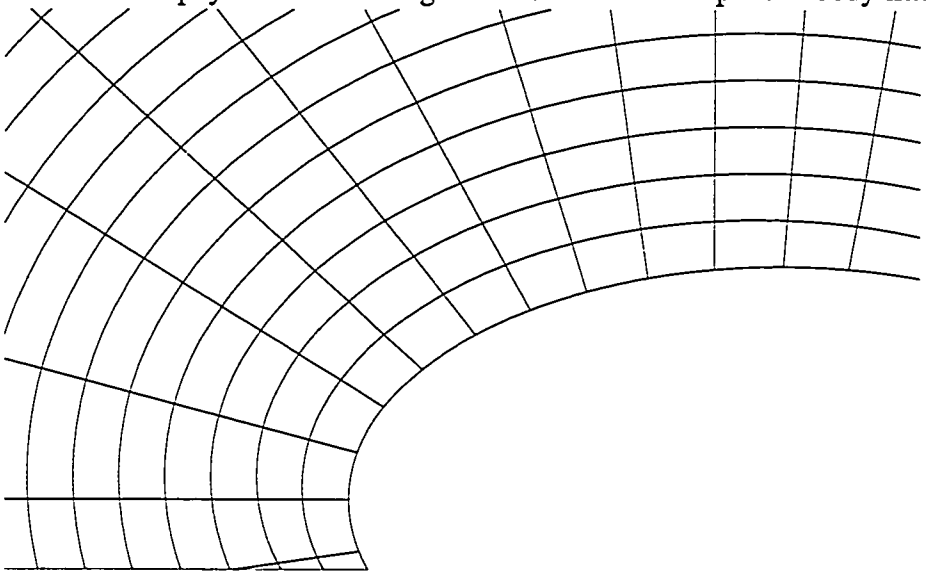

Figure 2-2:A Body-Fitted Computational Grid.

To allow solutions to be done on body-fitted grids, equation (2-2) is transformed to a generalized curvilinear coordinate system, (ξ, η, τ) . The chain rule for differentiation gives, for example,

$$\frac{\partial Q'}{\partial t} = \frac{\partial Q'}{\partial \tau}\frac{\partial \tau}{\partial t} + \frac{\partial Q'}{\partial \xi}\frac{\partial \xi}{\partial t} + \frac{\partial Q'}{\partial \eta}\frac{\partial \eta}{\partial t} \tag{2-3}$$

Setting $\tau = t$, the transformation can be expressed as

$$\begin{bmatrix} \partial_t \\ \partial_x \\ \partial_y \end{bmatrix} = \begin{bmatrix} 1 & \xi_t & \eta_t \\ 0 & \xi_x & \eta_x \\ 0 & \xi_y & \eta_y \end{bmatrix} \begin{bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\eta \end{bmatrix} \tag{2-4}$$

with the definitions

$$\begin{aligned} \xi_x &= \frac{\partial \xi}{\partial x} \qquad \xi_y = \frac{\partial \xi}{\partial y} \qquad \xi_t = \frac{\partial \xi}{\partial t} \\ \eta_x &= \frac{\partial \eta}{\partial x} \qquad \eta_y = \frac{\partial \eta}{\partial y} \qquad \eta_t = \frac{\partial \eta}{\partial t} \end{aligned} \tag{2-5}$$

Reversing the roles of dependent and independent variables in equation (2-4) gives

$$\begin{bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\eta \end{bmatrix} = \begin{bmatrix} 1 & x_\tau & y_\tau \\ 0 & x_\xi & y_\xi \\ 0 & x_\eta & y_\eta \end{bmatrix} \begin{bmatrix} \partial_t \\ \partial_x \\ \partial_y \end{bmatrix} \tag{2-6}$$

with the definitions

$$\begin{aligned} x_\tau &= \frac{\partial x}{\partial \tau} \qquad x_\xi = \frac{\partial x}{\partial \xi} \qquad x_\eta = \frac{\partial x}{\partial \eta} \\ y_\tau &= \frac{\partial y}{\partial \tau} \qquad y_\xi = \frac{\partial y}{\partial \xi} \qquad y_\eta = \frac{\partial y}{\partial \eta} \end{aligned} \tag{2-7}$$

Comparing equation (2-4) with equation (2-6), it can be immediately deduced that

$$\begin{bmatrix} 1 & \xi_t & \eta_t \\ 0 & \xi_x & \eta_x \\ 0 & \xi_y & \eta_y \end{bmatrix} = \begin{bmatrix} 1 & x_\tau & y_\tau \\ 0 & x_\xi & y_\xi \\ 0 & x_\eta & y_\eta \end{bmatrix}^{-1} = J \begin{bmatrix} x_\xi y_\eta - x_\eta y_\xi - x_\tau y_\eta + x_\eta y_\tau & x_\tau y_\xi - x_\xi y_\tau \\ 0 & y_\eta & -y_\xi \\ 0 & -x_\eta & x_\xi \end{bmatrix} \tag{2-8}$$

with J, the Jacobian of the transformation, defined by

$$J^{-1} = x_\xi y_\eta - x_\eta y_\xi \tag{2-9}$$

This results in the following relations:

$$\begin{aligned} \xi_t &= J(-x_\tau y_\eta + x_\eta y_\tau) \qquad \xi_x = J y_\eta \qquad \xi_y = -J x_\eta \\ \eta_t &= J(x_\tau y_\xi - x_\xi y_\tau) \qquad \eta_x = -J y_\xi \qquad \eta_y = J x_\xi \end{aligned} \tag{2-10}$$

Once a computational grid is defined, all of the terms in equations (2-9) and (2-10) can be calculated. For ease of calculation, each grid node is assigned a consecutive integral value of ξ and η, so that between neighboring grid cells, $\Delta\xi = \Delta\eta = 1$.

For a grid which does not move in time, $\xi_t = \eta_t = 0$. The transformed equations are formed by expanding each of the terms in equation (2-2) using equation (2-4), giving the following for a non-moving grid:

$$\begin{aligned}&\frac{\partial yQ'}{\partial\tau}+\xi_x\frac{\partial yF'}{\partial\xi}+\eta_x\frac{\partial yF'}{\partial\eta}+\xi_y\frac{\partial yG'}{\partial\xi}+\eta_y\frac{\partial yG'}{\partial\eta}\\&=\xi_x\frac{\partial yF'_v}{\partial\xi}+\eta_x\frac{\partial yF'_v}{\partial\eta}+\xi_y\frac{\partial yG'_v}{\partial\xi}+\eta_y\frac{\partial yG'_v}{\partial\eta}+H'+yW'\end{aligned}\tag{2-11}$$

Equation (2-11) is in a non-conservative form, and must be put back into conservation law form. This is done by multiplying by J^{-1} and again invoking the chain rule for each term, for example:

$$J^{-1}\xi_x\frac{\partial yF'}{\partial\xi}=\frac{\partial J^{-1}\xi_x yF'}{\partial\xi}-yF'\frac{\partial J^{-1}\xi_x}{\partial\xi}\tag{2-12}$$

This gives, after combining like terms,

$$\begin{aligned}&\frac{\partial}{\partial\tau}J^{-1}yQ'+\frac{\partial}{\partial\xi}J^{-1}y(\xi_xF'+\xi_yG')+\frac{\partial}{\partial\eta}J^{-1}y(\eta_xF'+\eta_yG')\\&\qquad-y(F'-F'_v)\left(\frac{\partial}{\partial\xi}J^{-1}\xi_x+\frac{\partial}{\partial\eta}J^{-1}\eta_x\right)\qquad(\textit{term 1})\\&\qquad-y(G'-G'_v)\left(\frac{\partial}{\partial\xi}J^{-1}\xi_y+\frac{\partial}{\partial\eta}J^{-1}\eta_y\right)\qquad(\textit{term 2})\\&=\frac{\partial}{\partial\xi}J^{-1}y(\xi_xF'_v+\xi_yG'_v)+\frac{\partial}{\partial\eta}J^{-1}y(\eta_xF'_v+\eta_yG'_v)+J^{-1}H'+J^{-1}yW'\end{aligned}\tag{2-13}$$

The terms labeled *term 1* and *term 2* above are invariants of the transformation, and are identically zero. With the definitions

$$\begin{aligned}&Q\equiv J^{-1}yQ'\qquad F\equiv J^{-1}y(\xi_xF'+\xi_yG')\qquad G\equiv J^{-1}y(\eta_xF'+\eta_yG')\\&F_v\equiv J^{-1}y(\xi_xF'_v+\xi_yG'_v)\qquad G_v\equiv J^{-1}y(\eta_xF'_v+\eta_yG'_v)\\&H\equiv J^{-1}H'\qquad W\equiv J^{-1}yW'\end{aligned}\tag{2-14}$$

equation (2-13) becomes

$$\frac{\partial Q}{\partial\tau}+\frac{\partial F}{\partial\xi}+\frac{\partial G}{\partial\eta}=\frac{\partial F_v}{\partial\xi}+\frac{\partial G_v}{\partial\eta}+H+W\tag{2-15}$$

Equation (2-15) is the form of the axisymmetric Navier-Stokes equations which will be solved. The individual terms in this equation set are discussed in section (2.3).

2.3 Navier-Stokes Equations

The axisymmetric Navier-Stokes equations represent the conservation of mass, momentum, and energy. To allow for multiple species with chemical reactions, the continuity equation can be replaced by species conservation equations, one for each species. In generalized curvilinear coordinates and non-dimensional variables, they are given by

$$\frac{\partial Q}{\partial \tau}+\frac{\partial F}{\partial \xi}+\frac{\partial G}{\partial \eta}=R_e^{-1}\frac{\partial F_v}{\partial \xi}+R_e^{-1}\frac{\partial G_v}{\partial \eta}+H+W \tag{2-16}$$

with the definitions

$$Q=J^{-1}y\begin{bmatrix}\rho_1\\ \vdots\\ \rho_n\\ \rho u\\ \rho v\\ e\end{bmatrix} \qquad F=J^{-1}y\begin{bmatrix}\rho_1 U\\ \vdots\\ \rho_n U\\ \rho uU+\xi_x p\\ \rho vU+\xi_y p\\ U(e+p)\end{bmatrix} \qquad G=J^{-1}y\begin{bmatrix}\rho_1 V\\ \vdots\\ \rho_n V\\ \rho uV+\eta_x p\\ \rho vV+\eta_y p\\ V(e+p)\end{bmatrix}$$

$$F_v=J^{-1}y\begin{bmatrix}-\rho_1\hat{U}_1\\ \vdots\\ -\rho_n\hat{U}_n\\ \xi_x\tau_{xx}+\xi_y\tau_{xy}\\ \xi_x\tau_{yx}+\xi_y\tau_{yy}\\ \xi_x\beta_x+\xi_y\beta_y\end{bmatrix} \qquad G_v=J^{-1}y\begin{bmatrix}-\rho_1\hat{V}_1\\ \vdots\\ -\rho_n\hat{V}_n\\ \eta_x\tau_{xx}+\eta_y\tau_{xy}\\ \eta_x\tau_{yx}+\eta_y\tau_{yy}\\ \eta_x\beta_x+\eta_y\beta_y\end{bmatrix} \tag{2-17}$$

$$W=J^{-1}y\begin{bmatrix}w_1\\ \vdots\\ w_n\\ 0\\ 0\\ 0\end{bmatrix} \qquad H=J^{-1}\begin{bmatrix}0\\ \vdots\\ 0\\ -\frac{2}{3}\frac{\partial}{\partial x}\mu v\\ p-\frac{4}{3}\mu\frac{v}{y}+\frac{2}{3}\mu\frac{\partial u}{\partial x}-\frac{2}{3}v\frac{\partial y}{\partial \mu}\\ -\frac{2}{3}\left(\frac{\partial}{\partial x}\mu uv+\frac{\partial}{\partial y}\mu v^2\right)\end{bmatrix}$$

R_e is the Reynold's number based on free stream sound speed, and is given by

$$R_e = \frac{\rho_\infty a_\infty L}{\mu_\infty} \tag{2-18}$$

It appears in equation (2-16) due to the nondimensionalization of the equations. U and V are the contravariant velocities, which are flow speeds normal to grid lines, scaled by grid metric terms. They are defined as:

$$U \equiv \xi_x u + \xi_y v \qquad V \equiv \eta_x u + \eta_y v \tag{2-19}$$

$\hat{U}$ and $\hat{V}$ are species diffusion contravariant velocities, functions of the species diffusion velocities and given by expressions identical in form to those of equation (2-19). Because species diffusion is not significant in the types of problems to be examined here, it will be neglected. The bulk viscosity is assumed to be zero, leading to $\lambda = -\frac{2}{3}\mu$. With this assumption, and heat conduction given by $q = -k\nabla T$, the remainder of the diffusive flux terms become:

$$\begin{gathered} \tau_{xx} \equiv \frac{4}{3}\mu u_x - \frac{2}{3}\mu v_y \qquad \tau_{yy} \equiv \frac{4}{3}\mu v_y - \frac{2}{3}\mu u_x \qquad \tau_{xy} = \tau_{yx} \equiv \mu (u_y + v_x) \\ \beta_x \equiv u\tau_{xx} + v\tau_{xy} + kT_x \qquad \beta_y \equiv u\tau_{xy} + v\tau_{yy} + kT_y \end{gathered} \tag{2-20}$$

2.4 Chemical Reactions

W in equation (2-16) contains species source terms due to chemical reactions. A typical chemical reaction is shown below:

$$\nu'_{A,j} A + \nu'_{B,j} B \leftrightarrow \nu''_{C,j} C + \nu''_{D,j} D \tag{2-21}$$

Here $\nu'_{A,j}$ atoms of species A combine with $\nu'_{B,j}$ atoms of species B to form $\nu''_{C,j}$ atoms of species C and $\nu''_{D,j}$ atoms of species D. A general chemical reaction can be expressed as

$$\sum_{i=1}^{ns} \nu'_i X_i \leftrightarrow \sum_{i=1}^{ns} \nu''_i X_i \tag{2-22}$$

where X_i is chemical species i, and ns is the number of chemical species.

If a single chemical reaction is in equilibrium, one constant, usually called K_c or K_p, can be used to calculate the concentrations or mass fractions of the involved species. For example, K_c is defined by

$$K_c \equiv \frac{\prod_{i=1}^{ns} [X_i]^{\nu''_i}}{\prod_{i=1}^{ns} [X_i]^{\nu'_i}} \tag{2-23}$$

where $[X_i]$ is the dimensional molar concentration of species i, calculated from

$$[X_i] = \frac{\rho_\infty \rho_i}{M_i} \tag{2-24}$$

M_i is the molecular weight of species i. Equation (2-23) is one form of the famous Law of Mass Action, and its use allows direct calculation of all species mass fractions from state variables and other information known a priori. If more than one equilibrium chemical reaction is present, more sophisticated techniques, such as Gibbs free energy minimization, must be used. Nevertheless, all species mass fractions can be calculated directly.

Chemical equilibrium is normally observed in a flow if all the chemical reactions occur at a rate far exceeding that of any other changes, such as those due to fluid dynamics. Another limiting occurrence is frozen flow, where chemical reactions occur so slowly compared to other changes in the flow that species mass fractions remain essentially unchanged. If neither of these cases is true, and chemical reaction rates approach those of other changes in the flow, then chemical non-equilibrium results. In this case, species mass fractions can no longer be calculated solely from state information, and must be represented by additional differential equations. This is accomplished, as noted before, by solv-

ing a continuity equation for each chemical species, and representing species production due to chemical reactions by source terms in these equations.

Instead of a single constant, non-equilibrium chemical reactions involve both a forward and a backward reaction constant to describe individually the progression of the reaction in either direction. Note that the word "constant" is a misnomer, because these parameters are normally functions of temperature. For the example reaction in equation (2-21), the forward and backward reaction rates, r_f and r_b, would be calculated from

$$r_f = k_f [X_A]^{\nu'_{A,j}} [X_B]^{\nu'_{B,j}} \qquad r_b = k_b [X_C]^{\nu''_{C,j}} [X_D]^{\nu''_{D,j}} \tag{2-25}$$

where k_f and k_b are the forward and backward reaction constants, respectively. The total species production is the forward rate minus the backward rate, multiplied by the net species production for the reaction. If a given reaction produces more of a given species than it consumes, then the production of that species will be positive if r_f is greater than r_b, such that the reaction progresses to the right, and negative if the reverse is true. In general, the production of species m is calculated from

$$\tilde{w}_m = M_m \sum_{j=1}^{nr} (\nu''_{m,j} - \nu'_{m,j}) \left[k_{f,j} \prod_{s=1}^{ns} [X_s]^{\nu'_{s,j}} - k_{b,j} \prod_{s=1}^{ns} [X_s]^{\nu''_{s,j}} \right] \tag{2-26}$$

where nr is the number of reactions and ns is again the number of chemical species. These source terms are non-dimensionalized as described in section (2.1), giving

$$w_m = \frac{L}{\rho_\infty a_\infty} \tilde{w}_m \tag{2-27}$$

The forward reaction constants, k_f, are calculated from Arrhenius expressions,

$$k_f = A T^B e^{-\frac{C}{T}} \tag{2-28}$$

where A, B, and C are constants which are different, in general, for each reaction. These may also be supplied for the reverse reaction. Otherwise, k_b is calculated from the equilibrium constant for the reaction, given by:

$$K_c = \frac{k_f}{k_b} = (R'\tilde{T})^{\sum_{s=1}^{ns}\left(\nu_s' - \nu_s''\right)} \exp\left(-\sum_{s=1}^{ns}(\nu_s' - \nu_s'')\frac{g_s^0}{R_s\tilde{T}}\right) \tag{2-29}$$

where R' refers to the universal gas constant in atmospheres, and the g_s^0 are the "one atmosphere" Gibb's free energy values, given by $g_s^0 = h_s - \tilde{T}s_s$. The calculation of h_s and s_s is described in §3.3.

Some chemical reactions involve a "third body", which can be any chemical species. An example of this type of reaction is the following, for the dissociation/recombination of oxygen:

$$O + O + M \leftrightarrow O_2 + M \tag{2-30}$$

M may be formed by summing the molar concentrations of all species. Some species, however, are more efficient catalysts for a particular reaction than are other species. This effect is accounted for by using third-body, or collision, efficiencies. Rather than sum the species concentrations directly, each concentration is first multiplied by its collision efficiency for that reaction, and the results are summed. This must be repeated for each third-body reaction.

2.5 Chemical Reaction Models

A significant challenge in computing chemically-reacting flows is determining which chemical species and chemical reactions are important. A mixture of hydrogen and air may produce over twenty species, and involve over a hundred chemical reactions. Many of these species, however, may only appear in insignificant quantities. These species and the associated reactions, therefore, may have little influence on the overall reaction and the flow. Various combustion models have been proposed which attempt to isolate only those species and reactions which significantly impact the flow. It is important to note that different flow conditions may require different reaction models.

Two reaction models for hydrogen-air combustion are used in this work. The first is a subset of the mechanism suggested by Waldman.[41] Waldman investigated methane/air combustion with pollutant formation, and recommended twenty-six reactions for sixteen species. Pratt[42] extracted a subset of these for hydrogen/oxygen combustion, involving eight species and nine reactions. He found that this model effectively predicted ignition delay in the temperature range represented in Lehr's experiments.

The second combustion model is a simplification of Jachimowski's hydrogen-air model.[43] It involves thirteen species, and in its original form had thirty-three reactions. Wilson,[33] however, modified the model. Noting that there is some uncertainty in the experimental determination of chemical reaction rates, Wilson "tuned" one of these rates within the range of uncertainty to cause his computations to match Lehr's experimental data. He also replaced one reaction, and deleted another which Jachimowski advised was probably unimportant. Yungster[37] further modified the model by determining that the reactions involving nitrogen (as other than a third-body) do not significantly affect the ignition delay, rate of heat release, or equilibrium temperature in the flows to be considered here. After eliminating these, the result is a set of 9 species and 19 reactions. This reaction model was also used by Matsuo *et al.*[44]

These combustion models are detailed in Appendix A.

2.6 The Logarithmic Species Conservation Law

Sussman and Wilson,[35] noted that the exponential variation of chemical species mass fractions commonly seen in reacting flow can lead to significant errors in convective flux calculations. These errors can inhibit proper resolution of flow phenomena, such as an induction zone between a shock and a combustion wave. This observation led to their development of a logarithmic transformation of the species conservation law. In this formulation, the species densities, ρ_i, are replaced by $\pi_i \equiv \rho \ln\left(\frac{\rho_i}{\rho}\right) = \rho \ln c_i$. The species conservation law then becomes

$$\frac{\partial}{\partial\tau}J^{-1}\pi_i+\frac{\partial}{\partial\xi}J^{-1}\pi_i U+\frac{\partial}{\partial\eta}J^{-1}\pi_i V = J^{-1}w_{\pi i} \tag{2-31}$$

The expression of $w_{\pi i}$ follows from the derivation of the logarithmic variables, which will now be outlined. The continuity equation in one dimension for species i is given by

$$\frac{\partial\rho_i}{\partial t}+\frac{\partial\rho_i u}{\partial x} = w_i \tag{2-32}$$

We first replace ρ_i with ρc_i, and divide the equation by c_i, giving

$$\frac{1}{c_i}\frac{\partial\rho c_i}{\partial t}+\frac{1}{c_i}\frac{\partial\rho c_i u}{\partial x} = \frac{w_i}{c_i} \tag{2-33}$$

The time derivative on the left-hand side is distributed as follows:

$$\frac{1}{c_i}\frac{\partial\rho c_i}{\partial t} = \frac{\partial\rho}{\partial t}+\frac{\rho}{c_i}\frac{\partial c_i}{\partial t} = \frac{\partial\rho}{\partial t}+\rho\frac{\partial\ln c_i}{\partial t} = \frac{\partial\rho}{\partial t}+\frac{\partial\rho\ln c_i}{\partial t}-\ln c_i\frac{\partial\rho}{\partial t} \tag{2-34}$$

Performing the same manipulation on the spatial derivative and rearranging the result gives

$$\left(\frac{\partial\rho}{\partial t}+\frac{\partial\rho u}{\partial x}\right)-\ln c_i\left(\frac{\partial\rho}{\partial t}+\frac{\partial\rho u}{\partial x}\right)+\frac{\partial\rho\ln c_i}{\partial t}+\frac{\partial\rho\ln c_i\, u}{\partial x} = \frac{w_i}{c_i} \tag{2-35}$$

The terms in the parentheses are recognized as the continuity equation, identically zero. The term $\rho\ln c_i$ is the logarithmic variable π_i. With this definition, equation (2-35) becomes

$$\frac{\partial\pi_i}{\partial t}+\frac{\partial\pi_i u}{\partial x} = \frac{w_i}{c_i} \tag{2-36}$$

from which it can be seen that the chemical source terms for the logarithmic variables are simply the sources for the species densities divided by the species mass fractions,

$$w_{\pi i} = \frac{w_i}{c_i} \tag{2-37}$$

The advantage of the logarithmic species is that exponential variations of species mass fractions become linear variations in the new variables. Wilson found that far fewer grid

points were required to adequately resolve the induction zone in his numerical reproduction of Lehr's experiments when he used the logarithmic species conservation law.

One drawback to this approach is that it does not guarantee conservation of atoms. To remedy this, some of the species conservation equations are replaced with elemental conservation equations. These take a form identical to the original species equations, but describe the conservation of all atoms of a particular element, which are typically distributed among several different chemical species. The mixture density can then be calculated by summing these atom densities. With these modifications, the conserved variables and convective fluxes in equation (2-16) become:

$$Q = J^{-1}y\begin{bmatrix} \rho_1^* \\ \vdots \\ \rho_{nl}^* \\ \pi_{nl+1} \\ \vdots \\ \pi_{ns} \\ \rho u \\ \rho v \\ e \end{bmatrix} \qquad F = J^{-1}y\begin{bmatrix} \rho_1^* U \\ \vdots \\ \rho_{nl}^* U \\ \pi_{nl+1} U \\ \vdots \\ \pi_{ns} U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ U(e+p) \end{bmatrix} \qquad G = J^{-1}y\begin{bmatrix} \rho_1^* V \\ \vdots \\ \rho_{nl}^* V \\ \pi_{nl+1} V \\ \vdots \\ \pi_{ns} V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ V(e+p) \end{bmatrix} \tag{2-38}$$

ρ^* represents the density of the atomic species, and will have no chemical source terms, since the chemical reactions do not alter the number of atoms of a particular element present. nl is the number of atomic species present. Note that the total number of equations is the same in both forms of the equations. Wilson and Sussman also observed that pressure remains a homogeneous function of the conserved variables, so that

$$p = \frac{\partial p}{\partial Q} \cdot Q \tag{2-39}$$

This allows use of conventional flux splitting schemes, which will be discussed in the following chapter.

2.7 Summary

In this work, the axisymmetric Navier-Stokes equations are numerically approximated in generalized curvilinear coordinates. Multiple chemical species with finite-rate chemical reactions are solved using a logarithmic transformation of the species conservations laws. Previously validated chemical reaction models are used to evaluate the numerical algorithm.

Chapter 3: Numerical Methods and Program Features

3.1 Discretization

The Navier-Stokes equations are a set of non-linear partial differential equations for which there is, in general, no closed-form solution. Thus, they must be approximated. The flow is first divided into many small "finite volumes" by defining a computational mesh, or grid, about the body. The equations are then solved in an approximate sense at each individual grid point.

3.1.1 Computational Grid

There are two basic types of computational grids, the structured and the unstructured grid. The simplest form of structured grid is the cartesian grid, where Δx and Δy are uniform throughout. An example of this type of grid was shown in Figure 2-1. A general structured grid can be thought of as a cartesian mesh which is deformed to fit around a body, such as that pictured in Figure 2-2. The advantage of a structured grid is that the grid points are stored in an orderly manner, so that determining a grid point's neighboring points is trivial. As a result, calculations on structured grids tend to be more efficient per grid point than on unstructured grids, especially on vector machines, where structured grid calculations tend to vectorize readily. Structured grids can, however, be very difficult to generate for complex bodies.

Unstructured grids are formed by distributing points about a body, and then connecting them to form triangles or quadrilaterals (or possibly other shapes). The advantage of unstructured grids is that they are very easy to generate. Since a point's neighbors must be

somehow determined and explicitly stored, however, calculations on an unstructured grid take longer per grid point, require more storage, and do not vectorize as readily.

For this work, a structured grid is used. It is formed by specifying points around the boundary of the desired domain, and then solving an elliptic differential equation which smoothly distributes the remaining grid points. This smooth distribution reduces discretization errors. Figure 3-1 shows a coarse grid of the type used.

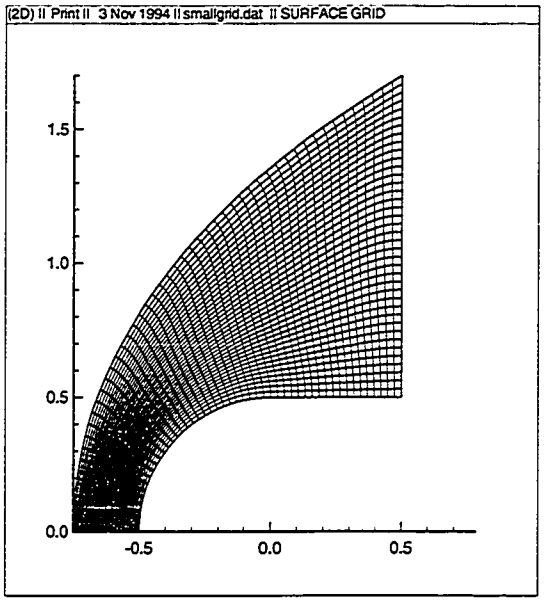

Figure 3-1: A Computational Grid

3.1.2 The Riemann Problem and Godunov's Method

The computational grid divides the flow into a number of small volumes. Each time step, or iteration, the fluxes of conserved quantities at each cell boundary are calculated. The changes in the conserved quantities in each cell are then calculated by summing these fluxes. A significant portion of the effort in developing CFD algorithms has been to determine a suitable method by which to calculate fluxes at the cell interfaces from the values of the conserved quantities within the cells. A common approach is to treat each cell interface as a Riemann problem, and to calculate the fluxes based on the solution to this problem.

A Riemann problem is a situation where a gas initially contains some sort of jump discontinuity. A common example of this is the pressure jump across the diaphragm of a shock tube, but the discontinuity could also be one in density or some other variable. In any case, when the diaphragm is ruptured, any or all of three distinct phenomena may be observed: a shock wave, an expansion fan, and a contact surface. Using the method of characteristics, the time-varying behavior of the gas in a Riemann problem may be completely described.[45] It is, in fact, a self-similar solution, where the properties of the gas are a function of the single variable x/t, where x is the distance from the initial discontinuity, and t is time. Figure 3-2 shows a typical Riemann problem.

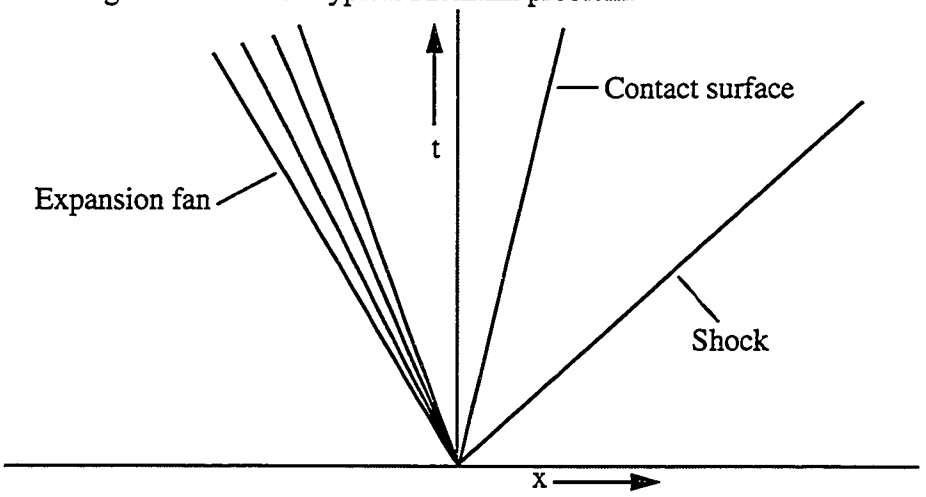

Figure 3-2:A Riemann Problem

An important aspect of the Riemann problem is that although the solution is unsteady in general, it is constant at x=0.[46] Thus, the fluxes of the conserved variables will also be constant at x=0. This fact is the basis of the flux calculations for many CFD algorithms. A first-order Computational Fluid Dynamics (CFD) algorithm treats the fluid as if its properties were uniform throughout each volume in the grid. This assumption of piecewise continuity leads to a Riemann problem at each cell interface. Each time step, the fluxes at every interface are calculated by solving the Riemann problem. Provided the time step is small enough that neighboring Riemann solutions do not interfere, these fluxes remain constant throughout the time step, and the net fluxes of conserved variables across the

boundaries of each cell can be calculated. These net fluxes are then used to update the cell average quantities, and the calculations for the time step are complete, aside from the imposition of boundary conditions at the edges of the domain, and other considerations to be addressed later.

The algorithm just described is commonly referred to as the Godunov Scheme, after the mathematician who introduced it.[47]

3.1.3 Approximate Riemann Solvers

A disadvantage of the Godunov Scheme is that it is computationally expensive, sometimes requiring an iterative process to determine the interface fluxes.[48] The goal of reducing this computational expense has led to a class of algorithms known as Approximate Riemann Solvers, which involve approximate solutions to the Riemann problem, or alternatively, solutions to a problem which approximates the Riemann problem. In these schemes, the interface fluxes can be calculated directly from the flow variables in the neighboring grid cells.

There are two families of Approximate Riemann Solvers, Flux Vector Splitting (FVS, sometimes simply called Flux Splitting), and Flux Difference Splitting (FDS). FVS involves a point-wise splitting of fluxes calculated from cell values. Fluxes are calculated from the cell values, and then split into right- and left-running fluxes, which are applied to the right and left interfaces of the cell. FVS schemes are very computationally efficient, and are widely used. They are generally regarded as inferior to FDS algorithms where viscous solutions are desired,[49] however, because they have had difficulty properly resolving boundary layers, although recent developments in FVS have reduced this disadvantage.[50]

Flux Difference Splitting involves calculating and splitting the difference between the fluxes based on neighboring cell values, and applying the split flux difference to the cell fluxes to determine the interface flux. Among FDS schemes, the two most well-known are those of Roe[51,52] and of Osher.[53] Of the two, Roe's scheme is probably the most widely used, and is readily extensible to multiple gas species.

3.1.4 Roe's Scheme

Roe's scheme replaces the Riemann problem between neighboring grid cells with an approximate linear problem. The original problem, in one dimension,

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} = 0 \tag{3-1}$$

can be written as

$$\frac{\partial Q}{\partial t} + A\frac{\partial Q}{\partial x} = 0 \tag{3-2}$$

with A, the flux Jacobian, defined by $A \equiv \frac{\partial F}{\partial Q}$. Roe's scheme replaces equation (3-2) with an approximate equation,

$$\frac{\partial Q}{\partial t} + \hat{A}\frac{\partial Q}{\partial t} = 0 \tag{3-3}$$

where $\hat{A}$ is now constant between neighboring grid cells. Equation (3-3) can be solved directly, and with a particular definition of $\hat{A}$, gives the following equivalent expressions for the interface flux:

$$\begin{aligned} \tilde{F}_{i+\frac{1}{2}} &= F_i + \Delta F^-_{i+\frac{1}{2}} \\ &= F_{i+1} - \Delta F^+_{i+\frac{1}{2}} \\ &= \frac{1}{2}\left(F_i + F_{i+1} - |\Delta F|_{i+\frac{1}{2}} \right) \end{aligned} \tag{3-4}$$

with the definitions

$$\begin{aligned} \Delta\hat{F}^{\pm}_{i+\frac{1}{2}} &= \hat{A}^{\pm}_{i+\frac{1}{2}} (Q_{i+1} - Q_i) \\ |\Delta\hat{F}| &= \Delta\hat{F}^+ - \Delta\hat{F}^- \\ &= |\hat{A}| (Q_{i+1} - Q_i) \end{aligned} \tag{3-5}$$

The final definition in equation (3-4) allows some flexibility in the definition of $\hat{A}$, and is the form used in this work. $\hat{A}^{\pm}$ represents a splitting of the flux Jacobian matrix based on

its eigenvalues. To calculate $|A|$, $\hat{A}$ is first diagonalized by

$$\hat{A} = \hat{R}\hat{\Lambda}\hat{R}^{-1} \tag{3-6}$$

where $\hat{R}$ contains the right eigenvectors of $\hat{A}$ as its columns, $\hat{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\hat{A}$, and $\hat{R}^{-1}$ contains the left eigenvectors of $\hat{A}$ as its rows. Then

$$\hat{A}^{\pm}_{i+\frac{1}{2}} = \frac{\hat{A}_{i+\frac{1}{2}} + |\hat{A}|_{i+\frac{1}{2}}}{2} \tag{3-7}$$

$$|\hat{A}| = \hat{R}|\hat{\Lambda}|\hat{R}^{-1}$$

where $|\hat{\Lambda}|$ indicates a diagonal matrix of the absolute values of the eigenvalues of $\hat{A}$. The flux Jacobian matrix $\hat{A}$ and its diagonalization are detailed in Appendix B for the logarithmic form of the species conservation law. The fluxes in two dimensions are calculated in an identical fashion, involving the diagonalization of the Jacobian matrices of the fluxes in both directions.

The carets (^) in the above expressions indicate matrices calculated from some symmetric average of the flow variables in the neighboring two grid cells. To calculate $\hat{A}$, Roe defines a unique, density-weighted average as follows:

$$\hat{x}_{i+\frac{1}{2}} = \mu(x) \equiv \frac{\sqrt{\rho_i}x_i + \sqrt{\rho_{i+1}}x_{i+1}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} \tag{3-8}$$

where x represents the flow speeds u and v, and the total enthalpy H. For a single species ideal gas, $\hat{A}$ and its diagonalization can be calculated from these three variables. For multiple species, however, the situation is more complicated because the derivatives of pressure are no longer purely a function of u, v, and H. Various researchers[54-59] have outlined averaging schemes which exhibit some, but not all, of the following properties: They (a) obey Roe's "property U", such that $\Delta F = \hat{A}\Delta Q$, (b) reduce to Roe's scheme for a single species ideal gas, (c) are stable, or (d) are easily calculated. Shinn *et al.*,[60] tested several different averagings, including that of Carofano.[61] and did not note significant differences in the resulting steady-state solutions, although there were some differences in stability. In

this work, the method introduced by Abgrall[62] is used, and is summarized below:

$$\begin{aligned} \hat{u} &= \mu(u) \\ \hat{v} &= \mu(v) \\ \hat{H} &= \mu(H) \\ \hat{c}_i &= \mu(c_i) \\ \hat{T} &= \mu(T) \end{aligned} \tag{3-9}$$

where μ again represents Roe's averaging. Other interface values required are the frozen speed of sound, $\hat{a}$, and the partial derivatives of pressure with respect to species density and total energy, $P_{\rho i}$ and P_e. These are calculated by:

$$\begin{aligned} \hat{R} &= \sum_{i=1}^{ns} \hat{c}_i R_i \\ \hat{c}_p &= \sum_{i=1}^{ns} \hat{c}_i c_{pi}(\hat{T}) \\ \hat{h}_i &= \int_0^{\hat{T}} c_{pi}(T)\, dT + h_i^0 \\ \hat{P}_e &= \frac{\hat{R}}{\hat{c}_p - \hat{R}} \\ \hat{P}_{\rho i} &= (1+\hat{P}_e) R_i \hat{T} + \hat{P}_e \left(\frac{\hat{u}^2 + \hat{v}^2}{2} - \hat{h}_i \right) \\ \hat{a}^2 &= (1+\hat{P}_e) \hat{R} \hat{T} \end{aligned} \tag{3-10}$$

where R, and c_p are the mixture's specific gas constant and specific heat at constant pressure, respectively, h_i is the species enthalpy per unit mass, and h_i^0 is the species heat of formation. Once these variables are calculated, the related quantities for the logarithmic form of the equations are calculated from them, as detailed in Appendices B and C. This method reduces to Roe's averaging for an ideal gas, and exhibits good stability.

In generalized curvilinear coordinates, various metric parameters, such as J^{-1}, must also be calculated at the interface. In a finite difference formulation, used here, these

parameters are calculated with an arithmetic average of the values on either side of the interface, e.g.

$$J^{-1}_{i+\frac{1}{2}} = \frac{1}{2}\left(J_i^{-1} + J_{i+1}^{-1}\right) \tag{3-11}$$

Once the interface fluxes are calculated, a time step is completed by calculating and applying changes to the conserved variables for each cell. Roe's scheme then becomes

$$\frac{\Delta Q_{i,j}}{\Delta\tau} = -\left(\hat{F}_{i+\frac{1}{2}} - \hat{F}_{i-\frac{1}{2}} + \hat{G}_{j+\frac{1}{2}} - \hat{G}_{j-\frac{1}{2}}\right) \tag{3-12}$$

where F represents the Roe fluxes in the ξ direction, and G represents those in the η direction.

3.1.5 The "Entropy Fix"

Roe's scheme replaces the non-linear Riemann problem with a linear approximation. It therefore treats all waves as discontinuities, where the true Riemann solution might involve expansion waves. This approximation is usually quite good (see Leveque[46] for details), but introduces significant error in the particular case of a transonic rarefaction. Since the self-similar Riemann solution for a transonic rarefaction gives an expansion fan which straddles the cell interface, clearly an approximation which treats this wave as a shock, and therefore lying completely on one side of the interface, will be inaccurate. As a result, Roe's scheme allows the existence of expansion shocks, which violate the entropy condition and are therefore non-physical. Thus, a "fix" is required to guarantee a physical solution.

To remove expansion shocks, Yee[63] and others employ eigenvalue smoothing, where the eigenvalues used to calculate $|A|$ at the cell interfaces are not allowed to become smaller in magnitude than a specified value. To enforce this requirement, a smoothing function is defined as follows for eigenvalue λ:

$$\psi(\lambda) = \begin{cases} |\lambda| & |\lambda| > \delta \\ \dfrac{\lambda^2 + \delta^2}{2} & |\lambda| < \delta \end{cases} \tag{3-13}$$

The definition of δ is given in the following section.

3.1.6 The "Carbuncle Phenomenon"

Another artifact of Roe's scheme is the so-called "carbuncle phenomenon," characterized by the distortion of a bow shock, usually near the stagnation line. Although it is not well understood, it can frequently be eliminated with the eigenvalue smoothing parameter δ. In this work, Yee's[64] expression for δ is modified following the approach of Imlay *et al.*[65]:

$$\delta = (\varepsilon_1 + \varepsilon_2|\Delta\nabla p| + \varepsilon_3|\Delta\nabla T|)\left[\frac{U+V+a\left(\sqrt{\xi_x^2+\xi_y^2}+\sqrt{\eta_x^2+\eta_y^2}\right)}{2}\right]$$

$$\Delta\nabla p = \frac{p_{i+\frac{3}{2}} - 2p_{i+\frac{1}{2}} + p_{i-\frac{1}{2}}}{p_{i+\frac{3}{2}} + 2p_{i+\frac{1}{2}} + p_{i-\frac{1}{2}}} \tag{3-14}$$

$$\Delta\nabla T = \frac{T_{i+\frac{3}{2}} - 2T_{i+\frac{1}{2}} + T_{i-\frac{1}{2}}}{T_{i+\frac{3}{2}} + 2T_{i+\frac{1}{2}} + T_{i-\frac{1}{2}}}$$

With ε_2 and ε_3 set to zero, the above becomes Yee's expression. This expression for δ adds a small amount of dissipation to the algorithm, but has a minimal effect on the flow solution, aside from eliminating the carbuncle and preventing expansion shocks. ε_1 is given a value between 0.1 and 0.3, and ε_2 and ε_3 are typically set to unity. The terms $\Delta\nabla p$ and $\Delta\nabla T$ are near zero everywhere except near shocks and combustion fronts, where they augment the smoothing parameter somewhat.

3.1.7 Extension to Second-Order Spatial Accuracy

The preceding algorithm is first-order accurate, that is, the leading error terms are multiples of Δt, Δx, and Δy. Second-order accuracy is desirable to allow more accurate solutions on coarse grids, and more rapid convergence to the exact solution as the grid is refined. Harten, Yee, and Shin, and others have developed various schemes by which the conservation equations may be approximated by second-order central differences in smooth regions of the flow, while reverting to Roe's scheme near discontinuities.[63,64,66-70] These schemes are of a class called Total Variation Diminishing (TVD). Actually, they are

only TVD in one dimension (indeed, it has been shown that TVD schemes in more than one dimension can be at most first-order accurate[46]), but TVD schemes are well-behaved when applied to multiple dimensions, even though they lose their TVD property.

The TVD scheme selected for this work is the Symmetric TVD scheme of Yee,[70] which is second-order accurate in smooth regions of the flow, is relatively robust, and is computationally efficient. The interface flux is defined as follows:

$$\hat{F}_{i+\frac{1}{2},j} = \frac{1}{2}\left(F_{i,j} + F_{i+1,j} - \hat{R}\phi_{i+\frac{1}{2},j} \right) \tag{3-15}$$

where ϕ is defined as:

$$\phi_{i+\frac{1}{2},j} = \psi\left(|\Lambda|_{i+\frac{1}{2},j} \right)\left(\alpha_{i+\frac{1}{2},j} - minmod\left(\alpha_{i-\frac{1}{2},j}, \alpha_{i+\frac{1}{2},j}, \alpha_{i+\frac{3}{2},j} \right) \right)$$

$$\alpha_{i+\frac{1}{2},j} = \hat{R}^{-1}_{i+\frac{1}{2},j}\left(\frac{(JQ)_{i+1} - (JQ)_i}{\frac{1}{2}(J_{i+1,j} + J_{i,j})} \right) \tag{3-16}$$

To achieve monotone solutions near shocks, it is necessary for the scheme to revert to first-order in the vicinity of the shock. The minmod function above is equal to zero if the three values of α are not all of the same sign, and is otherwise set to the minimum value in magnitude of the three. In smooth regions of the flow, α will be approximately constant, and the above expression will become a central difference. Near discontinuities or rapid changes in the flow, the minmod function will return a small value or zero, and equation (3-16) will revert to Roe's first-order scheme.

Yee *et al.*[64] observed that the values of α in equation (3-16) are proportional to changes in density when the standard definition of R^{-1} is used. Since the density ratio approaches a limiting value for shocks of increasing strength, They recommend that R^{-1} be multiplied by the sound speed, and that R be divided by it. This modification is allowable because the right and left eigenvectors of a matrix are not unique, so the resultant matrices still diagonalize A. The modified R^{-1} results in values of α proportional to pressure changes, which allows the above algorithm to detect shocks better in hypersonic flow.

3.1.8 An Approach to Efficient Calculations

Much of the computational expense in the present method is involved in calculating the convective fluxes at the cell interfaces and the chemical sources and Jacobians thereof. It is advantageous, therefore, to examine ways of maximizing the efficiency of these calculations. In this section, details of the calculations for Roe's scheme are given.

Roe's scheme and the second-order methods based on it involve matrix-vector multiplications for each grid point. In general, the computational expense of such operations scales with the number of equations squared. This turns out not to be the case, however, when multiple species are involved. The following outlines an approach which scales linearly with the number of equations in the limit of a large number of species. The approach is shown here for the logarithmic form of the species conservations equations, but nearly identical expressions apply to the standard (species density) formulation. First, α is calculated for each grid point from

$$\alpha = \begin{bmatrix} \hat{a}^2\Delta\rho_1^* - \frac{\rho_1^*}{\rho}\hat{\Delta p} \\ \vdots \\ \hat{a}^2\Delta\rho_{nl}^* - \frac{\rho_{nl}^*}{\rho}\hat{\Delta p} \\ \hat{a}^2\Delta\pi_{nl+1} - \frac{\pi_{nl+1}}{\rho}\hat{\Delta p} \\ \vdots \\ \hat{a}^2\Delta\pi_{ns} - \frac{\pi_{ns}}{\rho}\hat{\Delta p} \\ \xi'_y\rho\hat{\Delta u} - \xi'_x\rho\hat{\Delta v} \\ \frac{1}{2}\left(\hat{\Delta p} + \hat{a}\left(\xi'_x\rho\hat{\Delta u} + \xi'_y\rho\hat{\Delta v}\right)\right) \\ \frac{1}{2}\left(\hat{\Delta p} - \hat{a}\left(\xi'_x\rho\hat{\Delta u} + \xi'_y\rho\hat{\Delta v}\right)\right) \end{bmatrix} \tag{3-17}$$

with the definitions

$$\hat{\Delta p} \equiv \sum_{i=1}^{ns+3} \frac{\hat{\partial p}}{\partial q_i} \Delta q_i \qquad \rho\hat{\Delta} u \equiv \Delta\rho u - \hat{u}\Delta\rho \qquad \rho\hat{\Delta} v \equiv \Delta\rho v - \hat{v}\Delta\rho \tag{3-18}$$

where Δq_i, $\Delta\rho u$ and $\Delta\rho v$ are spatial differences in the conserved variables, e.g., $\Delta\rho u = (\rho u)_{i+1} - (\rho u)_i$, and $\Delta\rho = \sum \Delta\rho_i^*$. Note that the above terms include the factor of $\hat{a}^2$, as recommended by Yee. Then the elements of the vector Φ are calculated from

$$\phi_m = \Psi(\lambda_m)\left(\alpha_m - \hat{Q}_m\right) \tag{3-19}$$

where $\Psi(\lambda_m)$ represents the smoothed eigenvalues and $\hat{Q}_m$ is calculated by

$$\hat{Q}_{m,\,i+\frac{1}{2}} = minmod\left(\alpha_{i-\frac{1}{2}}, \alpha_{i+\frac{1}{2}}, \alpha_{i+\frac{3}{2}}\right) \tag{3-20}$$

The upwind correction to the interface flux is then calculated from

$$(R\Phi)_{i+\frac{1}{2}} = \frac{1}{\hat{a}^2}\begin{bmatrix} \phi_1 + \frac{\hat{\rho}_1^*}{\rho}K_1 \\ \vdots \\ \phi_{nl} + \frac{\hat{\rho}_{nl}^*}{\rho}K_1 \\ \phi_{nl+1} + \frac{\hat{\pi}_{nl+1}}{\rho}K_1 \\ \vdots \\ \phi_{ns} + \frac{\hat{\pi}_{ns}}{\rho}K_1 \\ \hat{u}K_2 + \xi'_y\phi_{ns+1} + \xi'_x\hat{a}K_3 \\ \hat{v}K_2 - \xi'_x\phi_{ns+2} + \xi'_y\hat{a}K_3 \\ \hat{H}K_2 + \hat{U}\hat{a}K_3 + K_4 \end{bmatrix} \tag{3-21}$$

with the definitions

$$
\begin{aligned}
K_1 &\equiv \phi_{ns+2} + \phi_{ns+3} \\
K_2 &\equiv \sum_{i=1}^{nl} \phi_i + \phi_{ns+2} + \phi_{ns+3} \\
K_3 &\equiv \phi_{ns+2} - \phi_{ns+3} \\
K_4 &\equiv -\frac{1}{p_e}\left(\sum_{i=1}^{nl} \hat{a}_i^{*2} \phi_i + \sum_{i=nl+1}^{ns} p_{\pi i} \phi_i \right) + \left(\xi'_y \hat{u} - \xi'_x \hat{v}\right) \phi_{ns+1} \\
\hat{a}_i^{*2} &\equiv p_{\rho i^*} + p_e \left(\hat{H} - \hat{u}^2 - \hat{v}^2\right) \\
\xi'_x &\equiv \frac{\xi_x}{\sqrt{\xi_x^2 + \xi_y^2}} \\
\xi'_y &\equiv \frac{\xi_y}{\sqrt{\xi_x^2 + \xi_y^2}} \\
\hat{U} &\equiv \xi'_x \hat{u} + \xi'_y \hat{v}
\end{aligned}
\tag{3-22}
$$

Finally, the interface fluxes are calculated from

$$
\tilde{F}_{i+\frac{1}{2}} = \frac{1}{2}\left(F_i + F_{i+1} - (R\Phi)_{i+\frac{1}{2}} \right) \tag{3-23}
$$

Examination of the above approach reveals that the computational effort scales linearly with the number of equations, in the limit of a large number of chemical species.

3.1.9 Efficient Calculation of Chemical Source Terms

Equation (2-26) showed the general form for calculating chemical source terms. For each reaction, two summations over all species are required. In both of these summations, each species is raised to an integral power. With increasing numbers of reactions and species, this process becomes very expensive, motivating a more efficient approach. To maximize the efficiency of the chemical source calculations, the approach of Pratt and Wormeck[71] is employed. They recognized that most chemical reactions involve at most six chemical species, and frequently only three, so that the exponents for many species are zero for a

given reaction, and can be eliminated from the expression. They accomplish this by classifying reactions according to how many species are involved. For each type of reaction, only the involved species are used in calculating the source terms. This approach also replaces the exponentiations with multiplications, as will be shown below. Four types of chemical reactions are considered:

$$\begin{aligned} &1.\ A + B(\ \ + C) \leftrightarrow D + E(\ \ + F) \\ &2.\ A + M \leftrightarrow B + C + M \\ &3.\ A + B + M \leftrightarrow C + M \\ &4.\ A + M \leftrightarrow B + M \end{aligned} \tag{3-24}$$

where M represents the sum of all species, and other terms represent individual chemical species. The parentheses in reaction type 1 indicate that the enclosed terms are optional.

To see the advantage of this approach, consider the reaction

$$2OH \leftrightarrow H_2O + O \tag{3-25}$$

with a forward reaction constant k_f, and a backward reaction constant k_b. This is classified as reaction type 1, with A and B both equal to OH. The source term for OH is

$$w_{OH} = -k_f[X_{OH}]\ [X_{OH}] + k_b\left[X_{H_2O}\right]\ [X_O] \tag{3-26}$$

If equation (2-26) were applied, $[X_{OH}]$ would be squared and multiplied by all other species raised to the zeroth power (i.e. unity) for the first term on the left-hand side. Here, all of these exponentiations are replaced by a single multiplication, saving significant computational expense.

3.1.10 Geometric Conservation Law Terms for Axisymmetry

In §2.2, certain groups of grid metric-related terms, the invariants of the coordinate transformation, were held to be identically zero, and were cancelled out to yield the conservation equations in generalized curvilinear coordinates. Discretization of those equations, however, replaced their partial derivatives with finite differences, as a result of which, the invariants can not be assumed to be zero, and in fact, may not be. The effect of the resulting errors is usually small, and can be neglected. In the axisymmetric formulation, however, these errors become significant near the line of symmetry, resulting in the algorithm

not preserving uniform flow. Following the approach of Gielda and McRae,[72] this is corrected by analytically applying the algorithm to uniform flow, noting the terms which do not cancel out, and subtracting these terms from the right-hand side. This is related to the identity called the geometric conservation law by Thomas and Lombard.[73] With the definitions

$$\begin{aligned} g_x &= \delta_\xi J^{-1} y \xi_x + \delta_\eta J^{-1} y \eta_x \\ g_y^1 &= y\left(\delta_\xi J^{-1} \xi_y + \delta_\eta J^{-1} \eta_y \right) \\ g_y^2 &= \delta_\xi J^{-1} y \xi_y + \delta_\eta J^{-1} y \eta_y \end{aligned} \tag{3-27}$$

where δ indicates a central difference, the terms to be subtracted from the right-hand side become

$$\begin{gathered} \rho_1^*\left(u g_x + v g_y^1 \right) \\ \vdots \\ \pi_{ns}\left(u g_x + v g_y^1 \right) \\ \left(\rho u^2 + p \right) g_x + \rho u v g_y^1 \\ y\left(\rho u v g_x + \rho v^2 g_y^1 \right) + \left(g_y^2 - J^{-1} \right) p \\ u(e+p) g_x + v(e+p) g_y^1 \end{gathered} \tag{3-28}$$

For two-dimensional flow, y is set to unity in the above expressions.

In practice, these terms have a negligible effect on the flow except on the first one or two grid lines away from the line of axisymmetry, where they eliminate undesirable oscillations.

3.1.11 Boundary Conditions

To properly solve for a given flow, it is extremely important to treat the boundaries correctly. Roughly speaking, flow variables at a boundary must be calculated from information taken from upstream of that boundary. More accurately, the direction of wave propagation, given by the eigenvalues near the boundary, determines what and how many pieces of information must be taken from each side. If the flow is supersonic to the right,

for example, then all waves flow to the right, and all information must be taken from left of the boundary. In subsonic flow, there are eigenvalues going in either direction, and the right amount of information must be taken from either side of the boundary.

In the problems considered here, all boundaries are very simple except for the body surface. The inflow boundary is supersonic everywhere, and is set to free stream conditions. The outflow boundary is also supersonic, so the solution is extrapolated from the interior of the flow regime to the boundary. For viscous flow this is still justified, even though the flow inside the boundary layer becomes subsonic.[74] At the line of symmetry, the grid collapses and there is no flux across it, so no conditions need be specified there at all. The surface of the body requires more careful consideration.

For inviscid flow, the velocity at the wall is calculated by extrapolating the tangential velocity from the interior, and setting the normal velocity to zero at the wall. For an $\eta = const$ boundary, the tangential velocity is calculated by

$$V_t = \frac{\eta_y u - \eta_x v}{\sqrt{\eta_x^2 + \eta_y^2}} \tag{3-29}$$

V_t is calculated from the flow and metric terms near the wall. u and v are then backed out by evaluating the following expressions using metric terms at the wall:

$$\begin{aligned} u &= \frac{\eta_y V_t}{\sqrt{\eta_x^2 + \eta_y^2}} \\ v &= \frac{-\eta_x V_t}{\sqrt{\eta_x^2 + \eta_y^2}} \end{aligned} \tag{3-30}$$

For viscous flow, the velocity is set to zero at the wall.

Species mass fractions are calculated by assuming zero gradient at the wall, equivalent to assuming a non-catalytic wall. For an $\eta = const$ wall this gives the following expression for species i:

$$(\eta_x \xi_x + \eta_y \xi_y)\, c_{i_\xi} + (\eta_x^2 + \eta_y^2)\, c_{i_\eta} = 0 \tag{3-31}$$

Equation (3-31) contains derivatives for the species mass fraction in both the ξ and η directions. The derivative in the η direction (normal to the wall) is approximated by a

one-sided finite difference, while the derivative in the ξ direction may be approximated using a one-sided difference or a central difference. A one-sided upwind difference was used in this work, because it gives more stable results for flows with a clearly identifiable upwind direction. Equation (3-31) is set up and inverted for each species mass fraction at the wall.

For inviscid flows, the wall is assumed to be adiabatic (ideally insulated), giving a zero temperature gradient condition at the wall. This assumption results in an expression for temperature identical to that for species mass fractions. For viscous flows, the insulated wall assumption may also be used, but an isothermal wall option is added, allowing a uniform wall temperature to be specified.

For both inviscid and viscous flows, pressure is calculated by solving the normal momentum equation. The form used here is derived from the conservative form recommended by Hoffman and Chiang:[75]

$$\begin{aligned}&\eta_x\left(J^{-1}\rho uU\right)_\xi+\eta_y\left(J^{-1}\rho vU\right)_\xi+\eta_x\left(J^{-1}\rho uV\right)_\eta+\eta_y\left(J^{-1}\rho vV\right)_\eta\\&+\eta_x\left(J^{-1}\xi_x p\right)_\xi+\eta_x\left(J^{-1}\eta_x p\right)_\eta+\eta_y\left(J^{-1}\xi_y p\right)_\xi+\eta_y\left(J^{-1}\eta_y p\right)_\eta=0\end{aligned} \tag{3-32}$$

Note that equation (3-32) requires values for density at the wall. Conventionally, the value of density calculated from the previous time step would be used. This technique can impair stability, however. Instead, density is eliminated from the expression by noting that it can be replaced using the perfect gas law,

$$\rho=\frac{P}{RT} \tag{3-33}$$

Since T has already been calculated, R may be calculated from the species mass fractions, and all other terms aside from pressure are already known, this reduces equation (3-32) to an equation for pressure only:

$$\begin{aligned}&\eta_x\left(J^{-1}\left(\xi_x+\frac{uU}{RT}\right)P\right)_\xi+\eta_y\left(J^{-1}\left(\xi_y+\frac{vU}{RT}\right)P\right)_\xi\\&+\eta_x\left(J^{-1}\left(\eta_x+\frac{uV}{RT}\right)P\right)_\eta+\eta_y\left(J^{-1}\left(\eta_y+\frac{vV}{RT}\right)\right)_\eta=0\end{aligned} \tag{3-34}$$

The ξ and η derivatives in the above expression are approximated with finite differences

as with temperature and species mass fractions, and the resulting algebraic expression is inverted to yield pressure at the wall. The two thermodynamic variables, T and P, the species mass fractions, c_i, and the flow speeds, u and v, are then used to calculate all of the conserved variables at the wall.

3.2 Implicit Schemes

3.2.1 Point-Implicit Method

The chemical source terms, represented by W in equation (2-16), are usually very stiff, meaning that the time scales required for relaxation of the chemical reactions can be very much shorter or longer than the characteristic time scales of the fluid flow. The result of this stiffness is an extreme time step limitation for explicit schemes.

An explicit scheme is a scheme where ΔQ is based only on information at the current time level. In other words, for

$$\Delta Q \equiv Q^{n+1} - Q^n = (-R + H + W)\,\Delta\tau \tag{3-35}$$

where R represents the flux differences in equation (2-16), all terms in R, H and W are calculated from Q^n. In an implicit scheme, some or all of the terms in on the right-hand side are calculated from Q^{n+1}, where Q^{n+1} represents the next time level to be calculated. A common implicit scheme in chemically reacting flow is the so-called point-implicit scheme, introduced by Bussing and Murman.[76]

The point-implicit scheme is an explicit scheme modified to treat only the chemical reactions implicitly. It takes the form

$$\frac{\Delta Q}{\Delta\tau} = -R^n + H^n + W^{n+1} \tag{3-36}$$

Since Q^{n+1} is not yet known, W^{n+1} is approximated as follows:

$$W^{n+1} \approx W^n + \left.\frac{\partial W}{\partial Q}\right|^n (Q^{n+1} - Q^n) = W^n + Z^n \Delta Q \tag{3-37}$$

The final term above is taken to the left-hand side, giving the following:

$$\left(\frac{I}{\Delta\tau} - Z^n\right)\Delta Q = -R^n + H^n + W^n \tag{3-38}$$

Equation (3-38) is the point-implicit scheme, and requires that the matrix $\left(\frac{I}{\Delta\tau} - Z^n\right)$ be inverted at each grid point to solve for the change to the conserved variables at that point. Z^n is the Jacobian of the chemical source terms with respect to the conserved variables. Neglecting nondimensionalization, it is calculated as follows for the species density formulation:

$$\begin{aligned} Z_{m,n} &= \frac{\partial w_m}{\partial q_n} = \frac{\partial}{\partial q_n}\left(M_m \sum_{j=1}^{nr} (\nu''_{m,j} - \nu'_{m,j}) \left[k_{f,j} \prod_{i=1}^{n} [X_i]^{\nu_{i,j}} - k_{b,j} \prod_{i=1}^{n} [X_i]^{\nu_{i,j}} \right] \right) \\ &= \frac{M_m}{M_n} \frac{\partial}{\partial [X_n]} \sum_{j=1}^{nr} (\nu''_{m,j} - \nu'_{m,j}) \left[k_{f,j} \prod_{i=1}^{n} [X_i]^{\nu_{i,j}} - k_{b,j} \prod_{i=1}^{n} [X_i]^{\nu_{i,j}} \right] \\ &= \frac{M_m}{M_n} \sum_{j=1}^{nr} (\nu''_{m,j} - \nu'_{m,j}) \left[\left(\frac{k_{f,j}}{[X_n]} + \frac{1}{k_{f,j}} \frac{\partial k_{f,j}}{\partial [X_n]} \right) \prod_{i=1}^{n} [X_i]^{\nu_{i,j}} \right. \\ &\qquad \left. - \left(\frac{k_{b,j}}{[X_n]} + \frac{1}{k_{b,j}} \frac{\partial k_{b,j}}{\partial [X_n]} \right) \prod_{i=1}^{n} [X_i]^{\nu_{i,j}} \right] \end{aligned} \tag{3-39}$$

The calculation for the logarithmic species formulation is given in the following section. The terms $\frac{\partial k_{f,j}}{\partial [X_n]}$ and $\frac{\partial k_{b,j}}{\partial [X_n]}$ exist because the reaction constants are functions of temperature, which is, in turn, a function of the conserved variables. In practice, these terms are commonly neglected, and are neglected in this work.

3.2.2 The Chemical Source Jacobian for the Logarithmic Species Formulation

To calculate the chemical sources and source Jacobian matrix for the logarithmic species variables, these quantities are first calculated for the species densities, and then converted

to the new variable set by:

$$w_i = \frac{w_i'}{c_i}$$
$$Z_{m,n} = \frac{\partial w_m}{\partial q'_k}\frac{\partial q'_k}{\partial q_n} \tag{3-40}$$

where the repeated index k indicates summation, and the prime (') indicates the old (species density) variable set. First, it is noted that

$$\frac{\partial w_m}{\partial q'_k} = \frac{\partial}{\partial q'_k}\left[\frac{w'_m}{c_m}\right] = \frac{\partial}{\partial q'_k}\left[\frac{w'_m}{\rho_m}\sum_{s=1}^{ns}\rho_s\right]$$
$$= \begin{cases} \frac{1}{c_m}\left[\frac{\partial w'_m}{\partial \rho_k} + \frac{w'_m}{\rho}\left(1 - \frac{\delta_{mk}}{c_m}\right)\right], & k \le ns \\ \frac{1}{c_m}\frac{\partial w'_m}{\partial q'_k}, & k > ns \end{cases} \tag{3-41}$$

where $\frac{\partial w'_m}{\partial q'_k}$ is recognized as the chemical source Jacobian matrix in the old variable set, represented as $Z'_{m,n}$. $\frac{\partial q'_k}{\partial q_n}$ must then be calculated by expressing the old variables as functions of the new ones, and taking the Jacobian of the result. First, density is calculated by summing the atomic densities:

$$\rho = \sum_{i=1}^{nl}\rho_i^* \tag{3-42}$$

For species represented by a logarithmic variable, the species density is then calculated from

$$\rho_i = \rho \exp\left(\frac{\pi_i}{\rho}\right) \tag{3-43}$$

Calculating the remaining species densities is more involved. In general, it is necessary to invert a matrix of the same order as the number of atomic elements present. First, it is

noted that the molar concentration of atomic species k can be calculated from

$$[X_k] = \frac{\rho_k^*}{M_k^*} = \sum_{s=1}^{ns} n_{ks}[X_s] = \sum_{s=1}^{ns} n_{ks}\frac{\rho_s}{M_s} \tag{3-44}$$

where M_k^* is the atomic weight of element k, and $n_{k,s}$ is equal to the number of atoms of element k in species s. Since the densities of some species can be calculated from equation (3-43), the final term in equation (3-44) is divided up into known and unknown species densities. All unknown quantities are taken to the left-hand side, and the resulting equation is multiplied by M_k^*, giving

$$\sum_{s=1}^{nl} n_{ks}\frac{M_k^*}{M_s}\rho_s = \rho_k^* - M_k^* \sum_{s=nl+1}^{ns} \frac{n_{ks}}{M_s}\rho_s \tag{3-45}$$

This equation is written for each element k, resulting in the following matrix equation for the remaining species densities:

$$\begin{bmatrix} n_{11}\frac{M_1^*}{M_1} & n_{12}\frac{M_1^*}{M_2} & \cdots & n_{1\,nl}\frac{M_1^*}{M_{nl}} \\ \vdots & & & \vdots \\ n_{nl\,1}\frac{M_{nl}^*}{M_1} & n_{nl\,2}\frac{M_{nl}^*}{M_2} & \cdots & n_{nl\,nl}\frac{M_{nl}^*}{M_{nl}} \end{bmatrix} \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{nl} \end{bmatrix} = \begin{bmatrix} \rho_1^* - M_1^* \sum_{s=nl+1}^{ns} \frac{n_{1s}}{M_s}\rho_s \\ \vdots \\ \rho_{nl}^* - M_{nl}^* \sum_{s=nl+1}^{ns} \frac{n_{nl\,s}}{M_s}\rho_s \end{bmatrix} \tag{3-46}$$

The matrix on the left-hand side of equation (3-46) is a constant matrix, so it can be inverted once and stored. If each of the unrepresented species (species not represented by a logarithmic variable) consists of only the corresponding atomic element, this matrix becomes the identity matrix, and the procedure is simplified.

The momentum and total energy are the same in both forms of the equations. Finally, the Jacobian matrix $\frac{\partial Q'}{\partial Q}$ is calculated from these expressions, and applied to equation (3-40). This Jacobian matrix is presented in Appendix C.

3.2.3 Steger-Warming Implicit Formulation

The point-implicit scheme treats only the chemical source terms implicitly. Many schemes, however, treat all terms in the right-hand side implicitly. In this section, the first-order Steger-Warming[77] algorithm will be put into implicit form. Although the Steger-Warming algorithm is not used in this work, this will serve to introduce the actual implicit algorithm used, which will be developed in the following two sections. In one dimension with no source terms, the Steger-Warming algorithm is given by

$$\frac{\Delta Q}{\Delta t} = -\frac{1}{\Delta x}\left(F^-_{i+1} - F^-_i + F^+_i - F^+_{i-1}\right) \tag{3-47}$$

with the definitions

$$\begin{aligned} F^+ &= A^+ Q \\ F^- &= A^- Q \\ A^\pm &= R\left(\frac{\Lambda \pm |\Lambda|}{2}\right)R^{-1} \end{aligned} \tag{3-48}$$

An implicit formulation advances all terms on the right-hand side of equation (3-48) to time level $n+1$ using a linearization similar to that in the point-implicit algorithm, for example

$$F^{+^{n+1}} \approx F^{+^n} + \left.\frac{\partial F^+}{\partial Q}\right|^n \Delta Q \approx F^{+^n} + A^{+^n}\Delta Q \tag{3-49}$$

Taking all ΔQ terms to the left-hand side gives

$$-A^+_{i-1}\Delta Q_{i-1} + \left(\frac{\Delta x}{\Delta t}I + |A|_i\right)\Delta Q_i + A^-_{i+1}\Delta Q_{i+1} = -R_i^{\,n} \tag{3-50}$$

Equation (3-50) is one line of a matrix equation; the matrix is formed by writing this equation for each point in the grid, and collecting all ΔQ terms into a vector. For non-reacting flow in two dimensions, this equation becomes (in generalized curvilinear coordinates)

$$\begin{aligned} -B^+_{i,j-1}\Delta Q_{i,j-1} - A^+_{i-1,j}\Delta Q_{i-1,j} + \left(\frac{I}{\Delta\tau} + |A|_{i,j} + |B|_{i,j}\right)\Delta Q_{i,j} \\ + A^-_{i+1,j}\Delta Q_{i+1,j} + B^-_{i,j+1}\Delta Q_{i,j+1} = -R_{i,j}^{\,n} \end{aligned} \tag{3-51}$$

Equation (3-51) represents a block penta-diagonal matrix equation, which is computationally expensive to invert directly by Gaussian elimination or related methods. One way to avoid this expense is to approximately invert the equation. A common approximate inversion technique is the Gauss-Seidel algorithm.

3.2.4 The Symmetric Gauss-Seidel Algorithm

The Gauss-Seidel algorithm for a general matrix equation $Ax = b$ expresses the matrix A as the sum of three matrices, the diagonal, and the lower and upper triangular portions of A, i.e.,

$$A = D - L - U \tag{3-52}$$

The original equation is then approximately inverted by taking either L or U to the right-hand side, multiplied by the initial guess of x. The remaining portion of A is then either lower or upper triangular, and can be inverted very cheaply to produce a refined estimate of x. For example, taking U to the right-hand side gives

$$(D - L)x^1 = b + Ux^0 \tag{3-53}$$

This process can then be repeated with L taken to the right-hand side, giving

$$(D - U)x^2 = b + Lx^1 \tag{3-54}$$

Equations (3-53) and (3-54) together are the Symmetric Gauss-Seidel (SGS) algorithm. This sequence can be repeated to successively refine the values of x. It will converge only if A is diagonally dominant, meaning that the diagonal elements of A must be greater in magnitude than the sum of the rest of the elements on each row.

The SGS algorithm can be applied to equation (3-51) in a block-wise manner. Rather than defining D to be just the scalar diagonal of the implicit operator, it is defined to be the block diagonal, giving

$$D = \frac{I}{\Delta\tau} + |A|_{i,j} + |B|_{i,j} \tag{3-55}$$

Performing the SGS relaxation on the implicit operator, then, requires the inversion of the block diagonal. This is much cheaper than inverting the entire implicit operator, and can

save much computational expense. The expense can still be considerable, however, a fact which motivates the LU-SGS algorithm, developed in the next section.

3.2.5 The LU-SGS Algorithm

To develop the LU-SGS algorithm, the SGS algorithm in equations (3-53) and (3-54) will first be expressed as an approximate lower-upper (LU) factorization of the original matrix A (which here signifies a general matrix, not to be confused with the flux Jacobian matrix). If x^0 is taken as zero, equation (3-53) becomes

$$(D-L)\,x^1 = b \tag{3-56}$$

Subtracting this from equation (3-54) gives

$$(D-U)\,x^2 = Dx^1 \tag{3-57}$$

Solving for x^1 and applying the result to equation (3-56) gives

$$(D-L)\,D^{-1}\,(D-U)\,x^2 = b \tag{3-58}$$

which can be viewed as an approximate LU factorization of the original matrix. The claim was previously made that the SGS algorithm will converge only for diagonally dominant matrices. This fact can be seen if the matrices in equation (3-58) are multiplied out. This gives

$$\left(D-L-U-LD^{-1}U\right)x^2 = \left(A-LD^{-1}U\right)x^2 = b \tag{3-59}$$

from which it can be seen that an error term of $LD^{-1}U$ is introduced. When more than one SGS iteration is performed, the expression becomes more complicated, but clearly a single SGS iteration will give a good approximation to the inversion of the original equation if the diagonal terms dominate the matrix.

The LU-SGS algorithm of Yoon and Jameson[78] is formed by performing an SGS iteration on the implicit operator in equation (3-51), and making the following approximations:

$$A^{\pm} \approx \frac{1}{2}(A \pm r_A I) \qquad B^{\pm} \approx \frac{1}{2}(B \pm r_B I)$$
$$|A| \approx r_A I \qquad |B| \approx r_B I \tag{3-60}$$

r_A and r_B are larger than the spectral radii of the flux Jacobian matrices A and B. The spectral radii are equal to the magnitude of the largest eigenvalues, and are known analytically. The diagonal block in equation (3-51) now becomes itself a diagonal matrix. That is,

$$\frac{I}{\Delta\tau} + |A|_{i,j} + |B|_{i,j} \rightarrow \left(\frac{1}{\Delta\tau} + r_A + r_B\right) I \tag{3-61}$$

Inversion of this diagonal is therefore trivial, and the major portion of the computational effort required to perform an SGS iteration on the implicit operator has been eliminated. In addition, it is diagonally dominant, ensuring that the SGS algorithm will produce a good approximate inversion. The LU-SGS algorithm, expressed as a lower matrix inversion followed by an upper matrix inversion, becomes (expressed in matrix form)

$$\left[-\frac{1}{2}(B + r_B I)_{i,j-1} \qquad -\frac{1}{2}(A + r_A I)_{i-1,j} \qquad (1 + r_A + r_B) I\right] \Delta Q^1 = -R^n$$
$$\left[(1 + r_A + r_B) I \qquad \frac{1}{2}(A - r_A I)_{i+1,j} \qquad \frac{1}{2}(B - r_B I)_{i,j+1}\right] \Delta Q^2 \tag{3-62}$$
$$= (1 + r_A + r_B) \Delta Q^1$$

To improve spatial accuracy, R in the above expression is usually replaced by a more accurate spatial difference than the Steger-Warming differencing shown in equation (3-47). The previously introduced Symmetric TVD algorithm is used in this work.

It should be noted that the approximations made to the flux Jacobian matrices above, while dramatically reducing the computational effort per iteration, also slow the steady-state convergence rate per iteration. This is a trade-off which must be considered when selecting an implicit algorithm. The author's experience indicates, however, that the benefits of this scheme outweigh the disadvantages in a wide variety of problems.

3.2.6 The Diagonal Approximate Chemical Source Jacobian

The LU-SGS algorithm is an extremely efficient algorithm for the computation of non-reacting flow. When chemical reactions are introduced, however, a complication arises. For stability, it is necessary to treat the chemical source terms implicitly. Yet if this is done, as was introduced with the point-implicit method, the diagonal blocks in equation (3-62) are no longer diagonal, because they have the full source Jacobian matrices (Z) added to them. This motivated the development of the diagonal approximation to the source term Jacobian by Eberhardt, and Imlay.[24]

Eberhardt's algorithm replaces the full chemical source Jacobian with a diagonal matrix, where each diagonal term represents a characteristic rate associated with the chemical reactions which produce and destroy the corresponding chemical species. The original formulation was to take the L2-norm of each row of the source Jacobian, place the result on the diagonal, and zero the rest of the row. With this modification, the diagonal blocks once again become purely diagonal, and the LU-SGS algorithm can be used with chemical reacting flow while maintaining the extremely cheap inversion described above.

The LU-SGS algorithm with Eberhardt's approximate source Jacobian is extremely efficient per iteration. Some researchers, however, have reported difficulty in achieving convergence to steady-state or unacceptably slow convergence.[26,79] Candler's solution to this problem was to replace some of the species continuity equations with elemental conservation equations, similar to what was described in §2.6, although retaining the remaining species density equations. This has the advantage of conserving elements, and resulted in a more stable and rapidly-converging algorithm.

Through numerical experimentation, Imlay[80] arrived at another form for the approximate source Jacobian which, in the author's experience, appears to produce results similar to those of Candler. Imlay noted that, neglecting the derivatives of the reaction constants with respect to temperature, the source Jacobian can be expressed as

$$Z_{m,n} = \frac{\partial \dot{\rho}_m}{\partial \rho_n} = \frac{M_m}{M_n} \frac{\partial [\dot{X}_m]}{\partial [X_n]} \tag{3-63}$$

where $\dot{\rho}_m$ and $\dot{X}_m$ indicate the production of these quantities by chemical reactions. Not-

ing that the L2-norm of each row is placed on the diagonal, where the ratio of molecular weights disappears, Imlay removed all molecular weight ratios from the Jacobian, and took the L2-norm of the result. Thus, his expression for the approximate source Jacobian is given by

$$\tilde{Z}_{m,n} = \sqrt{\sum_{s=1}^{ns} \left(\frac{M_n}{M_m} Z_{m,n} \right)^2} \tag{3-64}$$

When this modification is made, the chemical species are seen to relax much more quickly than in the original formulation, leading to faster convergence to steady-state, while still maintaining stability.

It should be noted that Imlay and Eberhardt employ a technique called "Damköhler limiting," where chemical reactions are artificially slowed to time scales closer to the fluid dynamic time scales. Even with the above modification to Eberhardt's and Imlay's scheme, this technique proved necessary for adequate performance. Damköhler numbers are typically set between 10 and 100. This technique was not used with the extension of this algorithm to the logarithmic formulation presented in the next section.

3.2.7 Application to the Logarithmic Species Conservation Law

In §3.2.2 the approach for calculating the full Jacobian of the chemical sources for the logarithmic variables, Z, was discussed. Eberhardt and Imlay's algorithm is now applied to this Jacobian matrix. Through numerical experimentation, the following form of the approximate Jacobian was found:

$$\tilde{Z}_{m,n} = \sqrt{\sum_{s=1}^{ns} \left(\frac{q_n}{q_m} Z_{m,n} \right)^2} \tag{3-65}$$

where $\tilde{Z}_{m,n}$ is the diagonal approximation of the Jacobian matrix, and q_n and q_m are the atomic densities and logarithmic variables introduced in §2.6. Of several tried, this is the only form of Eberhardt's algorithm which proved successful for the logarithmic species variables, and its use gives performance similar to Imlay's implementation for the standard flow variables, even though Damköhler limiting is not used.

3.2.8 Extension to Second-Order Time Accuracy

A drawback of Eberhardt's algorithm (and LU-SGS) is that it is not time accurate. Time accuracy can be restored, however, with a simple modification. Shuen *et al.*,[36] Ok,[81] and others have introduced second-order time accuracy with a sub-iterative method based on a three-point backward time discretization, expressed as

$$\frac{\partial Q}{\partial \tau} \approx \frac{3Q^{n+1} - 4Q^{n} + Q^{n-1}}{2\Delta\tau} \tag{3-66}$$

Yungster[37] implemented a similar approach, but allowed for a variable time step:

$$\frac{\partial Q}{\partial \tau} \approx \frac{\left(2+\frac{1}{\Delta}\right)Q^{n+1} - \left(2+\Delta+\frac{1}{\Delta}\right)Q^{n} + (\Delta)\,Q^{n-1}}{(1+\Delta)\,\Delta\tau^{n}}$$
$$\Delta \equiv \frac{\Delta\tau^{n}}{\Delta\tau^{n-1}} \tag{3-67}$$

where $\Delta\tau^{n}$ and $\Delta\tau^{n-1}$ are the current and the previous time steps. A fully implicit algorithm using this time differencing is given by

$$\frac{\left(2+\frac{1}{\Delta}\right)Q^{n+1} - \left(2+\Delta+\frac{1}{\Delta}\right)Q^{n} + (\Delta)\,Q^{n-1}}{(1+\Delta)\,\Delta\tau^{n}} = -R^{n+1} + H^{n+1} + W^{n+1} \tag{3-68}$$

If a vector $X\left(Q^{n+1}\right)$ is defined to be the left-hand side of equation (3-68) minus the right-hand side, a Newton iteration can be performed to find the value of Q^{n+1} which will satisfy this equation. This Newton iteration takes the form

$$\frac{\partial X\left(Q^{n+1}\right)}{\partial Q^{n+1}}\delta Q = -X\left(Q^{n+1}\right)$$
$$Q^{n+1} = Q^{n+1} + \delta Q \tag{3-69}$$

δQ is repeatedly calculated to refine the estimate of Q^{n+1} until the sequence converges. Time is then advanced by $\Delta\tau$, and the process is repeated for a new time step. It is important to note that Q^{n} and Q^{n-1} have already been calculated, and are fixed. Thus, X is considered a function only of the vector Q^{n+1}.

A closer examination of the terms in X will reveal why this represents a simple modification to the previously discussed scheme. X is given by the expression

$$X = \frac{\left(2+\frac{1}{\Delta}\right)Q^{n+1} - \left(2+\Delta+\frac{1}{\Delta}\right)Q^{n} + (\Delta)\,Q^{n-1}}{(1+\Delta)\,\Delta\tau^{n}} + R^{n+1} - H^{n+1} - W^{n+1} \tag{3-70}$$

so the Jacobian of X is calculated by

$$\frac{\partial X}{\partial Q^{n+1}} = \frac{2+\frac{1}{\Delta}}{(1+\Delta)\,\Delta\tau^{n}} + \frac{\partial R}{\partial Q} - \frac{\partial H}{\partial Q} - \frac{\partial W}{\partial Q} \tag{3-71}$$

and equation (3-69) becomes

$$\left(\frac{2+\frac{1}{\Delta}}{(1+\Delta)\,\Delta\tau^{n}} I + \frac{\partial R}{\partial Q} - \frac{\partial H}{\partial Q} - \frac{\partial W}{\partial Q}\right)\delta Q = -R^{n+1} + W^{n+1} - \frac{\left(2+\frac{1}{\Delta}\right)Q^{n+1} - \left(2+\Delta+\frac{1}{\Delta}\right)Q^{n} + (\Delta)\,Q^{n-1}}{(1+\Delta)\,\Delta\tau^{n}} \tag{3-72}$$

Comparison with a conventional, first-order implicit method,

$$\left(\frac{I}{\Delta\tau} + \frac{\partial R}{\partial Q} - \frac{\partial H}{\partial Q} - \frac{\partial W}{\partial Q}\right)\Delta Q = -R^{n} + H^{n} + W^{n} \tag{3-73}$$

reveals that the two are nearly identical in form. Thus, time accuracy can be achieved by defining a sub-iteration in δQ to calculate Q^{n+1} for each successive time step, and this sub-iteration can be performed with the algorithm presented in the previous sections, only slightly modified. With second-order spatial differencing for the terms in R, this algorithm becomes second-order accurate in space and time.

3.3 Thermodynamic Calculations

3.3.1 Specific Heat, Enthalpy, and Entropy

The specific heat of the gas mixture is calculated by mass-weighting the species specific

heats. Specific heats are generally a function of temperature. It has been shown that is necessary to account for this variation in order to solve accurately flows involving wide temperature variations, such as high-speed flow, or flow involving strongly exothermic reactions, such as combustion.[82] Each species specific heat is calculated from fifth- or seventh-order polynomial curve fits, taken from various sources. For the seventh-order fits, this takes the form

$$\frac{c_{p_i}}{R_i} = \frac{z_1}{\tilde{T}^2} + \frac{z_2}{\tilde{T}} + z_3 + z_4\tilde{T} + z_5\tilde{T}^2 + z_6\tilde{T}^3 + z_7\tilde{T}^4 \tag{3-74}$$

where the tilde (~) is used to emphasize that the temperature is the dimensional value, in Kelvin. Once each species specific heat is known, the mixture specific heat is given by

$$c_p = \sum_{s=1}^{ns} c_i c_{p_i} \tag{3-75}$$

The enthalpy, likewise, is a mass-weighted average of species enthalpies. These are calculated by integrating the species specific heats and adding the heat of formation, resulting in

$$\frac{h_i}{R_i\tilde{T}} = -\frac{z_1}{\tilde{T}^2} + \frac{z_2}{\tilde{T}}\ln\tilde{T} + z_3 + \frac{z_4}{2}\tilde{T} + \frac{z_5}{3}\tilde{T}^2 + \frac{z_6}{4}\tilde{T}^3 + \frac{z_7}{5}\tilde{T}^4 + z_{10} \tag{3-76}$$

In general, the change in the entropy of a system is given by

$$ds = \frac{dh}{T} - R\frac{dp}{p} = \frac{c_p dT}{T} - R\frac{dp}{p} \tag{3-77}$$

For these calculations, however, the second term is neglected, and the so-called "one-atmosphere" entropy is then calculated by integrating the first term over temperature, giving

$$\frac{s_i}{R_i} = -\frac{z_1}{2\tilde{T}^2} - \frac{z_2}{\tilde{T}} + z_3\ln\tilde{T} + z_4\tilde{T} + \frac{z_5}{2}\tilde{T}^2 + \frac{z_6}{3}\tilde{T}^3 + \frac{z_7}{4}\tilde{T}^4 + z_9 \tag{3-78}$$

These values are used only to calculate equilibrium constants of chemical reactions, so no mass-weighted average is required.

3.3.2 Temperature

The temperature of the gas mixture is calculated by a Newton iteration, where the temperature is "backed out" from the value of internal energy calculated from the conserved variables. Internal energy is equal to the mixture enthalpy minus $R\tilde{T}$. Thus,

$$\varepsilon = \frac{e}{\rho} - \frac{1}{2}\left(u^2 + v^2\right) = \sum_{s=1}^{ns} c_s \left[\int \left(c_{p_s} - R_s\right) dT + h_s^0\right] \tag{3-79}$$

The appropriate non-dimensionalization has been neglected in this expression for clarity. A Newton iteration now gives

$$\Delta T = \frac{\varepsilon - \sum_{s=1}^{ns} c_s \left[\int \left(c_{p_s} - R_s\right) dT + h_s^0\right]}{\sum_{s=1}^{ns} c_s \left(c_{p_s} - R_s\right)} \tag{3-80}$$

This iteration is repeated at each grid point until the temperature converges to within single-precision epsilon of the calculated value of T, usually within 2 or 3 iterations.

3.4 Molecular Transport Properties

3.4.1 Viscosity

The viscosities of individual gas species are calculated from curve fits by Blottner, of the form

$$\mu_s = 0.1 \exp\left[\left(A_s^b \ln T + B_s^b\right) \ln T + C_s^b\right]. \tag{3-81}$$

where A_s^b, B_s^b, *and* C_s^b are the curve fit coefficients for species s. To find the mixture viscosity, Wilke's mixing rule is then applied, giving

$$\mu = \sum_{s=1}^{ns} \frac{X_s \mu_s}{\phi_s} \tag{3-82}$$

where X_s is the mole fraction of species s, and

$$\phi_s = \sum_{m=1}^{ns} X_m \left[1 + \sqrt{\frac{\mu_s}{\mu_m}} \left(\frac{M_m}{M_s} \right)^{\frac{1}{4}} \right]^2 \left[8 \left(1 + \frac{M_s}{M_m} \right) \right]^{-\frac{1}{2}} \tag{3-83}$$

3.4.2 Thermal Conductivity

The species thermal conductivity is calculated from the species viscosity using the Euken correlation (discussed in Reid *et al.*[83]),

$$k_s = \mu_s \left(\frac{5}{2} c_{v,tr} + c_{v,r} + c_{v,e} \right) \tag{3-84}$$

where $c_{v,tr}$ is the species translational specific heat at constant volume, and $c_{v,r}$ and $c_{v,e}$ are the remaining contributions to the specific heat. The mixture thermal conductivity is then calculated from Wilke's mixing rule, as is done with viscosity. A potentially more accurate approach employing the Lennard-Jones (6-12) potential can be found in Hirschfelder *et al.*,[84] but the approach given here is considered adequate to demonstrate the viability of the algorithm.

3.4.3 Species Diffusivity

The diffusion of chemical species is not considered important in the types of problems examined here,[14,85] so its effects are neglected.

3.5 Other Considerations

A computation can only resolve flow features of the same scale as the grid spacing, or larger. Any sub-grid scale features desired to be accounted for must therefore be modeled. The most common modeling in CFD is that of turbulence, which can occur at scales orders of magnitude smaller than typical CFD grids. Turbulence modeling has met with some success, but the problem becomes more complicated in the presence of chemical reactions. Because chemical reaction rates are often exponentially dependent on temperature, temperature fluctuations due to turbulence can have a significant impact on the

progress of these reactions. The modeling of this turbulence/chemistry interaction, which often takes the form of a probability density function describing the temperature fluctuations in the flow, is in its infancy, and is beyond the intended scope of this work. For this reason, the effects of turbulence are neglected.

3.6 Graphical User Interface

In concert with this work, a library of Fortran-callable routines, *ICFD Tools*, was written which allows a Motif-based Graphical User Interface (GUI) to be quickly added to many existing flow solvers. This library was of significant benefit in the development of the algorithm presented in this chapter. It has also been used in introductory and advanced Computational Fluid Dynamics graduate courses. It allows the programmer or program user to observe the flow solution as it is generated, though contour and line plots, to stop and restart the flow solver, and to change solver parameters during the course of its execution. It also includes a Postscript® driver which allows printing of the solution plots to many laser printers.

Figure 3-3 shows an example of the *ICFD Tools* GUI. The top of the window contains a menu bar containing pull-down menu items to allow plot definition, printing, changing of parameters, etc. The left-hand side of the window contains buttons for control of the flow solver execution (Start, Stop, Step), as well as redrawing of the screen (Update). Below these buttons is a list of all plot variables, with the value of each of these variables at a particular point in the computational grid. To change this "probe" location, the user simply clicks on the desired location with the pointer (mouse). The remainder of the window contains the selected plot, in this case temperature contours, and an indicator of how many iterations have been performed. The user can zoom in on a portion of the plot by

drawing a rectangle with the pointer and selecting "Zoom" from the View menu. The scrollbars then allow scrolling to different areas of the image.

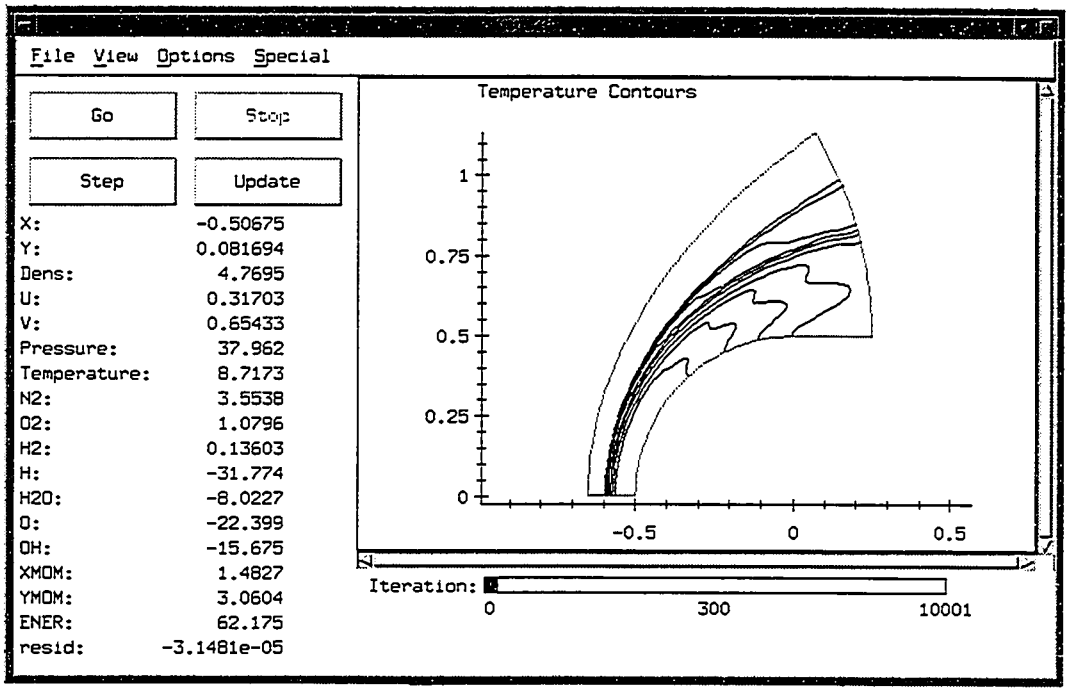

Figure 3-3:*ICFD Tools* Graphical User Interface

All of the above features can be added to many existing CFD programs through the use of as few as three subroutine calls from the *ICFD Tools* library. Other features and a complete description of the library routines are contained in the user's manual, included in Appendix D.

3.7 Summary

In this chapter, the original algorithm of Eberhardt and Imlay has been expanded to encompass Sussman and Wilson's logarithmic transformation of the species conservation law. It has then been further expanded with a sub-iterative approach to allow time-accurate calculations. The memory requirements and computational expense per iteration of the resulting algorithm scale linearly with the number of chemical species.

Chapter 4: Results

This chapter presents test cases used to evaluate the algorithm developed in Chapter 3. Both steady and unsteady computations are matched against Lehr's experiments, and flow solver performance issues are discussed.

4.1 Lehr's Experiments

The first case computed is that of a blunt projectile 1.5 cm in diameter flying at 2,605 m/s through a stoichiometric mixture of hydrogen and air. The detonation speed is 2055 m/s, making this a superdetonative case. The free stream temperature and pressure are 291.59 Kelvin and 320 torr, respectively, and the soundspeed is 403 m/s, giving a Mach number of 6.46.

Figure 4-1 shows a shadowgraph taken by Lehr of his experiment for these conditions. Near the stagnation line, the bow shock and the combustion front are coupled. Further away from the stagnation line, the bow wave curves, dropping the normal component of the Mach number. At the point where the component of the flow speed normal to the shock falls below the detonation speed, the shock and the combustion front separate, and continue to separate further as the shock angle decreases with respect to the free stream flow. For the given flight Mach number, the separation point is located where the shock is inclined about 38° from vertical.

Calculations of this case are performed on a 50 by 40 grid, which was shown in Figure 3-1. The flow is first initialized to free stream conditions. The flow solver is then run at an infinite time step, giving an approximate Newton iteration. To avoid instabilities due to

large starting transients, the first 100 iterations are performed without chemical reactions. r_A and r_B, the implicit damping terms introduced in equation (3-60), are set to 1.2 times the spectral radius of the flux Jacobian matrices A and B. The flow solver is run until the solution has converged seven orders of magnitude, roughly single precision machine zero.

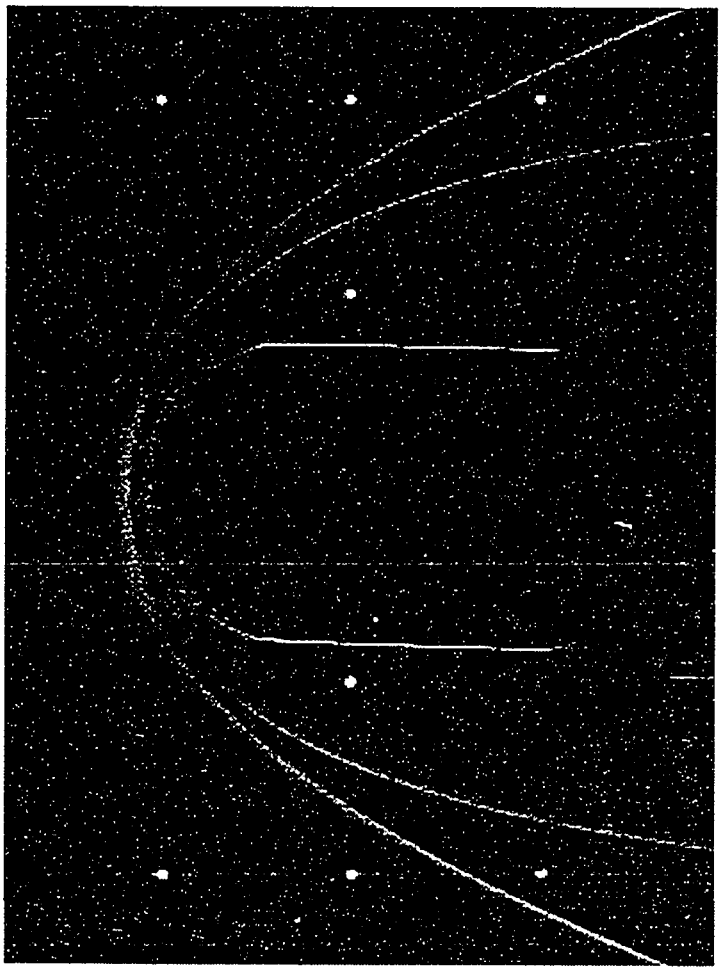

Figure 4-1: Lehr's Mach 6.46 Hydrogen-Air test case

Figure 4-2 shows density contours for two numerical simulations of this case using Waldman's reaction model. Note that the grid has been non-dimensionalized with respect to body diameter. Both figures contain data points indicating the location of the shock and the combustion front which Lehr observed. The figure on the left shows a calculation done with the standard form of the species conservation laws. The shock and the combustion

front are coupled, and the shock appears to be significantly displaced from its experimentally observed location, resembling an oblique detonation. The use of a 100 by 80 grid made no noticeable difference in the solution. The figure on the right, by contrast, is done with the logarithmic species variables, and shows proper shock location, as well as a good representation of the induction zone separating the shock and the combustion front.

It is clear from these calculations that the species density formulation introduces significant error. Specifically, it is unable to properly resolve the induction zone. This is not altogether surprising, because the species mass fractions are changing exponentially in this region, and a suitable grid should have enough points to resolve significant changes in the flow. The logarithmic formulation avoids this problem by using variables which change only linearly in the induction zone, requiring fewer points for proper resolution. The drawback to this approach, as Wilson points out, is additional numerical diffusion which may distort contact surfaces. This distortion does not become important in the problems examined here, however, and use of the logarithmic variables allows accurate computations of flows with far fewer grid points than would be required otherwise.

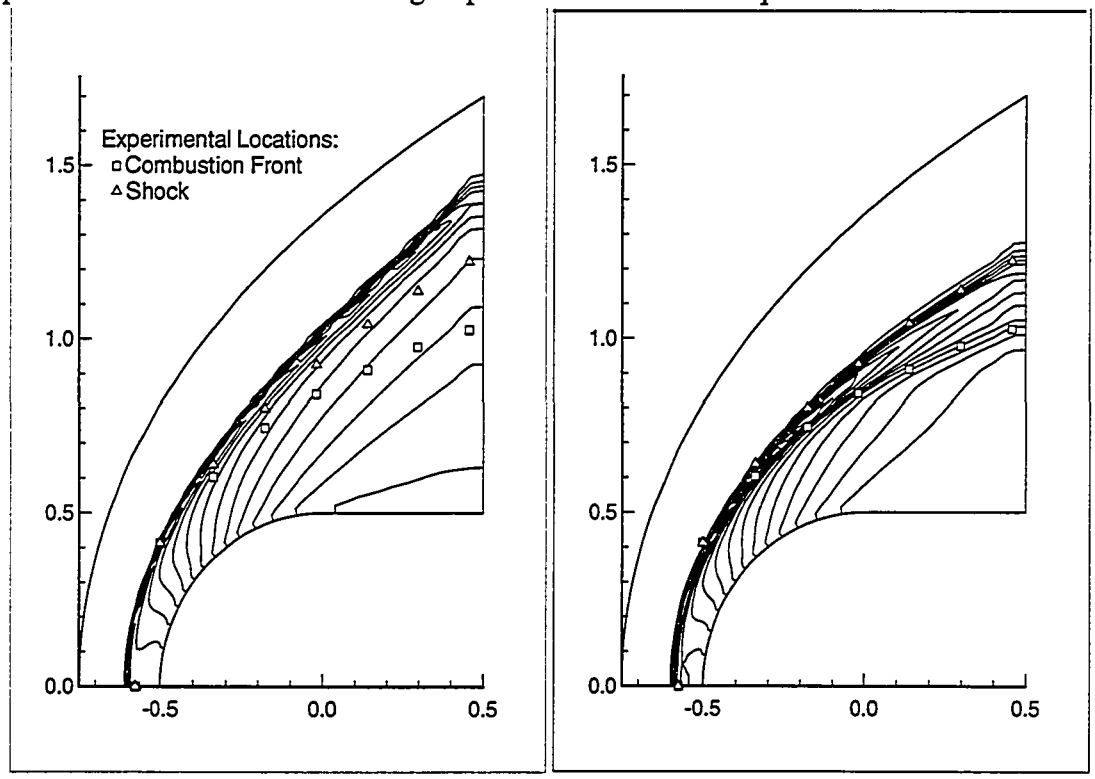

Figure 4-2: Numerical Simulations of Lehr's Mach 6.46 Case

Figure 4-3 shows contours of temperature and water mass fraction for this case. The temperature contours show both the shock and the combustion locations, while the water mass fraction contours show the region where combustion is complete. Lehr notes that the slope of the combustion front continues to decrease further downstream, having no limiting value above horizontal.

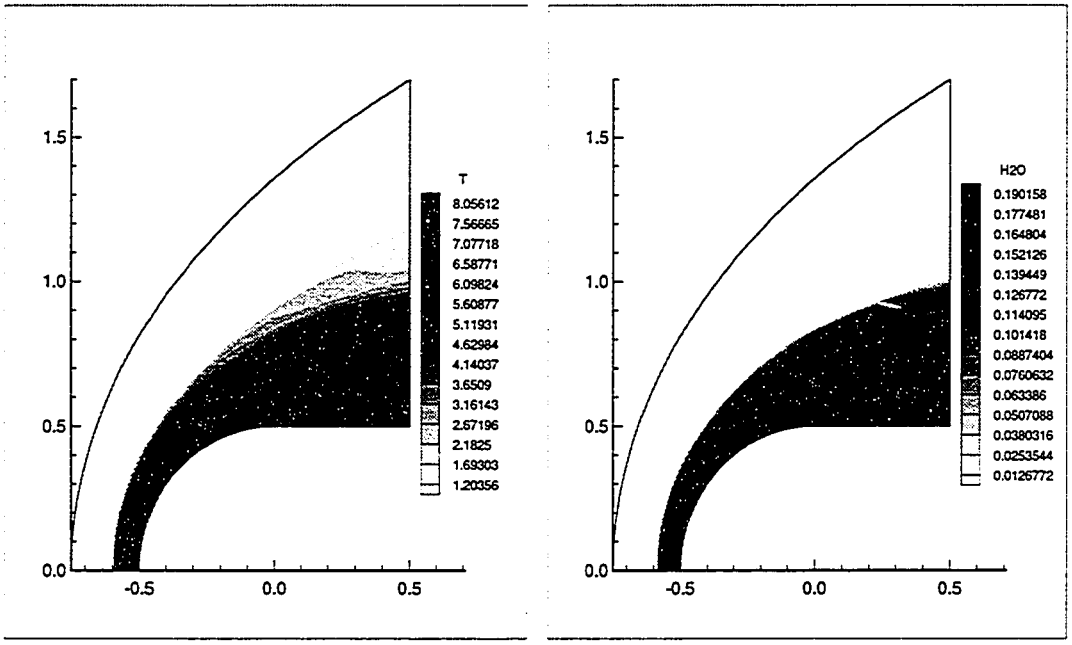

Figure 4-3: Temperature and Water Mass Fraction for the Mach 6.46 Case

Figure 4-4 shows the L2-norm of the total energy residual versus iteration number for the logarithmic species reacting flow, and a non-reacting case for comparison. Although the chemical reactions clearly slow the convergence, this comparison demonstrates that the approximation made with the chemical source Jacobian is not having an overly adverse impact on the convergence rate.

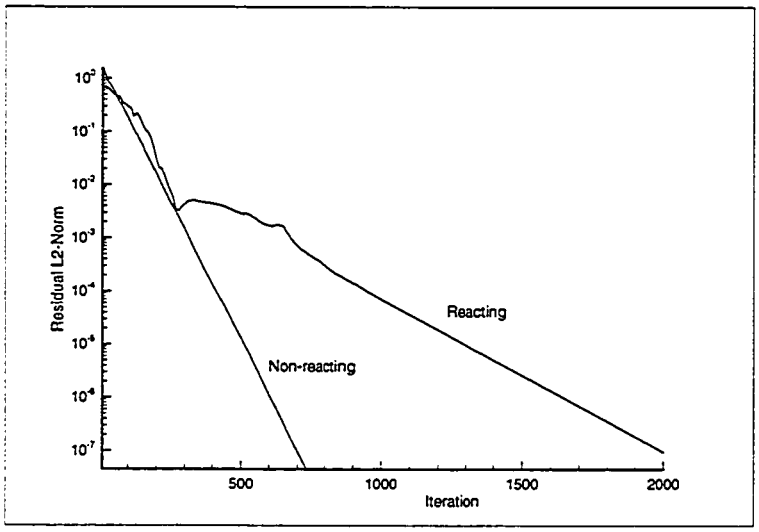

Figure 4-4: Convergence Rate for Lehr's Steady Mach 6.46 Case

The above demonstration provides a good validation of the present method for inviscid steady-state calculations. In addition, it shows the advantage of the logarithmic variables, and the potential pitfalls of using the standard variables in the presence of exponential variation of species mass fractions. To validate the unsteady capability of the present method, an unsteady case also studied by Lehr is calculated.

Lehr noted that an unsteady phenomenon occurs when the projectile's velocity is just below the detonation velocity of the gas mixture. McVey and Toong[86] studied this phenomenon and proposed a wave interaction model to explain it. This model consists of four distinct steps, all of which occur near the stagnation line:

(1) A compression wave traveling away from the body joins the shock and strengthens it. The shock then moves away from the body, and a rarefaction and a contact discontinuity travel toward the body. On the upstream side of the contact discontinuity, the gas is hotter due to the stronger shock. The rarefaction is ignored.

(2) The hotter gas on the upstream side of the contact discontinuity has a shorter ignition delay time, so the gas begins to burn prior to reaching the existing combustion front. This creates compression waves which travel both upstream toward the shock and downstream toward the body.

(3) The new combustion front reaches the previous combustion front, and extinguishes. This sends rarefaction waves (an expansion fan) upstream and downstream.

(4) Rarefaction waves begin to reach the shock and weaken it. As this happens, the shock moves back to its original location, a series of weak pressure waves travel downstream, and the fluid downstream of the shock cools to its original temperature. This restores the original ignition delay and combustion front location.

The above sequence of events completes one cycle. Note that the compression wave from step (2) reaches the shock prior to the rarefaction of step (3). In fact, before this compression wave reaches the shock, the rarefaction from the previous cycle reaches it and weakens it. Thus, the shock is subjected alternately to compression and rarefaction waves. The effect of the shock alternately strengthening and weakening causes the movement of the combustion front, which in turn generates compression and rarefaction waves to perpetuate the process. Figure 4-5 shows a schematic of this model. The complexity of this flow field represents a significant computational challenge, and an excellent test of the accuracy of the current algorithm.

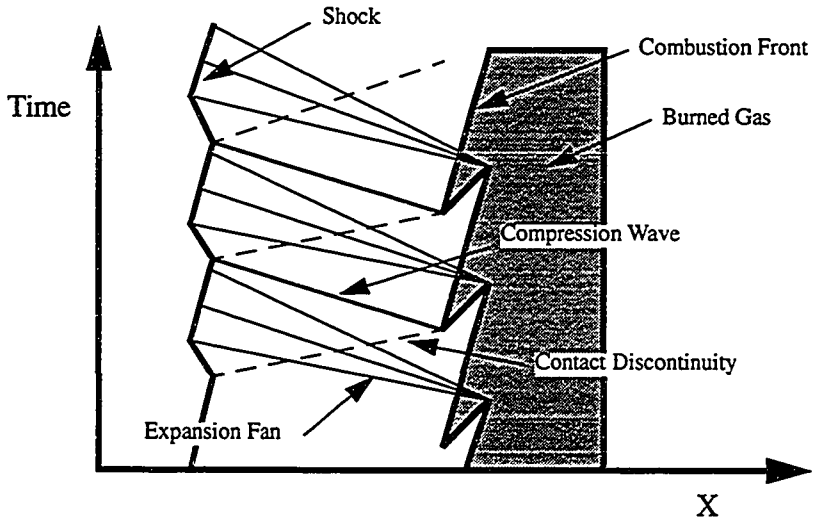

Figure 4-5: A Schematic of the Model of McVey and Toong

Figure 4-6 shows a shadowgraph taken from one of Lehr's unsteady cases. The shock appears smooth, because the deflections due to the compression and expansion waves are small. The combustion front, in contrast, has a corrugated appearance due to the unsteady

process convecting away from the stagnation line. By measuring the distance between the waves, Lehr estimated the frequency of the pulsations. Note that the vertical lines seen below result from the axisymmetry of the pulses.

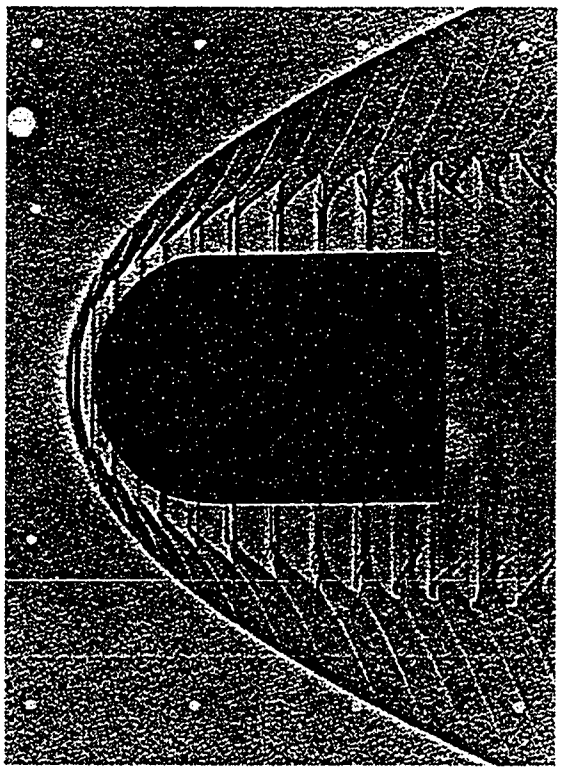

Figure 4-6: Lehr's Unsteady Mach 4.79 Case

In this case, the projectile is flying at 1,930 m/s, Mach 4.79. The calculation was performed on a 155 by 155 grid. It was observed that if the flow is initialized at free stream conditions, the time accurate solution exhibits a large transient due to ignition near the nose of the projectile which pushes the bow shock to the outer boundary of the grid. To avoid this, the flow solver was run at an approximate Newton iteration for 700 iterations, allowing the gas behind the shock to ignite. From this point, the solver was run in time accurate mode for an additional 20,000 subiterations. For each time step, 40 sub-iterations were performed. The time step was adjusted to ensure that the residual is reduced by six orders of magnitude during the course of these sub-iterations. The resulting Courant number varied between about 2 and 5. At the conclusion of the run, 500 time steps had been

calculated, for a total of about 20 μs, and 11 clearly identifiable pulses had been observed. Figure 4-7 shows density contours produced at the end of this time period.

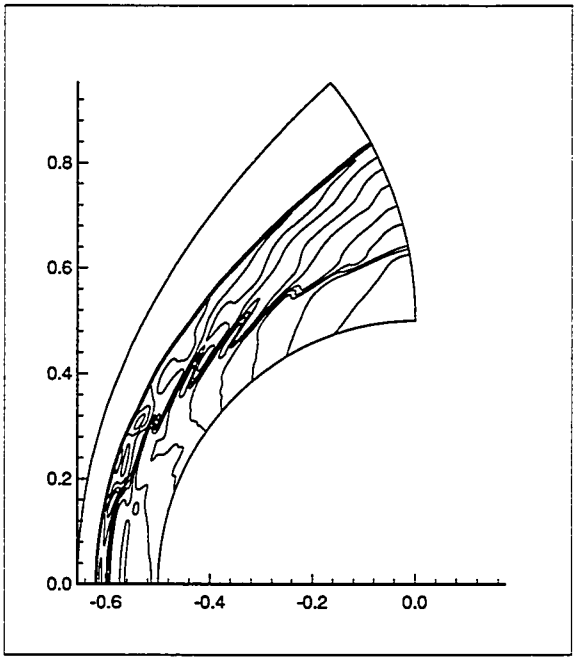

Figure 4-7: Density Contours from a Numerical Simulation of Lehr's Mach 4.79 Case

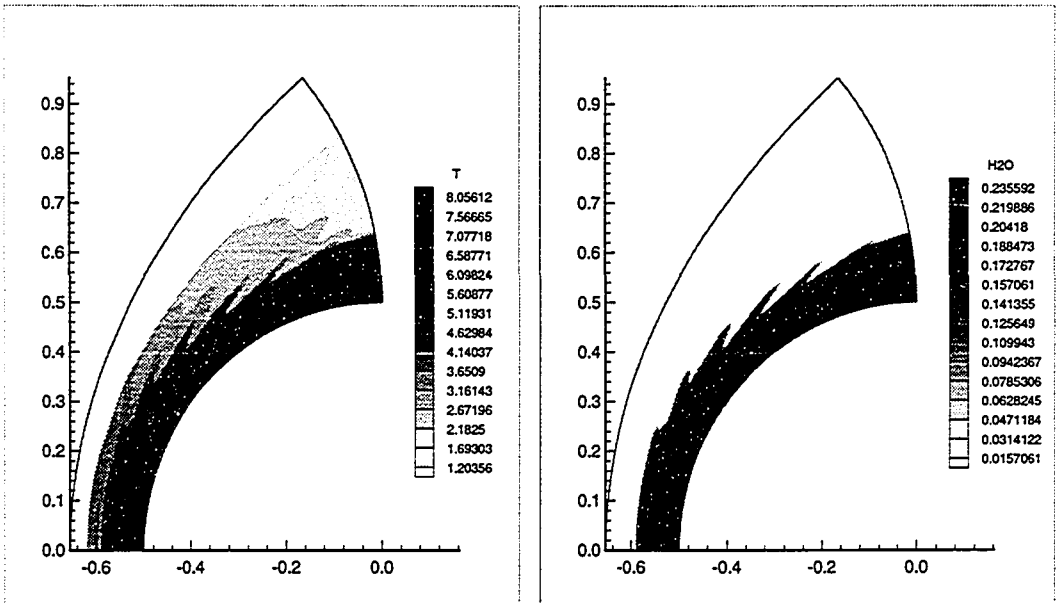

Figure 4-8: Temperature and Water Mass Fraction Contours for the Mach 4.79 Case

The perturbations in density take a form similar to those seen in Figure 4-6, although they are smeared out near the outflow boundary due to numerical damping. Figure 4-8 shows temperature and water mass fraction contours for this case. The locations of the shock and the combustion front conform to the model outlined above. Note that small deformations of the shock are visible in these plots. This computation can be compared with McVey and Toong's model by examining plots of density and pressure along the stagnation line as a function of time. These plots are shown in Figure 4-9.

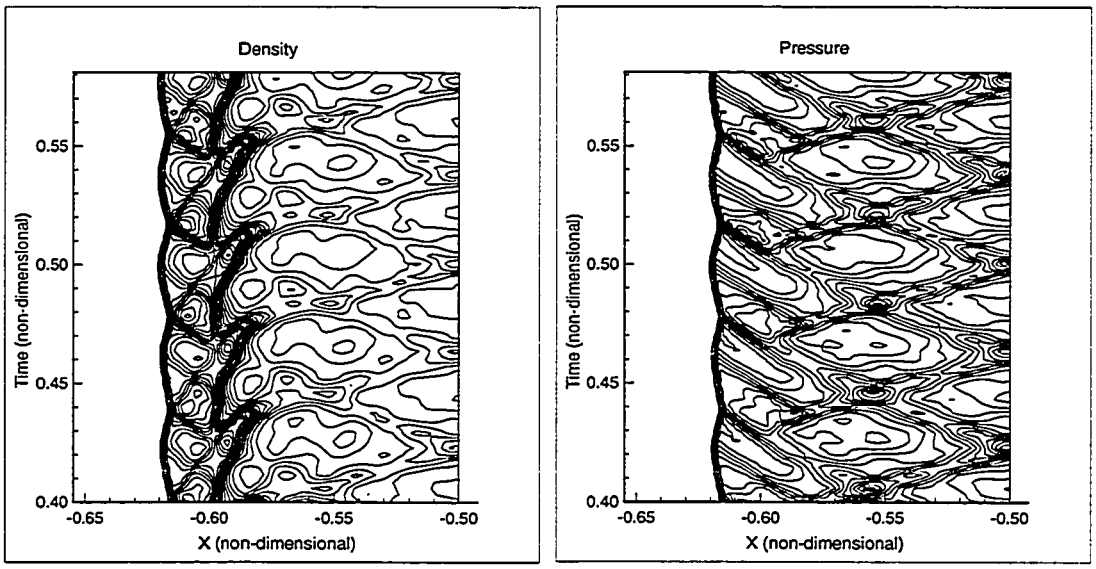

Figure 4-9: Density and Pressure Along the Stagnation Line for the Mach 4.79 Case

The density contours show the location of the shock and the combustion front, as well as interactions between the two. By comparing the density and pressure contours, we see that a compression wave indeed emanates from the beginning of the advanced combustion front. When this wave reaches the shock, the shock moves outward, producing a contact discontinuity visible in the density plot. At the point where the advanced combustion front reaches the existing front and extinguishes, the pressure plot reveals a rarefaction, which travels to the shock and restores it to its original position. Thus, the basic mechanism proposed by McVey and Toong appears to be correct, a result which has also been verified by other researchers.[33,34,37]

An interesting feature of this unsteady flow is the compression waves which travel downstream from the advanced combustion front (these were neglected in McVey and Toong's model). These compression waves reflect off the body, and apparently combine with and augment the compression waves which travel upstream from the advanced combustion front. Yungster noted these, and also noted that the relative timing of these two compression waves is not precise, leading to small irregularities in the pulsations. These irregularities were apparent in a movie produced from the present computation, but are not obvious in the above figures.

To evaluate the time accuracy of the algorithm, the duration of the final six pulses of the present computation was measured. The resulting frequency of 717KHz compares quite favorably with the experimentally determined frequency of 720KHz. By comparison, Yungster[37] calculated a frequency of 701-716 KHz, Matsuo *et al.* calculated a frequency of 725 KHz, Wilson and Sussman[34] calculated a frequency of 530 KHz, and Hosangadi *et al.* found a frequency of 450-500 KHz. Yungster suggests that errors in the results of Hosangadi *et al.* may be due to the simpler 7-species model used. The cause of error in the work of Wilson and Sussman is not clear. They reported that the frequency did not change as they further refined the grid, and suggested that errors in the chemical reaction model might be the cause. The present work, however, as well as that of Yungster and of Matsuo *et al.*, achieved accurate results using a simplification of Wilson and Sussman's reaction model, clearly indicating that the problem lay elsewhere.

The above calculations demonstrate that the present algorithm is capable of accurately predicting both steady and unsteady cases of shock-driven combustion. The efficiency of the algorithm will now be examined.

4.2 Performance Measurements

The performance of the present algorithm is evaluated by three criteria: (1) the computational efficiency for a problem of a given size, (2) the grid economy permitted by the use of the logarithmic species conservation law, and (3) the memory requirements.

4.2.1 Computational Efficiency

Computational requirements of the present algorithm are compared with results published by Wilson and Sussman,[34] and by Yungster[37] for reproductions of unsteady Lehr cases. Wilson and Sussman reported a requirement of approximately 2.8 X 10^{-4} CPU seconds per time step per grid point on a Cray Y-MP. Due, perhaps, to time step limitations of their point-implicit algorithm, they performed between 10,000 and 20,000 time steps. On the 155 by 155 grid used here, this would result in a run time of between 18 and 36 hours on a Cray Y-MP. Yungster reported that a typical calculation required approximately 4,000 time steps and 7.5 hours of CPU time on a Cray C-90 for a 220 by 220 grid. On the grid used here, this would scale to about 3.7 hours of CPU time on the Cray C-90. By comparison, the present algorithm was run on an IBM RS/6000 3AT for 500 time steps, which took about 48 hours of CPU time. Direct comparisons of these results are obviously impossible, but estimates of the speeds of the machines involved can give a good indication of the relative performance of these three algorithms.

Yungster reported that his algorithm runs at about 250 MFLOPS (Million FLoating-point Operations Per Second). A reasonable estimate of the performance of a Y-MP is 170 MFLOPS. The RS/6000 3AT has a Linpack double precision rating of about 50 MFLOPS, but an operation count revealed a sustained performance of 30 MFLOPS for the unsteady Lehr case. Using these figures, the total number of floating-point operations required to perform the unsteady calculation were estimated, and are displayed in Figure 4-10. These numbers, divided by the speed of a particular machine, give the total CPU time for a calculation.

It would appear that the present algorithm takes about 1.5 times as much CPU time as Yungster's algorithm, and one-half as much CPU time as the algorithm of Wilson and Sussman. In spite of some uncertainty in this comparison (especially in light of the dramatic difference in the number of time steps computed), it appears that the present algorithm is slower than that of Yungster, and faster than that of Wilson and Sussman.

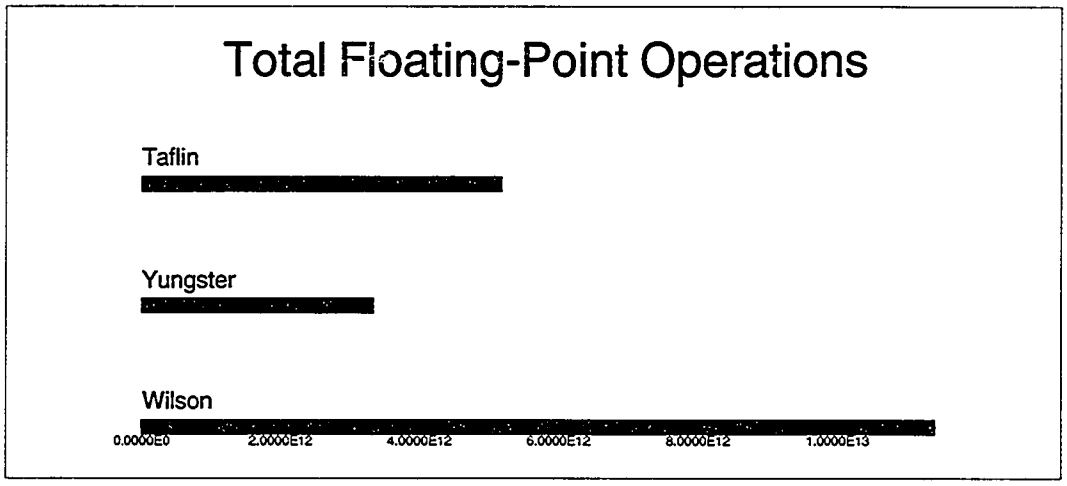

Figure 4-10: Total Floating-Point Operations for the Unsteady Lehr Case

In this comparison, the relative computational efficiencies are primarily the result of two differences seen among the three algorithms just discussed: (1) the degree of implicitness, and (2) the use of a full versus an approximate chemical source Jacobian. Wilson and Sussman used a point-implicit approach (refer to §3.2.1), where only the chemical sources are calculated at the new time level. In contrast, the present algorithm and that of Yungster are fully implicit—all terms on the right-hand side of the conservation equations are calculated at the new time level.

The present algorithm makes use of an approximate Jacobian of the chemical sources, while the others use the full Jacobian. This has the benefit of using less CPU time per iteration, but there is the danger of slowing the convergence rate to the point of not being beneficial.[79] For the nine species case tested here, inversion of the full Jacobian is apparently advantageous.

The advantage of the approximate Jacobian may become more important as the number of species is increased. The computational expense of inverting the full Jacobian scales with the number of species cubed. In contrast, the expense of inverting the approximate Jacobian increases linearly with number of species. The result is that the expense of the entire algorithm is nearly linear in the number of species, as is shown in Figure 4-11,

where the CPU time required per iteration is plotted for increasing numbers of species with a constant number of chemical reactions.

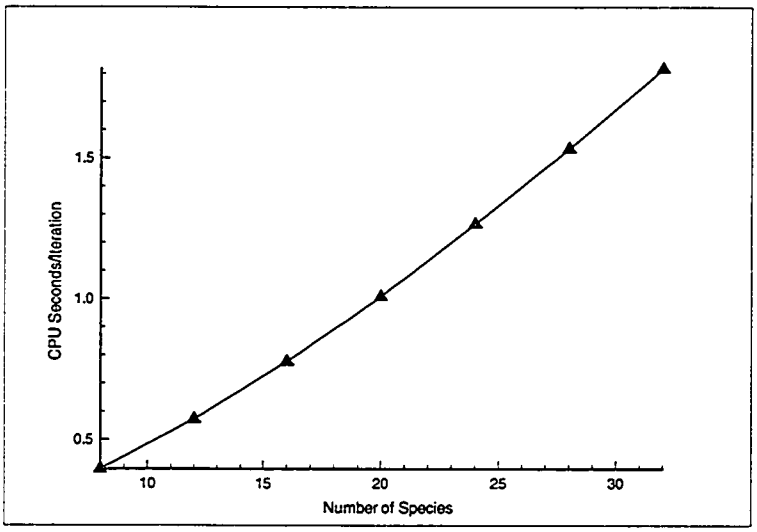

Figure 4-11:CPU Time per Iteration vs. Number of Chemical Species

To test this possibility, a computation was done with a methane-air reaction model proposed by Yungster and Rabinowitz.[39] The model consists of 20 species (including nitrogen) and 52 reactions. In this case, a cylinder with a diameter of 3mm moves through a stoichiometric mixture of methane and air at the superdetonative speed of 2,330 m/s (Mach 6.61). The temperature and pressure ar 295K and 0.51 bar, respectively.

The methane-air test case was computed on a 91 by 91 grid. It was run for 12,000 iterations, taking about 29 hours on an RS6000 workstation. The solution converged only about four orders of magnitude, after which the residual oscillated. This may indicate the need for refining the approximate Jacobian for use in cases with methane combustion. Figure 4-12 shows temperature and water mass fraction contours of the resulting solution. The shock and combustion front locations calculated by Yungster and Rabinowitz are indicated on the plot of temperature contours. The shock location of the present computation agrees well with that of Yungster and Rabinowitz. The location of the combustion front agrees well near the stagnation line, but diverges toward the outflow boundary. Although experimental data for this case are not available, it seems likely that Yungster and Rabinowitz's results are more accurate due to the convergence difficulty noted above.

Yungster and Rabinowitz did not report on the convergence for this case, but Yungster[88] indicated that full convergence may not be possible. The calculation of Yungster and Rabinowitz required 7 to 10 hours on a Cray C-90, indicating that the present algorithm may be faster on this problem.

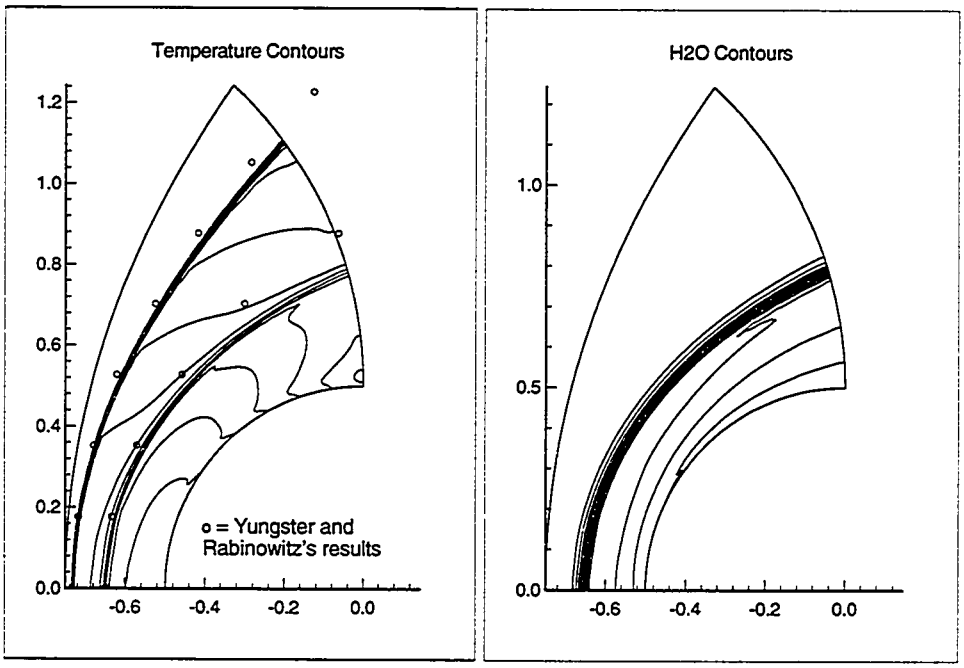

Figure 4-12: Methane-Air Test Case

4.2.2 Grid Economy

Lehr's steady Mach 6.46 case is computed here on a 50 by 40 grid, and achieves very good comparison with Lehr's experimental results. A computation with the species density formulation, even on a grid twice the size, is not able to successfully reproduce these results. In one-dimensional simulations, Sussman[89] showed that the logarithmic transformation allows the use of as few as 10 grid points within the induction zone between the shock and the energy release front to reproduce unsteady phenomena accurately. By contrast, the conventional formulation requires about 40 points. Without grid adaption, then, it would appear that for flows with shock-induced combustion, the present algorithm may be able to use only one-fourth as much grid due to its use of the logarithmic variables. In reality, this is problem-dependent. The steady case mentioned here highlights the advantage,

but for Lehr's unsteady Mach 4.79 case, additional grid was required to capture other details of the flow, reducing the advantage of the logarithmic transformation. This observation was also made by Wilson and Sussman.[34] In general, however, the use of the logarithmic variables has significant benefit in the computation of flows with shock-induced combustion.

4.2.3 Memory Requirements

The present algorithm makes very efficient use of memory by avoiding storage of the chemical source Jacobian matrices, whereas all other algorithms mentioned must store them or their inverse. This becomes significant as the number of chemical species is increased, since the size of these matrices increases with the number of species squared. For the hydrogen-air reaction model used by Yungster, the storage of these matrices requires 81 floating-point numbers per grid point. For a methane-air model used by Yungster and Rabinowitz,[39] this becomes 361 numbers per grid point, and easily dwarfs all other memory requirements of the algorithm. Storage of the approximate Jacobian, by contrast, only requires one number per species per grid point, offering increasing advantage as the number of species grows. In the extreme case of a complex hydrocarbon combustion model, such as that presented for methane-air combustion by Ranzi *et al.*,[90] nearly 1000 megabytes of memory would be required for the full-Jacobian methods on the 155 by 155 grid, while only about 80 megabytes would be required by the present algorithm.

The linear dependence of the present algorithm is illustrated by Figure 4-13, where the memory required is plotted for increasing numbers of species for a 46 by 34 grid. Also

shown in this figure for comparison is the memory requirement for a typical full-Jacobian method.

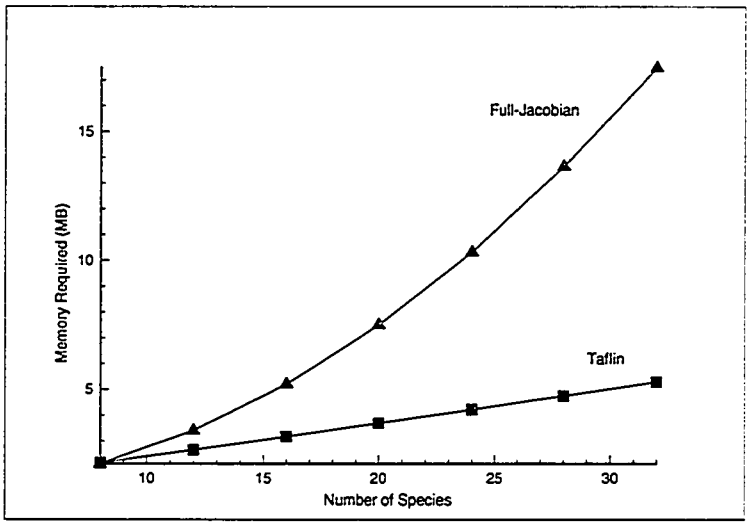

Figure 4-13: Memory Requirements for Increasing Numbers of Species

4.3 Discussion of Performance

The preceding section demonstrated that the present algorithm is superior to other algorithms in terms of memory usage, but not necessarily superior in terms of CPU time. Since increasing computing power continues to enable the consideration of larger and more complex problems, a discussion of the relative performance of this algorithm on larger problems is in order.

Problems may become larger due either to increasing grid size, or to increasingly complex reaction models. For all algorithms discussed here, CPU time and memory requirements will scale linearly with grid size (neglecting vectorization issues, memory bank conflicts, and other arcane details). Increasing the size of the reaction model, however, has a more complicated effect. Although computations were obviously not done with the reaction set of Ranzi *et al.* (the author would still be waiting for the results!), it will be used as an example of a complex reaction model.

The computational expense of inverting the left-hand side of the implicit equation has been emphasized. Yet the calculation of the chemical source terms themselves may also represent a significant expense, since they involve exponentiations. The largest hydrogen-air reaction model used here contains 9 species and 19 reactions. Ranzi *et al.* propose 70 species and more than 1600 reactions for methane-air combustion. Although many of these can be eliminated for most computations, these numbers may be realistic for more complex hydrocarbon combustion cases. Now we consider how the computational expense of the present algorithm would be affected if such a large model were used. It was noted that the expense of flux calculations scales linearly with the number of species. Likewise, the cost of inverting the implicit equation scales linearly with species. The expense of calculating the chemical source terms, however, scales linearly with the number of reactions. Thus, the computational expense related to flux calculations and inverting the implicit equation would increase by a factor of about 8 for Ranzi *et al.*'s model, while the expense of calculating the chemical sources would increase by a factor of 84. Clearly, the source term calculations would dominate the expense of the algorithm as a whole.

Now we consider a method, such as Yungster's, which inverts the full chemical source Jacobian. The cost of calculating the chemical sources would still increase by a factor of 84, but the cost of inverting the implicit equation would increase by a factor of about 470! The expense of these methods, then, becomes dominated by the expense of inverting the left-hand side. Since the expense of the two approaches scales differently as the size of the reaction model is increased, we may expect that each will have its advantages. Specifically, the advantage of using the full Jacobian for small numbers of equations to improve the convergence rate may become outweighed by the advantage of cheap inversion of the left-hand side with the approximate Jacobian for increasing numbers of species. The preceding methane-air computation gave evidence to support this hypothesis, but further investigation, beyond the intended scope of this work, is called for to confirm it for larger reaction models.

Chapter 5: Conclusions

The approximate chemical source Jacobian matrix of Eberhardt and Imlay was adapted for use with Sussman and Wilson's logarithmic transformation of the species continuity equations. Time accuracy was then introduced through a sub-iterative correction procedure. The resulting algorithm was tested on both steady and unsteady cases of hypersonic shock-driven combustion in hydrogen-air mixtures, and the performance of the algorithm was compared against other published results.

The algorithm was found to be capable of accurately reproducing both steady and unsteady cases of hydrogen-air shock-driven combustion. For steady cases, an accurate prediction of both shock and combustion front locations at super-detonative speed was obtained on a 50 by 40 grid. This was a much smaller grid than can be used with the original form of the species conservation equations, demonstrating the advantage of the logarithmic transformation. For unsteady cases, an accurate prediction was obtained for the frequency of the unsteady interaction of the combustion with the bow shock around a blunt projectile at near-detonation speed. Again, the logarithmic transformation allowed use of a smaller grid, although this advantage was reduced somewhat by the requirement to adequately resolve the finely detailed structure of the flow.

The new algorithm was found to be competitive in terms of CPU time. It was superior to a point-implicit method, but was inferior to a fully-implicit method which uses the full chemistry Jacobian. The full-Jacobian method apparently converges quickly enough to offset the added expense of inverting the Jacobian. To test the method for larger reaction models, a steady sub-detonative methane-air case was computed. The present method was more competitive with the full-Jacobian method for this case because its expense per iteration scales nearly linearly with the number of species, whereas the expense for the full-

Jacobian method scales with the number of species cubed. However, the new algorithm did not converge fully, indicating a necessity to refine it for methane combustion.

Since it does not store the full Jacobian of the chemical sources, the new algorithm requires much less memory than other algorithms. This advantage was shown to increase dramatically as the number of species is increased.

With refinement, future applications of the new algorithm could include its use with larger chemical reaction models, such as those seen in hydrocarbon combustion. The algorithm is also well-suited for use as a smoother in multi-grid methods, although care should be taken to use a multi-grid method capable of solving flows with strong shocks. Its low expense per iteration also makes this algorithm a likely candidate to be a preconditioner for conjugate gradient-type algorithms. The use of such algorithms, however, would eliminate the primary advantage of this scheme, its limited memory usage.

List of References

[1] Oates, G. C., *Aerothermodynamics of Gas Turbine and Rocket Propulsion*, American Institute of Aeronautics and Astronautics, Inc., Washington, DC, 1988.

[2] Rubins, P. M., and Bauer, R. C., "Review of Shock-Induced Supersonic Combustion Research and Hypersonic Applications," *Journal of Propulsion and Power*, Vol. 10, 1994, pp. 593-601.

[3] Hedges, L. S., "Numerical Simulation of the Acoustic Instability in the Spacially Developing, Confined, Supersonic Mixing Layer," Ph.D. Dissertation, Department of Aeronautics and Astronautics, University of Washington, Seattle, WA, 1991.

[4] Hedges, L. S., and Eberhardt, D. S., "Comparison of Confined, Compressible, Spacially Developing Mixing Layers with Temporal Mixing Layers," *AIAA Journal*, Vol. 31, 1993, pp. 1977-1983.

[5] Papamoschou, D., "Thrust Loss due to Supersonic Mixing," *Journal of Propulsion and Power*, Vol. 10, 1994, pp. 804-809.

[6] Bruckner, A. P., Bogdanoff, D. W., Knowlen, C., and Hertzberg, A., "Investigations of Gasdynamic Phenomena Associated with the Ram Accelerator Concept," AIAA Paper 87-1327, 1987.

[7] Knowlen, C., Bruckner, A. P., Bogdanoff, D. W., and Hertzberg, A., "Performance Capabilities of the Ram Accelerator," AIAA Paper 87-2152, 1987.

[8] Kaloupis, P., and Bruckner, A. P., "The Ram Accelerator: A Chemically Driven Mass Launcher," AIAA Paper 88-2968, 1988.

[9] Bruckner, A. P., Knowlen, C., Scott, K. A., and Hertzberg, A., "High Velocity Modes of the Thermally Choked Ram Accelerator," AIAA Paper 88-2925, 1988.

[10] Hertzberg, A., Bruckner, A. P., and Bogdanoff, D. W., "Ram Accelerator: A New Chemical Method for Accelerating Projectiles to Ultrahigh Velocities," *AIAA Journal*, Vol. 26, 1988, pp. 195-203.

[11] Hinkey, J. B., Burnham, E. A., and Bruckner, A. P., "High Spacial Resolution Measurements of Ram Accelerator Gas Dynamic Phenomena," AIAA Paper 92-3244, 1992.

[12] Yungster, S., and Bruckner, A. P., "Computational Studies of a Superdetonative Ram Accelerator Mode," *AIAA Journal*, Vol. 8, 1989, pp. 457-463.

[13] Yungster, S., Eberhardt, S., and Bruckner, A. P., "Numerical Simulation of Hypervelocity Projectiles in Detonable Gases," *AIAA Journal*, Vol. 29, 1991, pp. 187-199.

[14] Chuck, C., "Numerical Simulation of Oblique Detonation and Shock-Deflagration Waves with a Laminar Boundary-Layer," Ph.D. Dissertation, Department of Aeronautics and Astronautics, University of Washington, Seattle, WA, Jul. 1990.

[15] Chuck, C., Eberhardt, S., and Pratt, D. T., "Combusting Flow Simulations of Detonation and Shock-Induced Combustion Waves for Ram Accelerator Configurations," AIAA Paper 91-1674, Jun. 1991.

[16] Fort, J. A., "A Numerical Study of Attached Oblique Detonation," Ph.D. Dissertation, Department of Mechanical Engineering, University of Washington, Seattle, WA, Aug. 1993..

[17] Oran, E. S., Li, C., and Kailasanath, K., "Numerical Studies for the Ram Accelerator," AFOSR TR 93-0020, 1993.

[18] Bussing, T., and Pappas, G., "An Introduction to Pulse Detonation Engines," AIAA Paper 94-0263, 1994.

[19] Bussing, T. R. A., Hinkey, J. B., and Kaye, L., "Pulse Detonation Engine Preliminary Design Considerations," AIAA Paper 94-3220, 1994.

[20] Mattick, A. T., Hertzberg, A., and Russell, D. A., "Shock Controlled Reactors," *Proceedings of the 18th International Symposium on Shock Waves*, Sendai, Japan, Ed. K. Takayama, Springer-Verlag, 1991, pp. 1289-1294.

[21] Russell, D. A., Mattick, A. T., Hertzberg, A., and Knowlen, C., "A New Chemical Reactor Based on High-Energy Laser Technology," AIAA Paper 94-2457, 1994.

[22] Knowlen, C., Mattick, A. T., Russell, D. A., and Masse, R. K., "Petrochemical Pyrolysis with Shock Waves," AIAA Paper 95-0402, 1995.

[23] Soetrisno, M., Imlay, S. T., and Roberts, D. W., "Numerical Simulations of the Transdetonative Ram Accelerator Combusting Flow Field on a Parallel Computer," AIAA Paper 92-3249, Jul. 1992.

[24] Eberhardt, D. S., and Imlay, S. T., "A Diagonal Implicit Scheme for Computing Flows with Finite-Rate Chemistry," AIAA Paper 90-1577, 1990.

[25] Candler, G. V., and Olynick, D. R., "Hypersonic Flow Simulations Using a Diagonal Implicit Method," *Proceedings of the 10th International Conference on Computing Methods in Applied Sciences and Engineering*, Paris, France, Ed. R. Glowinski, Nova Science Publishers, Inc., New York, 1991, pp. 29-48.

[26] Hassan, B., Candler, G. V., and Olynick, D. R., "The Effect of Thermo-Chemical Nonequilibrium on the Aerodynamics of Aerobraking Vehicles," AIAA Paper 92-2877, Jul. 1992.

[27] Ting, J. M., Bussing, T. R. A., and Hinkey, J. B., "Experimental Characterization of the Detonation Properties of Hydrocarbon Fuels for the Development of a Pulse Detonation Engine," AIAA Paper 95-3154, 1995.

[28] Hinkey, J. B., Bussing, T. R. A., and Kaye, L., "Shock Tube Experiments for the Development of a Hydrogen-Fueled Pulse Detonation Engine," AIAA Paper 95-2578, 1995.

[29] Sterling, J., Ghorbanian, K., Humphrey, J., and Pratt, D., "Numerical Investigations of Pulse Detonation Wave Engines," AIAA Paper 95-2479, 1995.

[30] Lynch, E. D., Edelman, R., and Palaniswamy, S., "Computational Fluid Dynamic Analysis of the Pulse Detonation Engine Concept," AIAA Paper 94-0264, 1994.

[31] Lehr, H., "Experiments on Shock-Induced Combustion," *Astonautica Acta*, Vol. 17, 1972, pp. 589-597.

[32] McVey, B. J., and Toong, T. Y., "Mechanism of Instabilities of Exothermic Hypersonic Blunt-Body Flows," *Combustion Science and Technology*, Vol. 3, No. 2, 1971, pp. 63-76.

[33] Wilson, G. J., "Computation of Steady and Unsteady Shock-Induced Combustion Over Hypervelocity Blunt Bodies," Ph.D. Dissertation, Deparment of Aeronautics and Astronautics, Stanford University, Stanford, CA, Dec. 1991.

[34] Wilson, G. J., and Sussman, M. A., "Computation of Unsteady Shock-Induced Combustion Using Logarithmic Species Conservation Equations," *AIAA Journal*, Vol. 31, No. 2, 1993, pp. 294-301.

[35] Sussman, M. A., and Wilson, G. J., "Computation of Chemically Reacting Flows Using a Logarithmic Form of the Species Conservation Equations," *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, Davis, CA, Vol. 2, Sept. 1991, pp. 1113-1118.

[36] Shuen, J. S., Chen, K.-H., and Choi, Y., "A Time-Accurate Algorithm for Chemical Non-Equilibrium Viscous Flows at All Speeds," AIAA Paper 92-3639.

[37] Yungster, S., and Radhakrishnan, K., "A Fully Implicit Time Accurate Method for Hypersonic Combustion: Application to Shock-Induced Combustion Instability," AIAA Paper 94-2965.

[38] Ju, Yiguang, "Lower-Upper Scheme for Chemically Reacting Flow with Finite Rate Chemistry," *AIAA Journal*, Vol. 33, No. 8, 1995, pp. 1418-1432.

[39] Yungster, S., and Rabinowitz, M. J., "Numerical Study of Shock-Induced Combustion in Methane-Air Mixtures," AIAA Paper 93-1917.

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

[41] Waldman, C. H., Wilson, R. P., Jr., and Maloney, K. L., "Kinetic Mechanisms of Methane-Air Combustion with Pollutant Formation," Evironmental Protection Technology Series EPA-650/2-74-045, 1974.

[42] Pratt, D. T., private communication.

[43] Jachimowski, C. J., "An Analytical Study of Hydrogen-Air Reaction Mechanism With Application to Scramjet Combustion," NASA TP 2791, 1988.

[44] Matsuo, A., Fujiwara, T., and Fuji, K., "Flow Features of Shock-Induced Combustion Around Projectile Traveling at Hypervelocities," AIAA Paper 93-0451, 1993.

[45] Liepmann, H. W., and Roshko, A., *Elements of Gasdynamics*, Wiley, New York, 1957.

[46] Leveque, R. J., *Numerical Methods for Conservation Laws*, Birkhauser Verlagg, Basel, 1992.

[47] Godunov, S. K., "Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics," *Mat. Sb.*, Vol. 47, 1959, pp. 271-306.

[48] Chorin, A. J., "Random Choice Solution of Hyperbolic Systems," *Journal of Computational Physics*, Vol. 22, 1976, pp. 517-533.

[49] Liou, M.-S., and van Leer, B., "Choice of Implicit and Explicit Operators for the Upwind Differencing Method," AIAA Paper 88-0624, 1988.

[50] Liou, M.-S., and Steffen, C. J., Jr., "A New Flux Splitting Scheme," NASA TM 104404, 1991.

[51] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357-372.

[52] Roe, P. L., "Some Contributions to the Modelling of Discontinuous Flows," *Lectures in Applied Mathematics*, Vol. 22, 1985, pp. 163-193.

[53] Osher, S., "Numerical Solutions of Singular Perturbation Problems and Hyperbolic Systems of Conservation Laws," *Mathematical Studies*, Vol. 47, North-Holland, Amsterdam, 1981, pp. 179-205.

[54] van Leer, B., and Shuen, J.-S., "Splitting of Inviscid Fluxes for Real Gases," *Journal of Computational Physics*, Vol. 87, no. 1, 1990, pp. 1-24.

[55] Grossman, B., Cinnella, P., and Garret, J., "A Survey of Upwind Methods for Flows with Equilibrium and Nonequilibrium Chemistry and Thermodynamics," AIAA Paper 89-1653, 1989.

[56] Grossman, B., and Cinnella, P., "The Development of Flux-Split Algorithms for Flows with Nonequilibrium Thermodynamics and Chemical Reactions," AIAA Paper 88-3595, 1988.

[57] Grossman, B., and Walters, R. W., "An Analysis of Flux-Split Algorithms for Euler's Equations with Real Gases," AIAA Paper 87-1117, 1987.

[58] Montagne, J.-L., and Yee, H. C., Comparative Study of High-Resolution Shock-Capturing Schemes for a Real Gas," NASA TM 100004, 1987.

[59] Vinokur, M., "Flux Jacobian Matrices and Generalized Roe Average for an Equilibrium Real Gas," NASA CR 177512, 1988.

[60] Shinn, J. L., Yee, H. C., and Uenishi, K., "Extension of a Semi-Implicit Shock-Capturing Algorithm for 3-D Fully Coupled, Chemically Reacting Flows in Generalized Coordinates," AIAA Paper 87-1577, 1987.

[61] Carofano, G. C., "Blast Computation Using Harten's Total Variation Diminishing Scheme," TR ARLCB-TR-84029, 1984

[62] Abgrall, R., "Généralisation du Schéma de Roe pour le Calcul d'Écoulements de Mélanges de Gas a Concentrations Variables," La Recherche Aérospatiale, no. 1988-6, pp. 31-43.

[63] Yee, H. C., Warming, R. F., and Harten, A., "Implicit Total Variation Diminishing Schemes for Steady-State Calculations," *Journal of Computational Physics*, Vol. 57, no. 3, 1985, pp. 327-360.

[64] Yee, H. C., Klopfer, G. H., and Montagné, J.-L., "High-Resolution Shock-Capturing Schemes for Inviscid and Viscous Hypersonic Flows," NASA TM 100097, 1988.

[65] Imlay, S. T., Roberts, D. W., and Soetrisno, M., "An Efficient 3D Navier-Stokes Analysis for Evaluation of Hypersonic Vehicles," Final Report, NASA Contract No. NAS8-37406, Oct. 1990.

[66] Harten, A. "High Resolution Scheme for Hyperbolic Conservation Laws," *Journal of Computational Physics*, Vol. 49, 1983, pp. 357-393.

[67] Harten, A., "On a Class of High Resolution Schemes for Total Variation Stable Finite-Difference Schemes," *SIAM Journal of Numerical Analysis*, Vol. 21, no. 1, 1984, pp. 1-23.

[68] Yee, H. C., "Construction of Explicit and Implicit Symmetric TVD Schemes and Their Application," Journal of Computational Physics, Vol. 68, 1987, pp. 151-179.

[69] Chakravarthy, S. R., "Development of Upwind Schemes for the Euler Equations," NASA CR 4043, 1987.

[70] Yee, H. C., "Upwind and Symmetric Shock Capturing Schemes," NASA TM 89464, 1987.

[71] Pratt, D. T., and Wormeck, J. J., "CREK: A Computer Program for Calculation of Combustion Reaction Equilibrium and Kinetics in Laminar or Turbulent Flow," Report WSU-ME-TEL-76-1, Washington State Univ., Mar. 1976.

[72] Gielda, t. and McRae, D., "An Accurate, Stable, Explicit, Parabolized Navier-Stokes Solver for High-Speed Flows," AIAA Paper 86-1116, 1986.

[73] Thomas, P. D., and Lombard, C. K., "The Geometric Conservation Law — A Link between Finite-Difference and Finite-Volume Methods of Flow Computation on Moving Grids," AIAA Paper 78-1208, 1978.

[74] Nordstrom, J., "Accuracy of the Time-Dependent Navier-Stokes Equations Using Extrapolation Procedures at Outflow Boundaries," AIAA Paper 91-1605, 1991.

[75] Hoffman, K. A., and Chiang, S. T., *Computational Fluid Dynamics for Engineers*, Engineering Education System, Wichita, 1993.

[76] Bussing, T. R. A., and Murman, E. M., "A Finite Volume Method for the Calculation of Compressible Chemically Reacting Flows," AIAA Paper 85-03331, 1985.

[77] Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite-Difference Methods," NASA TM D-78605, 1978.

[78] Yoon, S., and Jameson, A., "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," AIAA Paper 87-0600, 1987.

[79] Yungster, S., "Navier-Stokes Simulation of the Supersonic Combustion Flowfield in a Ram Accelerator," NASA TM 104439, 1991.

[80] Imlay, S. T., Amtec Engineering, Inc., private communication.

[81] Ok, H., "Development of an Incompressible Navier-Stokes Solver and its Application to the Calculation of Separated Flows," Ph.D. Dissertation, Department of Aeronautics and Astronautics, University of Washington, Seattle, WA, Jun. 1993.

[82] Wada, Y., Kubota, S., and Ishiguro, T., "A Diagonalizing Formulation of General Real Gas-Dynamic Matrices with a New Class of TVD Schemes," AIAA Paper 88-3596-CP, 1988.

[83] Reid, R. C., Prausnitz, J. M., and Poling, B. E., *The Properties of Gases and Liquids*, McGraw-Hill, 1987.

[84] Hirschfelder, J. O., Curtiss, C. F., and Bird, R. B., *Molecular Theory of Gases and Liquids*, John Wiley and Sons, Inc., New York, 1954.

[85] Bruckner, A. P., Department of Aeronautics and Astronautics, University of Washington, private communication.

[86] McVey, J. B., and Toong, T. Y., "Mechanism of Instabilities of Exothermic Hypersonic Blunt-Body Flows," *Combustion Science and Technology*, Vol. 3, 1971, pp. 63-76.

[87] Hosangadi, A., York, B. J., Sinha, N., and Dash, S. M., "Progress in Transient Interior Ballistic Flowfield Simulations Using Multi-Dimensional Upwind/Implicit Numerics," AIAA Paper 93-1915, 1993.

[88] Yungster, S., ICOMP - NASA Lewis Research Center, private communication.

[89] Sussman, M. A., "A Computational Study of Unsteady Shock-Induced Combustion of Hydrogen-Air Mixtures," AIAA Paper 94-3101, 1994.

[90] Ranzi, E., Sogaro, A., Gaffuri, P., Pennati, G., and Faravelli, T., "A Wide Range Modeling Study of Methane Oxidation," *Combustion Science and Technology*, Vol. 96, 1994, pp. 279-325.

Appendix A: Chemical Reaction Models

Nine reactions are taken from Waldman to describe hydrogen/oxygen combustion. The units for these expressions are moles, centimeters, seconds, and Kelvins. In this table and the one which follows, the values of E_a (in calories) are given instead of C, to be consistent with the original references. C is calculated by

$$C = \frac{E_a}{R} \tag{A-1}$$

The reactions from Waldman are listed here with expressions for the forward reaction constants:

Table A-1: Waldman's Combustion Model

Reaction	A	B	C
1. $OH + O \leftrightarrow H + O_2$	2.50×10^{13}	0.0	0.0
2. $H + OH \leftrightarrow H_2 + O$	8.00×10^{9}	1.0	7,000
3. $OH + H_2 \leftrightarrow H + H_2O$	2.50×10^{13}	0.0	5,200
4. $OH + OH \leftrightarrow H_2O + O$	6.00×10^{12}	0.0	1,000
5. $H_2O + M \Rightarrow OH + H + M$	3.00×10^{15}	0.0	105,000
6. $H + O + M \leftrightarrow OH + M$	8.00×10^{15}	0.0	0.0
7. $O_2 + M \leftrightarrow O + O + M$	2.50×10^{19}	-1.0	118,700
8. $H + HO_2 \leftrightarrow OH + OH$	2.50×10^{14}	0.0	1,900
9. $H + O_2 + M \leftrightarrow HO_2 + M$	1.50×10^{15}	0.0	1,000

Jachimowski's hydrogen/air combustion model included 13 species and 33 chemical reactions. Wilson modified the reaction constants for reactions 2 and 6, deleted one reaction, and replaced another. The results are shown below. Yungster, as well as Matsuo, et. al., eliminated reactions 20 through 32, leaving 9 species and 19 reactions.

Table A-2: Jachimowski's Combustion Model

Reaction	A	B	E_a
1. $H_2 + O_2 \leftrightarrow HO_2 + H$	1.00×10^{14}	0.0	56000
2. $H + O_2 \leftrightarrow OH + O$	2.60×10^{14}	0.0	16800
3. $O + H_2 \leftrightarrow OH + H$	1.80×10^{10}	1.0	8900
4. $OH + H_2 \leftrightarrow H_2O + H$	2.20×10^{13}	0.0	5150
5. $OH + OH \leftrightarrow H_2O + O$	6.30×10^{12}	0.0	1090
6. $H + OH + M \leftrightarrow H_2O + M$	2.20×10^{22}	-2.0	0
7. $H + H + M \leftrightarrow H_2 + M$	6.40×10^{17}	-1.0	0
8. $H + O + M \leftrightarrow OH + M$	6.00×10^{16}	-0.6	0
9. $H + O_2 + M \leftrightarrow HO_2 + M$	2.10×10^{15}	0.0	-1000
10. $HO_2 + H \leftrightarrow OH + OH$	1.40×10^{14}	0.0	1080
11. $HO_2 + H \leftrightarrow H_2O + O$	1.00×10^{13}	0.0	1080
12. $HO_2 + O \leftrightarrow O_2 + OH$	1.5×10^{13}	0.0	950
13. $HO_2 + OH \leftrightarrow H_2O + O_2$	8.00×10^{12}	0.0	0
14. $HO_2 + HO_2 \leftrightarrow H_2O_2 + O_2$	2.00×10^{12}	0.0	0
15. $H + H_2O_2 \leftrightarrow H_2 + HO_2$	1.40×10^{12}	0.0	3600
16. $O + H_2O_2 \leftrightarrow OH + HO_2$	1.40×10^{13}	0.0	6400
17. $OH + H_2O_2 \leftrightarrow H_2O + HO_2$	6.10×10^{12}	0.0	1430
18. $H_2O_2 + M \leftrightarrow OH + OH + M$	1.20×10^{17}	0.0	45500
19. $O + O + M \leftrightarrow O_2 + M$	6.00×10^{13}	0.0	-1800
20. $N + N + M \leftrightarrow N_2 + M$	2.80×10^{17}	-0.75	0
21. $N + O_2 \leftrightarrow NO + O$	6.40×10^{9}	1.0	6300
22. $N + NO \leftrightarrow N_2 + O$	1.60×10^{13}	0.0	0
23. $N + OH \leftrightarrow NO + H$	6.30×10^{11}	0.5	0
24. $H + NO + M \leftrightarrow HNO + M$	5.4×10^{15}	0.0	-600
25. $H + HNO \leftrightarrow NO + H_2$	4.80×10^{12}	0.0	0
26. $O + HNO \leftrightarrow NO + OH$	5.00×10^{11}	0.5	0
27. $OH + HNO \leftrightarrow NO + H_2O$	3.6×10^{13}	0.0	0
28. $HO_2 + HNO \leftrightarrow NO + H_2O_2$	2.00×10^{12}	0.0	0
29. $HO_2 + NO \leftrightarrow NO_2 + OH$	3.40×10^{12}	0.0	-260
30. $H + NO_2 \leftrightarrow NO + OH$	3.50×10^{14}	0.0	1500
31. $O + NO_2 \leftrightarrow NO + O_2$	1.00×10^{13}	0.0	600
32. $NO_2 + M \leftrightarrow NO + O + M$	1.16×10^{16}	0.0	66000

Third-body efficiencies:

Reaction 6. H_2O = 6.0 Reaction 7. H_2O = 6.0 H_2 = 2.0

Reaction 8. H_2O = 5.0 Reaction 9. H_2O = 16.0 H_2 = 2.0

Reaction 10. H_2O = 15.0

Appendix B: Convective Flux Jacobian Diagonalization

The Euler equations are given by

$$\frac{\partial Q}{\partial \tau} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} = 0 \tag{B-1}$$

When the logarithmic form of the species continuity equations is used, Q, the vector of conserved variables, and F and G, the convective fluxes of these variables in the ξ and η directions, are given by

$$Q = J^{-1}\begin{bmatrix} \rho_1^* \\ \vdots \\ \rho_{nl}^* \\ \pi_{nl+1} \\ \vdots \\ \pi_{ns} \\ \rho u \\ \rho v \\ e \end{bmatrix} \qquad F = J^{-1}\begin{bmatrix} \rho_1^* U \\ \vdots \\ \rho_{nl}^* U \\ \pi_{nl+1} U \\ \vdots \\ \pi_{ns} U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ U(e+p) \end{bmatrix} \qquad G = J^{-1}\begin{bmatrix} \rho_1^* V \\ \vdots \\ \rho_{nl}^* V \\ \pi_{nl+1} V \\ \vdots \\ \pi_{ns} V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ V(e+p) \end{bmatrix} \tag{B-2}$$

The jacobian matrices are defined as

$$A_{m,n} \equiv \frac{\partial F_m}{\partial Q_n} \qquad B_{m,n} \equiv \frac{\partial G_m}{\partial Q_n} \tag{B-3}$$

The elements of A are given on the following page. B may be calculated by replacing ξ with η, and U with V in these expressions.

$A =$

$$\begin{bmatrix}
U\left(1-\frac{\rho_1^*}{\rho}\right) & -\frac{\rho_1^*}{\rho}U & 0 & 0 & \xi_x\frac{\rho_1^*}{\rho} & \xi_y\frac{\rho_1^*}{\rho} & 0 \\
-\frac{\rho_2^*}{\rho}U & U\left(1-\frac{\rho_2^*}{\rho}\right) & 0 & 0 & \xi_x\frac{\rho_2^*}{\rho} & \xi_y\frac{\rho_2^*}{\rho} & 0 \\
-\frac{\pi_3}{\rho}U & -\frac{\pi_3}{\rho}U & U & 0 & \xi_x\frac{\pi_3}{\rho} & \xi_y\frac{\pi_3}{\rho} & 0 \\
-\frac{\pi_4}{\rho}U & -\frac{\pi_4}{\rho}U & 0 & U & \xi_x\frac{\pi_4}{\rho} & \xi_y\frac{\pi_4}{\rho} & 0 \\
-uU+\xi_x p_{\rho 1^*} & -uU+\xi_x p_{\rho 2^*} & \xi_x p_{\pi 3} & \xi_x p_{\pi 4} & U+\xi_x u(1-p_e) & \xi_y u-\xi_x v p_e & \xi_x p_e \\
-vU+\xi_y p_{\rho 1^*} & -vU+\xi_y p_{\rho 2^*} & \xi_y p_{\pi 3} & \xi_y p_{\pi 4} & \xi_x v-\xi_y u p_e & U+\xi_y v(1-p_e) & \xi_y p_e \\
U\left(p_{\rho 1^*}-H\right) & U\left(p_{\rho 2^*}-H\right) & U p_{\pi 3} & U p_{\pi 4} & \xi_x H-uU p_e & \xi_y H-vU p_e & U(1+p_e)
\end{bmatrix} \qquad \text{(B-4)}$$

To implement an upwinding scheme, the flux jacobians A and B must be diagonalized. A matrix diagonalization is an expression of a matrix as a similarity transform of a diagonal matrix, that is,

$$A = R\Lambda R^{-1} \tag{B-5}$$

where the columns of R consist of the right eigenvectors of A, the rows of R^{-1} consist of the left eigenvectors, of A, and Λ is a diagonal matrix of the eigenvalues of A. Λ is given by

$$\begin{bmatrix} U & & & \\ & \ddots & & \\ & & U + a\sqrt{\xi_x^2 + \xi_y^2} & \\ & & & U - a\sqrt{\xi_x^2 + \xi_y^2} \end{bmatrix} \tag{B-6}$$

The elements of R and R^{-1} are given on the following two pages for A. Again, for B, these may be calculated by replacing ξ with η, and U with V in these expressions.

$R =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \frac{\rho_1^*}{\rho} & \frac{\rho_1^*}{\rho} \\ 0 & 1 & 0 & 0 & 0 & \frac{\rho_2^*}{\rho} & \frac{\rho_2^*}{\rho} \\ 0 & 0 & 1 & 0 & 0 & \frac{\pi_1}{\rho} & \frac{\pi_1}{\rho} \\ 0 & 0 & 0 & 1 & 0 & \frac{\pi_2}{\rho} & \frac{\pi_2}{\rho} \\ u & u & 0 & 0 & \xi_y' & u+\xi_x' a & u-\xi_x' a \\ v & v & 0 & 0 & -\xi_x' & v+\xi_y' a & v-\xi_y' a \\ H-\frac{(a_1^*)^2}{P_e} & H-\frac{(a_2^*)^2}{P_e} & -\frac{p_{\pi 1}}{p_e} & -\frac{p_{\pi 2}}{p_e} & \xi_y' u-\xi_x' v & H+aU' & H-aU' \end{bmatrix} \tag{B-7}$$

with the definitions

$$(a_m^*)^2 = p_{\rho m^*} + p_e\left(H-u^2-v^2\right) \qquad \xi_x' = \frac{\xi_x}{\sqrt{\xi_x^2+\xi_y^2}} \qquad \xi_y' = \frac{\xi_y}{\sqrt{\xi_x^2+\xi_y^2}} \qquad U' = \xi_x' u + \xi_y' v \tag{B-8}$$

R^{-1} =

$$
\begin{bmatrix}
1-\frac{\rho_1^* p_{\rho 1}^*}{\rho\, a^2} & -\frac{\rho_1^* p_{\rho 2}^*}{\rho\, a^2} & -\frac{\rho_1^* p_{\pi 3}}{\rho\, a^2} & -\frac{\rho_1^* p_{\pi 4}}{\rho\, a^2} & \frac{\rho_1^* u p_e}{\rho\, a^2} & \frac{\rho_1^* v p_e}{\rho\, a^2} & -\frac{\rho_1^* p_e}{\rho\, a^2} \\
-\frac{\rho_2^* p_{\rho 1}^*}{\rho\, a^2} & 1-\frac{\rho_2^* p_{\rho 2}^*}{\rho\, a^2} & -\frac{\rho_2^* p_{\pi 3}}{\rho\, a^2} & -\frac{\rho_2^* p_{\pi 4}}{\rho\, a^2} & -\frac{\rho_2^* u p_e}{\rho\, a^2} & -\frac{\rho_2^* v p_e}{\rho\, a^2} & -\frac{\rho_2^* p_e}{\rho\, a^2} \\
-\frac{\pi_3 p_{\rho 1}^*}{\rho\, a^2} & -\frac{\pi_3 p_{\rho 2}^*}{\rho\, a^2} & 1-\frac{\pi_3 p_{\pi 3}^*}{\rho\, a^2} & -\frac{\pi_3 p_{\pi 4}^*}{\rho\, a^2} & \frac{\pi_3 u p_e}{\rho\, a^2} & \frac{\pi_3 v p_e}{\rho\, a^2} & -\frac{\pi_3 p_e}{\rho\, a^2} \\
-\frac{\pi_4 p_{\rho 1}^*}{\rho\, a^2} & -\frac{\pi_4 p_{\rho 2}^*}{\rho\, a^2} & -\frac{\pi_4 p_{\pi 3}^*}{\rho\, a^2} & 1-\frac{\pi_4 p_{\pi 4}^*}{\rho\, a^2} & \frac{\pi_4 u p_e}{\rho\, a^2} & \frac{\pi_4 v p_e}{\rho\, a^2} & -\frac{\pi_4 p_e}{\rho\, a^2} \\
-\xi_y' u+\xi_x' v & -\xi_y' u+\xi_x' v & 0 & 0 & \xi_y' & -\xi_x' & 0 \\
\frac{1}{2}\left(\frac{p_{\rho 1}^*}{a^2}-\frac{U'}{a}\right) & \frac{1}{2}\left(\frac{p_{\rho 2}^*}{a^2}-\frac{U'}{a}\right) & \frac{p_{\pi 3}}{2a^2} & \frac{p_{\pi 4}}{2a^2} & \frac{1}{2}\left(\frac{\xi_x'}{a}-\frac{u p_e}{a^2}\right) & \frac{1}{2}\left(\frac{\xi_y'}{a}-\frac{v p_e}{a^2}\right) & \frac{p_e}{2a^2} \\
\frac{1}{2}\left(\frac{p_{\rho 1}^*}{a^2}+\frac{U'}{a}\right) & \frac{1}{2}\left(\frac{p_{\rho 2}^*}{a^2}+\frac{U'}{a}\right) & \frac{p_{\pi 3}}{2a^2} & \frac{p_{\pi 4}}{2a^2} & \frac{1}{2}\left(-\frac{\xi_x'}{a}-\frac{u p_e}{a^2}\right) & \frac{1}{2}\left(-\frac{\xi_y'}{a}-\frac{v p_e}{a^2}\right) & \frac{p_e}{2a^2}
\end{bmatrix}
\tag{B-9}
$$

In the preceding matrices, the derivatives of pressure with respect to all conserved variables are required. For the derivatives

with respect to momentum and energy, these are the same as for the species density formulation, and are given by

$$\begin{aligned} p_m &= -up_e \\ p_n &= -vp_e \\ p_e &= \frac{R}{c_v} \end{aligned} \tag{B-10}$$

The definitions of p_m and p_n have already been substituted into the matrices. For the remaining derivatives, the derivatives of pressure with respect to species densities are calculated, and then transformed to the logarithmic variable set with the transformation

$$\frac{\partial p}{\partial Q} = \frac{\partial p}{\partial Q'}\frac{\partial Q'}{\partial Q} \tag{B-11}$$

where Q' represents the species densities, and Q represents the logarithmic species variables. The matrix $\frac{\partial Q'}{\partial Q}$ is presented in Appendix C.

Appendix C: Jacobian Matrix of Species Density Variables with Respect to Logarithmic Variables

Q represents the conserved variables for the logarithmic species formulation, and Q' represents the species density formulation:

$$Q = J^{-1}y\begin{bmatrix} \rho_1^* \\ \\ \rho_{nl}^* \\ \pi_{nl+1} \\ \\ \pi_{ns} \\ \rho u \\ \rho v \\ e \end{bmatrix} \qquad Q' = J^{-1}y\begin{bmatrix} \rho_1 \\ \\ \rho_n \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{C-1}$$

The Jacobian matrix $\frac{\partial Q'}{\partial Q}$ is desired, so Q' is first expressed in terms of Q. This was detailed in §3.2.2. The Jacobian is then taken with respect to Q, the result of which is given below, with the definition

$$S_m = M_m^* \sum_{s=nl+1}^{ns} \frac{n_{m,s}}{M_s} c_s (1 - ln\ c_s) \tag{C-2}$$

$$\frac{\partial Q'}{\partial Q} =$$

$$\begin{bmatrix} n_{11}\frac{M_1^*}{M_1} & n_{12}\frac{M_1^*}{M_2} & 0 & 0 & 0 & 0 & 0 \\ n_{21}\frac{M_2^*}{M_1} & n_{22}\frac{M_2^*}{M_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1-S_1 & -S_1 & -\frac{M_1^*}{M_3}n_{1,3}c_3 & -\frac{M_1^*}{M_4}n_{1,4}c_4 & 0 & 0 & 0 \\ -S_2 & 1-S_2 & -\frac{M_2^*}{M_3}n_{2,3}c_3 & -\frac{M_2^*}{M_4}n_{2,4}c_4 & 0 & 0 & 0 \\ c_3(1-ln\ c_3) & c_3(1-ln\ c_3) & c_3 & 0 & 0 & 0 & 0 \\ c_4(1-ln\ c_4) & c_4(1-ln\ c_4) & 0 & c_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(C-3)}$$

If each of the species $1 \ldots nl$ (species 1 and 2 here) contains only atoms of the corresponding element, i.e., $n_{jk} = 0$ for all $j \neq k$, then the first matrix in the above equation becomes the identity matrix.

Appendix D: ICFD Tools User's Manual

ICFD Tools is a library based on Motif and the X Window System which allows CFD programmers to quickly and easily add a graphical user interface to their programs. Its development was begun as this author's Master's thesis, and it was further refined to enhance CFD instruction at the University of Washington. It has proven to be extremely useful not only in the instructional capacity, but also as a debugging tool, because it allows users to observe and interact with their programs. The users can observe line and contour plots of the data while the program is running (with full zooming and scrolling), stop the program or execute one iteration at a time, save their data to a text file, and print plots on Postscript printers or to a text file. Through optional subroutines, the interface is customizable. Programmers can design their own dialog boxes to access program parameters and change them "on the fly," and create pull-down menu items which will invoke FORTRAN subroutines they write (such as an initialization subroutine to allow resetting the program to initial conditions). Use of this package significantly aided the development of the algorithm presented in this work. The ICFD Tools User's Manual begins on the following page.

ICFD Tools

Interactive Graphical User Interface Tools for Computational Fluid Dynamics

D.1 Introduction

ICFD Tools is a set of six FORTRAN-callable subroutines which allow CFD programmers to add an X Window-based graphical user interface to their programs. It supports 1-D or 2-D, structured grid, single or multiple zone CFD programs. These tools can generally be added to existing programs with little or no further modification to the programs. Only the first three subroutines are required for basic control (start, stop, data display, and printing and saving functionality), while the other three subroutines allow significant enhancements, such as the ability to change your program's variables without interupting program execution.

D.2 CFD Program Requirements

ICFD Tools provides a powerful graphical user interface through a simple set of subroutines. It is able to do this by making certain basic assumptions about your CFD program. The following list sets forth the requirements your program must meet in order for you to use *ICFD Tools:*

- You must have at least one array which holds spacial coordinates (i.e. x-axis values), which.

- You must have at least one array which holds the dependent variable or variables for the problem you are solving (e.g. an array of flow velocities).

- Your program must solve for these dependent variables by executing a DO-loop or similar loop in which the dependent variables are repeatedly updated.

- For two-dimensional calculations, only "structured grids" are allowed (if you don't know what this means, your grid is probably a structured one).

If your program meets the above criteria, you are ready to enhance it with *ICFD Tools*!

D.3 Getting Started

Only three of the six subroutines are required to begin using *ICFD Tools*: **ICFDINIT, ICFDZONE,** and **ICFD**. The following sections explain how to use these subroutines.

D.3.1 ICFDINIT

ICFDINIT must be called only once, after you dimension your arrays, and before your program enters its main DO-loop.

The FORTRAN declaration for **ICFDINIT** would look like this:

```
SUBROUTINE ICFDINIT( NARRAY, NTYPE, ANAME1, ANAME2, ...)
INTEGER NARRAY, NTYPE
CHARACTER*(*) ANAME1, ANAME2, ...
```

The ellipses (...) indicate that **ICFDINIT** can take a variable number of arguments. NARRAY is the number of array names which follow; because you must have at least an array of spacial coordinates and a dependent variable array, NARRAY must be at least 2. NTYPE indicates how the arrays are dimensioned - NTYPE = 1 indicates the arrays are integer arrays, NTYPE = 2 indicates single-precision floating point (REAL*4), and NTYPE = 3 indicates double precision (REAL*8). The character strings ANAME1, ANAME2, ...are the names for your variable arrays. For convenience, you should have your spacial array(s) (e.g. your array of X values) first. Note that these names do not need

to be the same as the names you chose for your arrays when you dimensioned them. The following are some examples:

```
PROGRAM MYPROG
DIMENSION X(100), U(100)
CHARACTER*20 XNAME, UNAME
XNAME = 'X'
UNAME = 'Flow Velocity'
CALL ICFDINIT(2, 2, XNAME, UNAME)
      .
      .
      .
DO 1 I = 1, 500
      .
      .
      .
```

In the above example, the character strings are explicitly dimensioned before being passed to **ICFDINIT**. This actually isn't necessary. The next example shows a simpler way:

```
PROGRAM MYPROG
DOUBLE PRECISION X(100), RHO(100), MOM(100), ENERGY(100)
CALL ICFDINIT( 4,3, 'X', 'Density', 'Momentum', 'Energy')
      .
      .
DO 1 I = 1, 500
      .
      .
```

Note that any characters can be used within the character strings, including special characters such as '?' and '-'. It is not a good idea to begin a variable name with a special character, however -- that will cause problems with TECPLOT™, if you choose to save your data to a TECPLOT-compatible text file (more on that later).

D.3.2 ICFDZONE

ICFDZONE must be called once for each zone, after **ICFDINIT** and before your main loop. It provides *ICFD Tools* with your program's arrays (or for you Computer Science folks, pointers to the first element of each array). If you don't understand what is meant by a "zone", then you probably have only one, and need call this routine only once.

The FORTRAN declaration for **ICFDZONE** would look like this:

SUBROUTINE ICFDZONE(ZNAME, IDIM, JDIM, IMIN, IMAX, JMIN, JMAX, ARRAY1, ARRAY2, ...)
CHARACTER*(*) ZNAME
INTEGER IDIM, JDIM, IMIN, IMAX, JMIN, JMAX
DIMENSION ARRAY1(IDIM,JDIM), ARRAY2(IDIM,JDIM), ...

ZNAME is the name you by which you wish this zone to be called. IDIM and JDIM are the sizes you have dimensioned your arrays (note: all arrays in a given zone must be dimensioned the same size and type, e.g. real*4). IMIN, IMAX, JMIN, and JMAX are the index ranges you wish to be plotted. These can be changed during program execution, as will be described later. The following is an example:

```
DIMENSION X(100,50), Q(100, 50, 3)
        .
        .
        .
CALL ICFDINIT(4, 2, 'X', 'Density', 'Momentum', 'Energy')
CALL ICFDZONE('Zone 1', 100, 50, 1, 100, 1, 50, X, Q(1,1,1), Q(1,1,2), Q(1,1,3))
DO 1 I = 1, 500
        .
        .
        .
```

As you probably noticed, this example includes a 3-dimensional array. Although *ICFD Tools* can only handle 1- or 2-dimensional data, some programmers find it more convenient to store all of their dependent variable arrays in one higher-dimensional array. If your program does this, the **first** index or indices in the array declaration must be the **length(s)** of each array, i.e. IDIM (and JDIM, for 2-D arrays). This ensures that each array is stored in a contiguous block of memory. In place of the names of individual arrays, then, pass the first element of each array, as shown above. Please also note that each array must be the same size. Passing arrays smaller than IDIM by JDIM will cause serious problems!

D.3.3 ICFD

ICFD must be called every iteration within your main loop. It is normally best to call it at the very beginning of the loop. It must, however, be called at a point where your data is ready for viewing.

The FORTRAN declaration for **ICFD** would look like this:

```
SUBROUTINE ICFD(T, TFINAL, ITER, IFINAL)
REAL T, TFINAL
INTEGER ITER, IFINAL
```

T is the problem time, as calculated by you program, and TFINAL is the problem time at which you wish **ICFD** to halt execution. Pass 0.0 for these values if you do not wish to use them. ITER is the iteration number (probably the index of your main loop), and IFINAL is the iteration at which you wish **ICFD** to halt execution. Pass 0 for these values if you do not wish to use them. Note that you can pass positive values for T or ITER while still passing zeros for TFINAL or IFINAL, although if you pass zeros for both TFINAL and IFINAL, **ICFD** will not be able to halt execution for you. Also note that the values passed are REAL*4. This is an important distinction, and will require special attention if your default real values are double precision. The following is an example:

```
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
DIMENSION X(100), Q(100, 3)
REAL*4 TM
        .
        .
        .
CALL ICFDINIT( 4,3, 'X', 'Density', 'Momentum', 'Energy')
CALL ICFDZONE('Zone 1', 100,1,1,100,1,1,X,Q(1,1),Q(1,2),Q(1,3))
I = 0
DO 1 WHILE (.TRUE.)
CALL ICFD(0.0, 0.0, I, 500)
        .
        .
        .
I = I + 1
        .
        .
```

Note that we passed REAL(0.0) instead of just 0.0 for TFINAL. This is only necessary when your FORTRAN compiler promotes floating-point constants to double precision, but we show it here just in case you need to use it. And why did this example use an infinit DO WHILE loop? Since we passed an IFINAL of 500, **ICFD** will halt execution when I equals 500, so the only thing we need to ensure is to increment I each loop.

D.4 Compiling and Running Your ICFD Application

Once you have inserted **ICFDINIT** and **ICFD** into your program, the following command will compile and link your program. You must ensure that the file *libicfd.a* resides in your current working directory. Insert the name of your FORTRAN source file in place of *myprog.f:*

```
f77  myprog.f  -L./ -licfd -lXm  -lXt  -lX11
```

You may also have to use the option -lPW (right after the -lX11 above) on some machines. The compilation and linking will take a minute or two. From any workstation or terminal running X Window, you can then run your program:

```
a.out
```

After a short delay, a new window should appear on your screen, looking something like this:

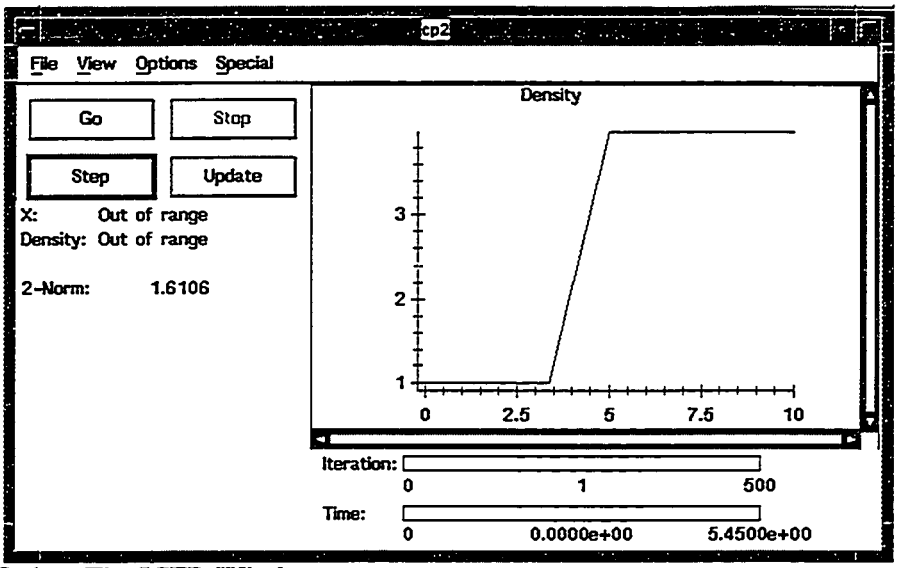

Figure D-1: The ICFD Window

The window has several different areas, including four control buttons, a panel for point data display, a data plot, a timeline panel, and a menu bar.

The menu bar lies across the top of the window and offers four pull-down menus. The **File** menu allows you to save data to a TECPLOT™-compatible text file, to print the currently displayed plot to a PostScript™ printer or to save it to a PostScript disk file, and to exit the program. Selecting the **Save data as...** or the **Print to disk...** option brings up a dialog box which allows you to specify the name of the file you wish to create or overwrite. The Print option sends PostScript output to the printer specified by your PRINTER environment variable, or the default printer if PRINTER is undefined. When you have the UNIX™ C-shell system prompt (a % prompt), you can set this variable with

setenv PRINTER xxxxx

Under the Bourne shell or Korn shell (a $ prompt) simply enter

PRINTER=xxxxx

where xxxxx is the name of the printer. See the /etc/printcap file on your computer for a list of printers.

The **View** menu allows you to display various plots of any of your arrays. For 2-D data, both contour and line plots are available, and a contour plot will be the default. For 1-D data, only line plots are available. Selecting one of the variables causes that variable

to be plotted. The **View** menu also has four scaling options, **Zoom, Fit, Y-Compressed fit,** and **Specify scale. Fit** fits the plot just inside the boundaries of the plot area, while **Y-Compressed fit** (only available for line plots) will compress the plot in the y direction. **Zoom** can only be selected after a box has been drawn on the plot. To draw a box, position the cursor in the plot area, press and hold down the left-hand mouse button, drag the cursor in the desired direction, and release the mouse button. A box will be drawn, to which you can then zoom by selecting **Zoom** from the view menu.

Selecting **Specify Scale** pops up the following dialog box:

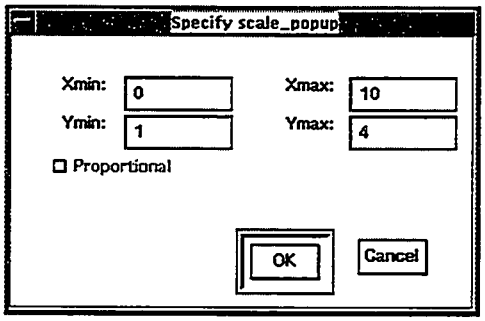

Figure D-2: The Specify Scale Dialog

This dialog allows you to specify the area to be viewed in the current plot. The 'Proportional' toggle button, when selected, forces the plot to have equal scales in the X and Y directions. Because this is only meaningful for contour plots, it is not available when the current plot is a line plot.

The **Options** menu offers three options. **Periodicity** allows you to adjust the update interval, that is, whether to redraw the plot every iteration (the default), or to "skip" a specified number of iterations between updates. Selecting this option brings up a dialog box which allows you to change this update interval. Use of this option can greatly increase the speed of execution by reducing the number of screen redraws.

For each variable, **ICFD** maintains information which describes one line plot and one contour plot. Making changes to the line plot for a particular variable does not affect the contour plot for that same variable, and vice versa; they are two separate entities. The following options from the **View** menu allow you to customize these plots. Their use may not be necessary, however. The index information you provided in your call(s) to **ICFD-**

ZONE, combined with some intelligent guessing by **ICFD,** may provide you with the plots you desire.

The **Options -> Line plot options** menu option brings up the dialog box show in figure D-3, which allows you to specify various parameters for line plots.

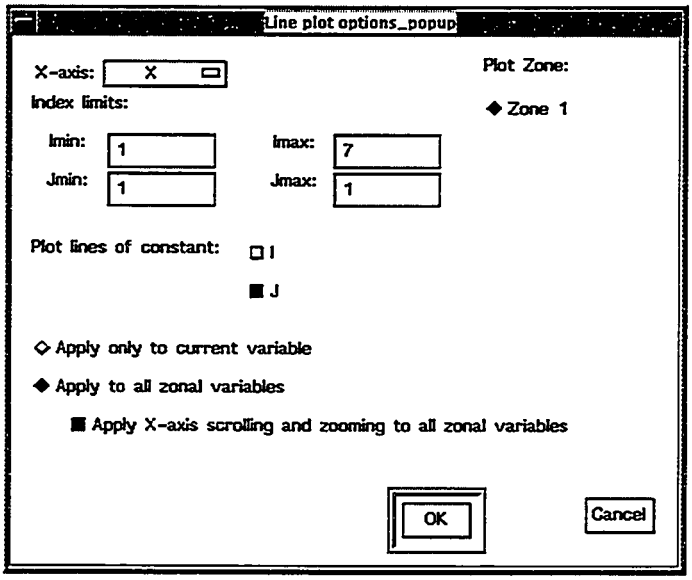

Figure D-3: The Line Plot Options Dialog

The X-axis pop-up menu allows you to specify the variable versus which the line plots will be plotted (i.e. the X-axis variable). Below this, you may specify the minimum and maximum indices of your arrays to be plotted. The default values which appear for Imin and Imax are those you supplied in your call to **ICFDZONE.** Entering a value of 0 for Imax or Jmax will use the maximum available size (Idim and Jdim, which you provided in your call to icfdzone). If your arrays are 1-D, do not change the values of Jmin and Jmax! Further below, you may specify line of constant I, constant J, or both. To the right, you may specify which zone is to be plotted. In this example, there is only one zone, called "Zone 1", so no other selections are available. Finally, you may choose whether to apply these settings to the line plots of all zonal variables, or just the currently plotted variable. If you

choose to apply them to all zonal variables, you have the further option of specifying uniform X-axis scrolling and zooming. Selecting this option automatically adjusts the X-axis range to be viewed for all line plots when you adjust it for the current plot (by scrolling, zooming, or using the **View -> Specify** dialog box).

The **Options -> Contour plot options** menu option is only available when your zonal data is two-dimensional. It pops up a sub-menu with two or more options. The first is called **All zones**, and pops up the dialog box in figure D-4.

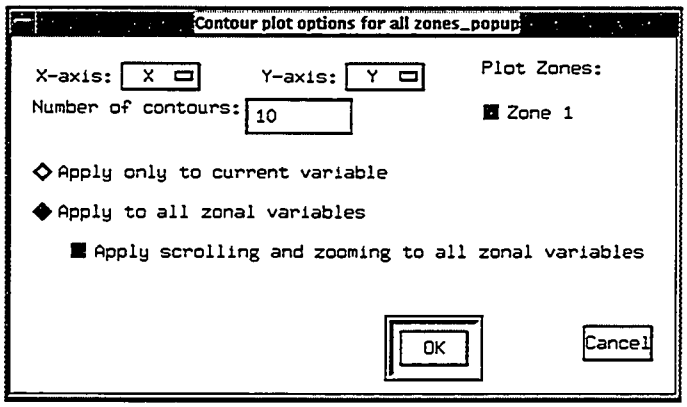

Figure D-4: The 'All Zones' Dialog Box

This dialog allows you to specify information which affects the contour plots of all zones. Similar to the line plot dialog, you can select the X- and Y- axis variables, and select whether to apply these setting to all zonal variables, or only to the current one. For contour plots, uniform scrolling and zooming affects both the X- and Y- axis locations and scales, that is, with this option selected, the area plotted will automatically be the same for all zonal variables. You can also set the number of contours to be plotted, and which zones will be plotted. Note that, unlike line plotting, contour plotting allows simultaneous plotting of multiple zones.

The **Options -> Contour plot options** submenu also contains an entry for each zone. Selecting one of these displays the following dialog box:

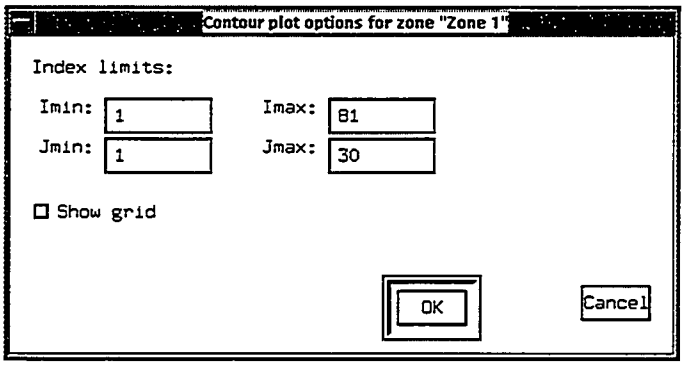

Figure D-5: The Zonal Contour Plot Options Dialog

As in the line plot dialog, this dialog allows you to specify the index limits to be plotted for each zone. The default values which appear for the indices are those you supplied in your call(s) to **ICFDZONE.** You can also choose whether to plot the grid for this zone. Note that each of these dialogs applies only to its own zone, and all zonal variables for that zone are affected by the dialog.

Once you have defined the plots you desire for each variable, you can easily switch between them using the **View** menu options.

The **Special** menu is empty. Using subroutine calls described in the next section, you will be able to add your own items to these menus.

Below the **File** menu are the control buttons. These allow you to control the execution of your program. They are activated by positioning the cursor in the desired button, and pressing and releasing the left-hand mouse button (hereafter referred to as a "left-click"). **Go** begins execution of your program, **Stop** halts it, and **Step** will execute one iteration. If you have specified an extremely long update interval, **Update** will force an update of the plot without halting program execution.

Below the control buttons is an area where a specific value of each of your variables is displayed. This acts as a "probe" in the flow, and as the solution changes, these values will be updated. To change the location of the probe, left-click once in the data plot. For a line plot, the (interpolated) values of your variables corresponding to the X-axis location of the click will be displayed. For a contour plot, the values will correspond to the the X- and Y-axis locations of the click.

The data plot is accompanied by scroll bars, which allow you to scroll around the plot if you zoom in on a particular area.

Finally, the timeline panel draws up to two timelines to graphically indicate the progress of your CFD code. If your pass zeros for T, TFINAL, ITER, and IFINAL, this panel will be blank. If you pass a value for either T or ITER, but pass a zero for the corresponding FINAL value, only that value will be displayed. If you pass both T and TFINAL (or ITER and IFINAL), a timeline will be drawn for it.

When you press the **Go** button, your program will begin executing, and the updated solution will be displayed every iteration, along with updates to the timelines, if any, and the probe values. If a timeline is displayed (i.e. if you are passing TFINAL or IFINAL to **ICFD**), **ICFD** will ensure that these values are not exceeded -- it will halt execution at the appropriate time or iteration number. This allows you to print or save your solution before quitting, or possibly to execute more advanced features described in the next few sections. You can then terminate the program by selecting **Exit** from the **File** menu.

An important point to note is that all plotting each iteration is done when your program calls **ICFD**. Therefore, it is important to call it at a point in your program where your data is displayable (and meaningful).

D.5 Enhancing Your ICFD Application

Once you are comfortable with the basic functions provided by **ICFDINIT**, **ICFDZONE**, and **ICFD**, you are ready to begin enhancing your user interface. **ICFDVARIABLE** allows you to dispiay the value of a variable (not an array) your program calculates. **ICFDDIALOG** allows you to create a dialog box and insert it into the **Special** menu. This will allow you to change the values of your program parameters without halting execution of the program. **ICFDSUBROUTINE** allows you to make a FORTRAN subroutine you write directly callable from the user interface, inserting its name into the **Special** menu.

D.5.1 ICFDVARIABLE

ICFDVARIABLE prints the name and value of a variable you supply just below the probe values. It must be called once for each variable you wish to display, before your main loop. It can be called before or after **ICFDINIT**.

The FORTRAN declaration for **ICFDVARIABLE** would look like this:

```
SUBROUTINE ICFDVARIABLE(NAME, ITYPE, VALUE)
CHARACTER *20 NAME
INTEGER ITYPE
-------------EITHER-------------
INTEGER VALUE
----------------OR----------------
REAL VALUE
----------------OR----------------
DOUBLE PRECISION VALUE
```

As before, NAME does not have to be the same as the variable's name in your program; call it anything you like. ITYPE indicates whether VALUE is an integer or a real number. Pass 1 if VALUE is an integer, 2 if it is a real number, and 3 if it is double precision. The following shows an example with two different variables to be displayed:

```
PROGRAM EULER1D
REAL ANORM
INTEGER IMAX
        .
        .
        .
CALL ICFDVARIABLE("2-NORM", 2, ANORM)
CALL ICFDVARIABLE("IMAX", 1, IMAX)
        .
        .
        .
DO 1 I = 1, 500
        .
        .
        .
```

D.5.2 ICFDDIALOG

ICFDDIALOG allows you to set up a dialog box, callable from the **Special** menu, which allows you to change the values of variables in your program. It must be called once for each dialog box you wish to implement, before your main loop. It can be called before or after **ICFDINIT**.

The FORTRAN declaration for **ICFDDIALOG** would look like this:

```
      SUBROUTINE ICFDDIALOG(DNAME, NITEM, NAME1, ITYPE1, VALUE1,
     & NAME2, ITYPE2, VALUE2, NAME3, ITYPE3, VALUE3, ...)
      CHARACTER *20 DNAME, NAME1, NAME2, NAME3, ...
      INTEGER NITEM, ITYPE1, ITYPE2, ITYP3, ...
-------INTEGER, REAL, DOUBLE PRECISION, or LOGICAL*4 declarations for VALUEs------
```

DNAME is a name for the dialog box, which will also be inserted into the **Special** menu. NITEM indicates how many dialog items follow it. The remaining parameters are groups of three: a NAME, an ITYPE, and a VALUE. NITEM represents how many of these **groups of three parameters** there are. The NAMEs are names you choose for each value to be displayed in the dialog box. Simliar to ICFDVARIABLE, the ITYPE parameters indicate whether their associated VALUE is an integer or a real variable, but also allows logicals and option menu definitions; 1 for an integer, 2 for a real variable, 3 for double precision, 4 or 5 for logical*4, 6 for an option menu, and 7 for the options to be placed in the option menu (more on this later). Logical variables will be represented by toggle buttons. For an ITYPE of 4, the toggle buttons will be the one-of-many type, which allow only one of a given group of toggle buttons to be set (or .TRUE.) at any give time. Successive dialog items of ITYPE 4 will be considered one group of toggle buttons. The next example demonstrates this for two buttons. An ITYPE of 5 indicates a logical variable which will be represented by an n-of-many toggle button (any number of this type of button may be set in a given group). The VALUE parameter is, of course, the variable itself.

ICFDDIALOG adds DNAME to the **Special** menu. When you select this menu item, a dialog box appears with the names and the values of the variables you passed it. You can

then edit the values using the mouse and the keyboard (left-click on the value you wish to change). Left-clicking on **Ok** puts the changes into effect, while left-clicking on **Cancel** makes no changes to the variables. An example follows:

```
      PROGRAM EULER1D
      REAL COURANT, TFINAL
      INTEGER NSTEPS
      LOGICAL SHOCK, NOZZLE
      VOLATILE COURANT, TFINAL, NSTEPS, SHOCK, NOZZLE
C      See the explanation below for the 'volatile' declaration
            .
            .
            .
      CALL ICFDDIALOG('Values', 5, 'Courant', 2, courant,
     & 'Number of iterations', 1, nsteps,
     & 'Shock tube', 4, shock,
     & 'Nozzle', 4, nozzle,
     & 'Final time', 2, tfinal)
            .
            .
            .
      I = 1
      DO 1 WHILE(.TRUE.)
            .
            .
            .
```

When 'Values' is selected from the Special menu, the above example would produce a dialog box like that pictured below:

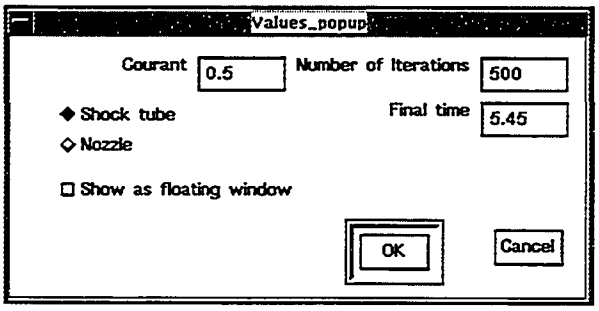

Figure D-6: A Dialog Box

Notice that there is an extra dialog item labeled "Show as floating window." This item is automatically added to each dialog you define. Selecting this item creates a movable "floating window" in the plot area which displays the dialog's values. This floating win-

dow can be moved to any portion of the plot by clicking and dragging it with the pointer. If you choose the **File -> Print** menu option while one or more floating windows are displayed, they will be printed as well.

Also notice in this example that the variable which will control the number of iterations performed is a parameter in the **ICFDDIALOG** call. Using this mechanism, the number of iterations to be performed (the IFINAL parameter in the **ICFD** call) can be changed during program execution. If you use a DO-loop, the number of iterations is fixed at the beginning of the loop's execution, and cannot be changed. Changing your program's main loop to a DO WHILE-loop eliminates this problem, and allows you added flexibility. The example also shows problem time, which can, of course, also be used to control program execution. Any other program parameters can also be put into dialog boxes, to allow them to be changed during program execution.

An option menu is very similar to a pull-down menu, except that it does not appear in the menu bar. In this case, it will appear in your dialog box. The ITYPE value 6 lets you define an option menu, and the ITYPE value of 7 (which must follow an ITYPE 6!) lets you specify the options to appear in the menu, and the integer values which correspond to each option. Consecutive options (ITYPE 7) will all be added to the option menu (ITYPE 6) which precedes them. Let us say that you have an integer variable called 'method', which takes the value 1, 2, or 3 (these values have some meaning to your program). Adding the following code fragment to your call to **ICFDSUBROUTINE** will create an option menu in the resulting dialog box:

```
... & 'Method', 6, method,
    & 'Forward Euler', 7, 1,
    & 'Backward Euler', 7, 2,
    & 'MacCormack', 7, 3, ...
```

When this dialog is displayed, it will have an option menu labeled 'Method', and there will be three options available in the menu, 'Forward Euler', 'Backward Euler', and 'MacCormack'. When you click **OK**, the value corresponding to the selected option will be assigned to the variable 'method' in your program. For example, if you select 'Backward Euler' and click **OK**, method will be given the value 2.

One potential hitch with dialogs is with optimizing compilers (such as f77 on DECstations). The optimizer can play "tricks" with variables which could interfere with the dialog box's operation, preventing changes from actually being made to the variables. The VOLATILE declaration tells the optimizer to leave these variables alone, ensuring proper functioning of the dialog box. This declaration is not required, and is not always necessary, but it is a good idea where available.

Generally speaking, though, if you want to reset the iteration number or the problem time to zero, you are also going to want to reset the dependent variables to their original state. This requires a subroutine call, specifically, a call to whatever subroutine you have written which initializes your dependent variable arrays. **ICFDSUBROUTINE** provides this capability, and will be discussed next.

One final note is a strange bug which, for reasons too in-depth to cover (i.e. the author doesn't know why), cause problems with **ICFDDIALOG**. The bug only occurs when a dialog is named 'Parameter'. So don't name your dialog 'Parameter'. 'Parameters' is ok, just not 'Parameter'. 'Nuff said?

D.5.3 ICFDSUBROUTINE

ICFDSUBROUTINE must be called once for each user-callable subroutine you wish to implement, before your main loop. It can be called before or after **ICFDINIT**.

The FORTRAN declaration for **ICFDSUBROUTINE** would look like this:

```
SUBROUTINE ICFDSUBROUTINE(NAME, SUBRTN, NPARAM, PARAM1, PARAM2, ...)
CHARACTER *20 NAME
EXTERNAL SUBRTN
INTEGER NPARAM
----------PARAMs can be any type of FORTRAN variable----------
```

NAME is the subroutine name as you wish it to appear in the Special menu. SUBRTN is the actual subroutine name in your CFD program. NPARAM is the number of parameters you are supplying for your subroutine (the number of parameters which follow NPARAM). The PARAMs are the parameters that you want to be passed to your subroutine. When you select NAME from the Special menu, SUBRTN is called with all of the

PARAMs as arguments. Confused? This is not a simple concept, but the following example might help.

```
      PROGRAM EULER1D
      EXTERNAL INIT
      DIMENSION X(100), U(100)
      INTEGER MAXITER
            .
            .
            .
      CALL ICFDSUBROUTINE('Reset', INIT, 3, MAXITER, X, U)
            .
            .
            .
      DO 1 WHILE(.TRUE.)
            .
            .
            .
1     CONTINUE
      END

      SUBROUTINE INIT(MAXITER, X, U)
      INTEGER MAXITER
      DIMENSION X(100), U(100)
      MAXITER = 0
            .
            .
            .
      RETURN
      END
```

The EXTERNAL declaration tells the FORTRAN compiler that you have a subroutine in your program that you want to pass as an argument to another subroutine. **ICFD-SUBROUTINE** makes this subroutine available as a menu option, inserting, in this case, 'Reset', into the **Special** menu. For this example, selecting 'Reset' from the **Special** menu would call INIT with the parameters MAXITER, X, and U. A warning here is that, due to compiler optimization, it may not be safe to pass constants (i.e. actual numbers rather than variables) in the list of arguments for your subroutine. It would be better to declare a variable, assign it the desired value, and pass it as a subroutine parameter.

D.6 Tying It All Together

To further clarify the uses of the six subroutines which have been discussed, a complete example CFD program is presented here which makes use of all of these subroutines. The program solves the one-dimensional, first-order wave equation, with U as a function of X and time. If you are having difficulty implementing any of the subroutines, please examine this example, or type it in and run it.

```
      program wave
      dimension x(100), u(100), du(100)
      integer iter, niter, imax
      real anorm, courant, speed, t, tfinal, dx, dt
      external init
      volatile t, iter, niter, courant, speed, tfinal
c Remember that the 'volatile' declaration is not applicable to all compilers

      niter = 500
      imax = 100
      speed = 1.0
      tfinal = 3.14159
      courant = 0.5
      call init(iter, t, x, u)
      dx = x(2) - x(1)
      call icfdinit(2, 2, 'X', 'Flowspeed')
c Remember the '2' in icfdvariable denotes a real*4 variable
      call icfdvariable('2-Norm', 2, anorm)
      call icfddialog('Parameters', 4,
     &          'Iterations:', 1, niter,
     &          'Final Time:', 2, tfinal,
     &          'Courant number:',2, courant,
     &          'Wave speed', 2, speed)
      call icfdsubroutine('Reset', init, 4, iter, t, x, u)
      call icfdzone('Zone 1',100,1,1,100,1,1,x,u)
      do 1 while(.true.)
            call icfd(t, tfinal, iter, niter)
            du(1)= -courant/2.*(u(2)-u(imax-1))
     &              +courant**2/2.*(u(2)-2.*u(1)+u(imax-1))
            do 2 i=2,imax-1
                  du(i)= -courant/2.*(u(i+1)-u(i-1))
     &                    +courant**2/2.*(u(i+1)-2.*u(i)+u(i-1))
2           continue
            du(imax)=du(1)
            anorm = 0.0
            do 3 i=1,imax
                  u(i)=u(i)+du(i)
                  anorm = anorm + du(i)**2
```

```
 3              continue
                anorm = sqrt(anorm)
                iter = iter + 1
c dt is recalculated every iteration because the dialog box
c could change it!
                dt = courant * dx / speed
                t = t + dt
 1      continue
        end

        subroutine init(iter, t, x, u)
        integer iter
        real t
        dimension x(100), u(100)
        iter = 0
        t = 0.0
        pi=2.*asin(1.)
        do 1 i=1,100
                x(i)=float(i-1)*pi/99.
                if(x(i).lt.0.5) then
                        u(i)=0.0
                elseif(x(i).le.1.0) then
                        u(i)=1.0
                else
                        u(i)=0.0
                endif
 1       continue
        return
        end
```

Vita

David E. Taflin was born on August 4, 1962 in Munich, Germany. He graduated from Anoka Senior High School, Anoka, Minnesota, in May, 1980. He attended Rensselaer Polytechnic Institute from August, 1980 until May, 1984, obtaining a Bachelor of Science degree in aeronautical engineering. He was then commissioned an Ensign in the United States Navy. He served aboard USS O'Callahan (FF-1052) and USS Pledge (MSO-492). He left active duty in 1989, and attended graduate school at the University of Washington from 1989 until 1995. He received a Master of Science in Aeronautics and Astronautics (MSAA) in 1991, and a Doctor of Philosophy in 1995.