1995

Numerical solutions to the non-linear Euler equations

Elizabeth S. Milanovich
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Milanovich, Elizabeth S., "Numerical solutions to the non-linear Euler equations" (1995). *Master's Theses*. 1090.

DOI: <https://doi.org/10.31979/etd.rrhz-fx3h>

https://scholarworks.sjsu.edu/etd_theses/1090

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

**A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600**

**NUMERICAL SOLUTIONS TO THE
NON-LINEAR EULER EQUATIONS**

A Thesis

Presented to

**The Faculty of the Department of Mathematics and Computer Science
San Jose State University**

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Elizabeth S. Milanovich

August, 1995

UMI Number: 1375714

UMI Microform 1375714

Copyright 1995, by UMI Company. All rights reserved.

**This microform edition is protected against unauthorized
copying under Title 17, United States Code.**

UMI

**300 North Zeeb Road
Ann Arbor, MI 48103**

© 1995

Elizabeth S. Milanovich

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF
MATHEMATICS AND COMPUTER SCIENCE

Mohammad Saleem

Dr. Mohammad Saleem

Leslie Foster

Dr. Leslie Foster

Howard Swann

Dr. Howard Swann

APPROVED FOR THE UNIVERSITY

Serena H. Stanford

ABSTRACT

NUMERICAL SOLUTIONS TO THE NON-LINEAR EULER EQUATIONS

BY ELIZABETH S. MILANOVICH

This Master's thesis looks at numerical solutions to the non-linear one-dimensional Euler equations. These equations are derived for airflow through a nozzle. To solve these equations numerically, the Flux Splitting Algorithm is used. This algorithm enables two different finite-differencing schemes to be used when discretizing the space term of the Euler equations. Once a solution is obtained, three procedures that accelerate convergence of the numerical solution will be presented. One procedure is Successive Over Relaxation. Although this procedure is primarily used for linear systems, it is used to see if it will accelerate the solution of this non-linear system. The other two are eigenvalue annihilation procedures. These procedures will eliminate the largest real eigenvalue and the dominant pair of complex eigenvalues of the system. An analysis is made to determine the best acceleration procedure for this non-linear system. A savings of 10% to 29% in the number of iterations was achieved.

ACKNOWLEDGMENT

I would like to thank Dr. Saleem for his tireless efforts put forth towards this thesis. I appreciate his guidance and patience in seeing me through this endeavor. I would also like to thank Drs. Foster and Swann for their suggestions and contributions toward this thesis. With their help, this thesis has become a sound document. My eternal gratitude goes out to these three people for their support.

TABLE OF CONTENTS

INTRODUCTION	1
CHAPTER 1 THE EQUATIONS	3
2 FLUX SPLITTING METHOD	11
3 THE ALGORITHM	16
4 SET-UP AND SOLUTION	30
5 ACCELERATION METHODS	36
REFERENCES	48
APPENDIX A - PROGRAM LISTING	50
APPENDIX B - NOMENCLATURE	60

LIST OF TABLES

Table 5.1	Comparison of the Three Acceleration Methods	44
-----------	--	----

LIST OF FIGURES

Figure 1.1	Pressure on a Particle Through a Tube	3
1.2	Particle Through a Converging-Diverging Nozzle	5
1.3	Volume of the Fluid Particle	8
3.1	Volume for Stability	19
3.2	Outline of Iterative Procedure	28

4.1	View of the Nozzle	30
4.2	Initial Distribution	32
4.3	Converged Solution	33
4.4	Converged Solution for Mach Number with Different Values for Δx	34
5.1	Relaxation	37
5.2	Dominant Eigenvalue Estimation	39
5.3	Single Eigenvalue Annihilation	40
5.4	Estimated Dominant Eigenvalues	43
5.5	Two Eigenvalue Annihilation	44
5.6	Two Eigenvalue Annihilation in Early Stages	45
5.7	Two Eigenvalue Annihilation at Iterations 150, 210 and 240	46

INTRODUCTION

Over the last thirty years, research in the field of computational fluid dynamics (or CFD) has made many advances. Much of the progress has been due to the technological expansion of computers. What was achievable in 1970 only on a mainframe can in 1995 be done on a personal computer. With these improvements, many different algorithms have been developed to use the power of computers.

This history of using numerical methods to solve the non-linear Euler equations for fluid flow has not been without conflict. During the 1960's, a prevailing attitude was that using the computer to solve CFD problems was meaningless number crunching contributing no new physical insight [13]. Nevertheless, algorithms were developed. During the early stages, one effective algorithm was based on the Soviet scientist Godunov's method, which solves a nonlinear Riemann problem at each cell interface. However, this method was only first order accurate and slow.

As computer use became more accepted by the CFD community, more efficient algorithms were developed. The Method of Characteristics was a favorite algorithm for solving the Euler equations during the 1970's. This method lost favor when computers became powerful enough to calculate solutions of multidimensional problems. This advancement led the way for a number of characteristics-based algorithms to be developed. One method, created by G. Moretti, was the "Lambda" method. Another, the Flux Splitting Method, was introduced in 1979 by J. L. Steger and R. F. Warming [1]. Since this time, other methods have been presented. The flux split procedures seem to be the favorite for multidimensional problems.

In this thesis, the Flux Splitting Method will be used to solve the Euler equations. The one-dimensional Euler equations are used to model airflow through a nozzle. This

nozzle can have many applications. For instance, airflow through a jet engine can be simulated by using an ideal nozzle. Also, the thrusters on the space shuttle can be simulated by using a nozzle.

Although the Euler equations have been solved before, there is always room for improvement. As P. Richardson, et al. state in their paper [15],

...the NASA program will concentrate on code accuracy, innovative ideas, faster codes, and improved test techniques and measurement technology.

There is a need for accurate measurements when trying to simulate flight vehicles whether they are commercial carriers, military fighter planes or space exploration vehicles. This thesis will look at accurate and faster converging techniques for the Euler equations.

In Chapter 1, the one-dimensional Euler equations will be derived for inviscid, compressible airflow through a nozzle. Chapter 2 develops the Flux Splitting Method. The algorithm used to implement the Flux Splitting Method is discussed in Chapter 3. Within the algorithm, the initial conditions and boundary conditions will be discussed. In Chapter 4, a specific nozzle application will be set-up and a steady-state solution will be calculated. In Chapter 5, various acceleration methods will be implemented to speed-up the convergence of the solution. Also, an analysis of these acceleration methods will be made in order to determine the "best" procedure.

CHAPTER 1

THE EQUATIONS

The Euler equations are three different equations. These equations are derived from the theory of inviscid compressible fluid. The equations are usually written for unsteady, compressible flows in two-dimensions, using Cartesian coordinates in conservation form. However, the one-dimensional case reflects more closely the flow of air through a nozzle. The assumptions are that external forces (gravity, etc.) and external heat are negligible for this application. The equations are derived from the mass equation, the momentum equation and the energy equation.

Figure 1.1 Pressure on a Particle Through a Tube

The first equation is derived from the law of conservation of mass [12]. In figure 1.1, the mass enclosed between the incoming cross-section and the outgoing cross-section is $\rho A \Delta x$, where ρ is the density, A is the cross-sectional area of the nozzle and Δx is the incremental distance along the nozzle. The mass increases at the rate $\frac{\partial}{\partial t}(\rho A \Delta x)$, where t is time. Since mass is conserved, this rate must be equal to the flow, with velocity u ,

through the incoming side minus the flow through the outgoing side. Hence, the net inflow is

$$-\frac{\partial}{\partial x}(\rho u A) \Delta x = \frac{\partial}{\partial t}(\rho A \Delta x). \quad (1.1)$$

Both A and Δx do not depend on time. Therefore, they may be brought outside the derivative and divided through to give

$$-\frac{1}{A} \frac{\partial}{\partial x}(\rho u A) = \frac{\partial}{\partial t} \rho. \quad (1.2)$$

Upon further rearranging, equation 1.2 becomes

$$\frac{\partial}{\partial t} \rho + \frac{1}{A} \frac{\partial}{\partial x}(\rho u A) = 0. \quad (1.3)$$

This equation is called the continuity equation.

The second equation is the momentum equation. This equation is a linear combination of Euler's equation and the continuity equation (equation 1.3). Euler's equation is derived from Newton's second law of motion. The law (force = mass $\times$ acceleration) is applied to a flowing fluid. In particular, the acceleration of a "delta" particle flowing through a tube will be observed.

The acceleration is the time rate of change of velocity and is due to two effects. First, since conditions vary along the tube, there is a velocity gradient $\frac{\partial u}{\partial x}$ in the direction of the flow. The rate of change of velocity is proportional to this gradient and to the speed with which the particle moves through it. Thus, the acceleration due to "convection" through the velocity gradient is [12]

$$u \frac{\partial u}{\partial x}.$$

Second, since the flow is non-steady, the conditions at any given section may fluctuate. This fluctuation gives rise to a nonstationary acceleration term,

$$\frac{\partial u}{\partial t}.$$

The acceleration of the particle is given by adding the convection term and the nonstationary term. Thus, the acceleration is

$$a_x = \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x}. \quad (1.4)$$

Now, the force on the particle needs to be determined. The force will be a force per unit mass so that the mass term in Newton's law can be neglected. Considering a particle as in figure 1.1, the force in the x -direction is determined to be (assuming non-viscous flow) $-\frac{\partial p}{\partial x}(\Delta x A)$, where p is the pressure. Since the volume of the particle is

$\Delta x A$, divide the force above by the volume to get the force per unit volume equal to $-\frac{\partial p}{\partial x}$.

Dividing through by the density gives the force per unit mass,

$$f_x = -\frac{1}{\rho} \frac{\partial p}{\partial x}. \quad (1.5)$$

This force is valid for any shape of the particle. In particular, the particle can be reshaped into one that passes through a converging-diverging nozzle (figure 1.2)

Figure 1.2 - Particle Through a Converging-Diverging Nozzle

Now, the acceleration and force (equations 1.4 and 1.5) can be substituted into Newton's second law for unit mass ($a_x = f_x$) to give

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = -\frac{1}{\rho} \frac{\partial p}{\partial x}. \quad (1.6)$$

This equation is called Euler's equation.

Multiply Euler's equation (equation 1.6) by ρA and the continuity equation (equation 1.3) by u to get, respectively,

$$\rho A \frac{\partial u}{\partial t} + \rho u A \frac{\partial u}{\partial x} = -A \frac{\partial p}{\partial x}$$

and

$$u \frac{\partial}{\partial t}(\rho A) + u \frac{\partial}{\partial x}(\rho u A) = 0.$$

Adding these two equations gives

$$\rho A \frac{\partial u}{\partial t} + u \frac{\partial}{\partial t}(\rho A) + \rho u A \frac{\partial u}{\partial x} + u \frac{\partial}{\partial x}(\rho u A) = -A \frac{\partial p}{\partial x}. \quad (1.7)$$

Since A does not depend on time, the $\frac{\partial}{\partial t}$ terms can be combined to give $\frac{\partial}{\partial t}(\rho u A)$. The $\frac{\partial}{\partial x}$ terms can be combined to give $\frac{\partial}{\partial x}(\rho u^2 A)$. Use the chain rule on $\frac{\partial}{\partial x}(pA)$ and rearrange terms to give

$$-A \frac{\partial p}{\partial x} = -\frac{\partial}{\partial x}(pA) + p \frac{\partial A}{\partial x}.$$

Substituting these expressions into equation 1.7 gives

$$\frac{\partial}{\partial t}(\rho u A) + \frac{\partial}{\partial x}(\rho u^2 A) = -\frac{\partial}{\partial x}(pA) + p \frac{\partial A}{\partial x}.$$

This equation can be rearranged to give

$$\frac{\partial}{\partial t}(\rho u) + \frac{1}{A} \frac{\partial}{\partial x}(\rho u^2 A + pA) = \frac{p}{A} \frac{\partial A}{\partial x}. \quad (1.8)$$

This equation is the one-dimensional momentum equation.

The last equation comes from the law of conservation of energy. The First Law of Thermodynamics for a fluid particle can be expressed as [9]

$$\begin{aligned} & \text{rate of increase of total energy, } e, \text{ of the fluid particle} \\ &= \text{rate at which work is done on the particle by the body forces} \\ &+ \text{rate at which work is done on the particle, at its surface, by the surface forces.} \end{aligned}$$

Under the assumptions of this problem, the work done by the body forces is negligible.

Therefore, the equation is reduced to

$$\begin{aligned} & \text{rate of increase of total energy of the fluid particle} \\ &= \text{rate at which work is done on the particle, at its surface, by the surface forces.} \end{aligned}$$

The rate at which the energy is changing inside the particle has a nonstationary part and a convective part [12]. The nonstationary part is given by the change in energy with respect to time while the convective part is given by the change in energy times the velocity of the fluid. The rate of work done per unit volume is due to the pressure and surface forces which is given by the change in the pressure times the velocity. Thus, the energy equation is given by [1]

$$\frac{\partial e}{\partial t} + \frac{\partial}{\partial x}(eu) = -\frac{\partial}{\partial x}(pu)$$

or

$$\frac{\partial e}{\partial t} + \frac{\partial}{\partial x}(eu) + \frac{\partial}{\partial x}(pu) = 0, \quad (1.9)$$

where e is the total energy of the system.

Figure 1.3 - Volume of the Fluid Particle

Integration of equation 1.9 over the volume, V , of the particle (figure 1.3) gives

$$\int_V \left[\frac{\partial e}{\partial t} + \frac{\partial}{\partial x}(eu) + \frac{\partial}{\partial x}(pu) \right] dV = 0.$$

Using the divergence theorem, the integral reduces to

$$\bar{A} \Delta x \frac{\partial e}{\partial t} + \int_{\partial V} [(e+p)u] \cdot dS = 0,$$

where $\bar{A} = \frac{A_1 + A_2}{2}$. Integrating the second terms leads to

$$\bar{A} \Delta x \frac{\partial e}{\partial t} + (e+p)uA \Big|_x^{x+\Delta x} = 0.$$

Dividing through the equation by $\bar{A} \Delta x$ and taking the limit as $\Delta x \rightarrow 0$ results in

$$\frac{\partial e}{\partial t} + \frac{1}{A} \frac{\partial}{\partial x} (A(e+p)u) = 0. \quad (1.10)$$

Equations 1.3, 1.8 and 1.10 together model the air flow through a converging-diverging nozzle. To solve these equations, it is useful to combine them into a single vector equation,

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \frac{\partial}{\partial x} (A \mathbf{F}) = \mathbf{R}, \quad (1.11)$$

where $\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ e \end{bmatrix}$, $\mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(e + p) \end{bmatrix}$ and $\mathbf{R} = \begin{bmatrix} 0 \\ \frac{p}{A} \frac{\partial A}{\partial x} \\ 0 \end{bmatrix}$.

Equation 1.11 contains three equations in four unknowns. Therefore, one more equation is needed to relate these unknowns. The thermodynamic state of the system is described by two state variables, namely e and ρ . The equation of state is $p = p(e, \rho)$ [1].

One such equation is the perfect gas equation of state,

$$p = \rho RT, \quad (1.12)$$

where R is the gas constant and T is the temperature of the gas. The following relationships exist for a perfect gas:

$$c_v = \frac{R}{\gamma - 1} \text{ or } R = c_v(\gamma - 1)$$

and

$$e_{\text{int}} = c_v T \text{ or } T = \frac{e_{\text{int}}}{c_v},$$

where c_v is the specific heat at constant volume, γ is the ratio of specific heats (i.e. specific heat at constant pressure / c_v) and e_{int} is the internal energy per unit mass. These relationships can be substituted into equation 1.12 to give

$$p = (\gamma - 1)\rho e_{\text{int}} \quad (1.13)$$

The internal energy and the total energy of the system are related by

$$e = \rho \text{ (internal energy per unit mass)} \\ + \text{kinetic energy per unit mass} + \text{potential energy per unit mass} \quad (1.14)$$

The potential energy is negligible since gravity has been assumed to be negligible.

Therefore, equation 1.14 becomes

$$e = \rho \left(e_{\text{int}} + \frac{u^2}{2} \right)$$

or

$$e_{\text{int}} = \frac{e}{\rho} - \frac{u^2}{2}.$$

This relation can be substituted into equation 1.13 to give

$$p = (\gamma - 1) \left(e - \frac{\rho u^2}{2} \right), \quad (1.15)$$

Thus, equation 1.15 gives the necessary fourth equation to solve the system in equation 1.11. These equations model airflow through a converging-diverging nozzle.

CHAPTER 2

FLUX SPLITTING METHOD

The Euler equations are solved by the splitting of the $\mathbf{F}$ term in the equation:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \frac{\partial}{\partial x} (A \mathbf{F}) = \mathbf{R}. \quad (2.1)$$

The method used involves the splitting of the positive and negative eigenvalues of the system. The use of this method produces better results when trying to analyze a shock-capturing situation. Also, this method enables two different finite-differencing schemes to be employed when discretizing the $\frac{\partial}{\partial x}$ term. As will be seen, one portion of the $\frac{\partial}{\partial x}$ term can be discretized by using forward differences and one portion can be discretized by using backward differences.

For the nozzle application, the product and chain rule of differentiation can be used on equation 2.1 to rewrite it in the form [18]

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \left(A \frac{\partial \mathbf{F}}{\partial x} + \mathbf{F} \frac{\partial A}{\partial x} \right) = \mathbf{R}$$

or

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \left(A \frac{\partial \mathbf{F}}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial x} + \mathbf{F} \frac{\partial A}{\partial x} \right) = \mathbf{R}. \quad (2.2)$$

It can be shown by explicitly calculating the partial derivative that

$$\mathbf{F} = \frac{\partial \mathbf{F}}{\partial \mathbf{Q}} \mathbf{Q} = \mathbf{A} \mathbf{Q},$$

where $\mathbf{A} \equiv \frac{\partial \mathbf{F}}{\partial \mathbf{Q}} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}u^2(\gamma-3) & u(3-\gamma) & (\gamma-1) \\ (\gamma-1)u^3 - \frac{\gamma e u}{\rho} & \frac{e\gamma}{\rho} - \frac{3}{2}(\gamma-1)u^2 & \gamma u \end{bmatrix}$. (Note: $\mathbf{A}$ is not related to

A). Substituting this fact into equation 2.2 gives

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \left(A \mathbf{A} \frac{\partial \mathbf{Q}}{\partial x} + \mathbf{A} \mathbf{Q} \frac{\partial A}{\partial x} \right) = \mathbf{R}$$

or

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \mathbf{A} \left(A \frac{\partial \mathbf{Q}}{\partial x} + \mathbf{Q} \frac{\partial A}{\partial x} \right) = \mathbf{R}$$

or

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \mathbf{A} \frac{\partial A \mathbf{Q}}{\partial x} = \mathbf{R}. \quad (2.3)$$

In order to start the splitting process, the equations need to be rewritten in terms of the

primitive variables $\tilde{\mathbf{Q}} = \begin{bmatrix} \rho \\ u \\ p \end{bmatrix}$. Equation 2.3 is thus:

$$\frac{\partial \mathbf{Q}}{\partial \tilde{\mathbf{Q}}} \frac{\partial \tilde{\mathbf{Q}}}{\partial t} + \frac{1}{A} \mathbf{A} \frac{\partial A \mathbf{Q}}{\partial \tilde{\mathbf{Q}}} \frac{\partial \tilde{\mathbf{Q}}}{\partial x} = \mathbf{R}$$

or

$$\frac{\partial \mathbf{Q}}{\partial \tilde{\mathbf{Q}}} \frac{\partial \tilde{\mathbf{Q}}}{\partial t} + \mathbf{A} \frac{\partial \mathbf{Q}}{\partial \tilde{\mathbf{Q}}} \frac{\partial \tilde{\mathbf{Q}}}{\partial x} = \mathbf{R}$$

or

$$\mathbf{S} \frac{\partial \tilde{\mathbf{Q}}}{\partial t} + \mathbf{A} \mathbf{S} \frac{\partial \tilde{\mathbf{Q}}}{\partial x} = \mathbf{R}, \quad (2.4)$$

where $\mathbf{S} \equiv \frac{\partial \mathbf{Q}}{\partial \tilde{\mathbf{Q}}} = \begin{bmatrix} 1 & 0 & 0 \\ u & \rho & 0 \\ \frac{1}{2}u^2 & \rho u & (\gamma-1)^{-1} \end{bmatrix}$. It can be shown that

$$\mathbf{S}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -u/\rho & 1/\rho & 0 \\ \frac{1}{2}u^2(\gamma-1) & -u(\gamma-1) & (\gamma-1) \end{bmatrix}.$$

Multiplying equation 2.4 on the left by $\mathbf{S}^{-1}$ gives

$$\frac{\partial \tilde{\mathbf{Q}}}{\partial t} + \mathbf{S}^{-1} \mathbf{A} \mathbf{S} \frac{\partial \tilde{\mathbf{Q}}}{\partial x} = \mathbf{S}^{-1} \mathbf{R}.$$

Let $\tilde{\mathbf{A}} \equiv \mathbf{S}^{-1} \mathbf{A} \mathbf{S} = \begin{bmatrix} u & \rho & 0 \\ 0 & u & 1/\rho \\ 0 & \gamma p & u \end{bmatrix}$ or $\mathbf{A} = \mathbf{S} \tilde{\mathbf{A}} \mathbf{S}^{-1}$. Substituting this definition of $\mathbf{A}$ into

equation 2.1, with $\mathbf{F} = \mathbf{A} \mathbf{Q}$, the modified equation becomes

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \frac{\partial A (\mathbf{S} \tilde{\mathbf{A}} \mathbf{S}^{-1}) \mathbf{Q}}{\partial x} = \mathbf{R}. \quad (2.5)$$

$\tilde{\mathbf{A}}$ has three distinct eigenvalues (u , $u + a$ and $u - a$), where $a = \sqrt{\frac{\gamma p}{\rho}}$, which

implies $\tilde{\mathbf{A}}$ is diagonalizable. At least two of the eigenvalues are positive since u and $a > 0$.

Therefore, $\tilde{\mathbf{A}} = \mathbf{T}_{\tilde{\mathbf{A}}} \mathbf{\Lambda}_{\tilde{\mathbf{A}}} \mathbf{T}_{\tilde{\mathbf{A}}}^{-1}$ where $\mathbf{\Lambda}_{\tilde{\mathbf{A}}}$ is a diagonal matrix containing the eigenvalues and $\mathbf{T}_{\tilde{\mathbf{A}}}$ contains the corresponding eigenvectors as its columns:

$$\Lambda_{\bar{A}} = \begin{bmatrix} u & 0 & 0 \\ 0 & u+a & 0 \\ 0 & 0 & u-a \end{bmatrix} \text{ and } T_{\bar{A}} = \begin{bmatrix} 1 & 1/a^2 & -1/a^2 \\ 0 & 1/\rho a & -1/\rho a \\ 0 & 1 & 1 \end{bmatrix}.$$

It can be shown that $T_{\bar{A}}^{-1} = \begin{bmatrix} 1 & 0 & -1/a^2 \\ 0 & \rho a/2 & 1/2 \\ 0 & -\rho a/2 & 1/2 \end{bmatrix}$. Therefore, $A = S T_{\bar{A}} \Lambda_{\bar{A}} T_{\bar{A}}^{-1} S^{-1}$. Now, the

$\Lambda_{\bar{A}}$ matrix can be split into two matrices. One matrix, $\Lambda_{\bar{A}}^+$, contains the positive eigenvalues and the other, $\Lambda_{\bar{A}}^-$, contains the negative eigenvalues so that $\Lambda_{\bar{A}} = \Lambda_{\bar{A}}^+ + \Lambda_{\bar{A}}^-$.

Note that $\Lambda_{\bar{A}}^- = 0$ when there are no negative eigenvalues. Hence,

$$A = S T_{\bar{A}} (\Lambda_{\bar{A}}^+ + \Lambda_{\bar{A}}^-) T_{\bar{A}}^{-1} S^{-1} = S T_{\bar{A}} \Lambda_{\bar{A}}^+ T_{\bar{A}}^{-1} S^{-1} + S T_{\bar{A}} \Lambda_{\bar{A}}^- T_{\bar{A}}^{-1} S^{-1}.$$

For notational purposes, let $\Gamma^+ \equiv S T_{\bar{A}} \Lambda_{\bar{A}}^+ T_{\bar{A}}^{-1} S^{-1}$ and $\Gamma^- \equiv S T_{\bar{A}} \Lambda_{\bar{A}}^- T_{\bar{A}}^{-1} S^{-1}$ so that

$A = \Gamma^+ + \Gamma^-$. With $A = \tilde{S} \tilde{A} S^{-1}$, equation 2.5 becomes:

$$\frac{\partial Q}{\partial t} + \frac{1}{A} \left(\frac{\partial A (\Gamma^+ + \Gamma^-) Q}{\partial x} \right) = R$$

or

$$\frac{\partial Q}{\partial t} + \frac{1}{A} \left(\frac{\partial A \Gamma^+ Q}{\partial x} + \frac{\partial A \Gamma^- Q}{\partial x} \right) = R. \quad (2.6)$$

For Γ^- to make a contribution to the numerical scheme, the local speed must be subsonic (i.e. $u < a$). Thus, Γ^- contains the negative eigenvalue $u-a$ and the positive eigenvalues u and $u+a$ are stored in Γ^+ . Otherwise, when the local speed is supersonic ($u > a$), the Γ^+ term will contain all three of the positive eigenvalues.

Equation 2.6 is now split in terms of its eigenvalues. Hence, this equation can be discretized and solved numerically using an explicit method. Although an implicit method can be utilized, an explicit first-order method will be employed. When solving equation 2.6, each of the Γ terms incorporate a 3×3 matrix containing the eigenvalues.

Since the Γ^+ contains all of the eigenvalues when the local speed is supersonic, a backward differencing ("upwind") scheme is used to discretize the $\frac{\partial A \Gamma^+ Q}{\partial x}$ term. The conditions from upstream contribute more to the result since the flow is moving at a supersonic speed. Therefore, extra data points obtained upstream in a supersonic flow will lead to more accurate results. At subsonic speeds, the upstream points do not contribute more information than the downstream points. Since Γ^- contains the subsonic information, the usual forward difference is used for the $\frac{\partial A \Gamma^- Q}{\partial x}$ term.

Small oscillations occur when a sonic line is crossed because the flux splitting depends on the eigenvalues [1]. At these points, not only does the eigenvalue $u-a$ change sign but it is momentarily equal to 0. Hence, the matrix $\Lambda_{\tilde{A}}$ becomes singular for an instant. This singularity leads to the Γ^+ and Γ^- terms having a discontinuity in the first derivative with respect to x . To overcome this problem, a smoothing factor can be incorporated into the eigenvalues. With the smoothing, the eigenvalue $u-a$ will not become zero. The adjustment to the eigenvalues allows for a smoother transition when $u-a$ changes sign. Therefore, redefine the eigenvalues as

$$\lambda^{\pm} = \frac{\lambda \pm \sqrt{\lambda^2 + \varepsilon^2}}{2}$$

where ε is the smoothing factor. The value for ε^2 is chosen to be 0.001. By comparing the residual (defined in the next chapter) and ε^2 , $\varepsilon^2 = 0.001$ produced the largest smoothing factor without changing the residual.

CHAPTER 3

THE ALGORITHM

An explicit algorithm will be used to solve equation 2.5. Many people have solved this equation using an implicit method [1, 2, 6, 18]. Although explicit methods tend to converge faster than implicit methods, explicit methods have a disadvantage. These tend to become unstable if the appropriate conditions are not observed whereas implicit methods tend to be unconditionally stable. Once equation 2.5 has been discretized, the question of consistency, stability and convergence of the algorithm will be addressed. As will be seen, this method is convergent and appropriate initial conditions and boundary conditions will need to be defined in order to solve equation 2.5. Once all conditions are known, the iterative process will be explained.

As discussed in the last chapter, equation 2.6 can be discretized by using a forward difference for the $\frac{\partial Q}{\partial t}$ and $\frac{\partial A\Gamma^+Q}{\partial x}$ terms and a backward difference for the $\frac{\partial A\Gamma^-Q}{\partial x}$ term.

The discretized equation becomes

$$(Q_t)_i + \frac{\Delta t}{A\Delta x} \left\{ \left[(A\Gamma^+Q)_i^n - (A\Gamma^+Q)_{i-1}^n \right] + \left[(A\Gamma^-Q)_{i+1}^n - (A\Gamma^-Q)_i^n \right] \right\} = \Delta t R_i^n, \quad (3.1)$$

where the subscript i refers to the grid location in the x -direction and n is the current time step. The algorithm is first-order accurate in both space and time. With

$$Q_t \equiv \Delta Q_i = Q_i^{n+1} - Q_i^n,$$

equation 3.1 becomes

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{A\Delta x} \left\{ \left[(A\Gamma^+Q)_i^n - (A\Gamma^+Q)_{i-1}^n \right] + \left[(A\Gamma^-Q)_{i+1}^n - (A\Gamma^-Q)_i^n \right] \right\} + \Delta t R_i^n \quad (3.2)$$

This equation gives the solution for node i at the next time level. In order to use this equation, initial and boundary data must be provided. Once this is known, equation 3.2 can be implemented. An outline of the implementation is given in figure 3.2 at the end of this chapter where it will be discussed further.

One question arises from the use of equation 3.2: Will the use of this method converge to the theoretical solution? Lax's equivalence theorem states [17] :

Given a properly posed linear initial-value problem and a linear finite-difference approximation to it that satisfies the consistency condition, stability is the necessary and sufficient condition for convergence.

Therefore, if a method is consistent and stable then the numeric solution will converge to the theoretical solution. For a system to be consistent, the truncation error of the method must go to zero as the mesh size goes to zero. For a system to be stable, the calculated solution must remain bounded as the mesh size goes to zero. However, this theorem is valid for linear systems and the system in equation 2.6 is a non-linear system. Therefore, this theorem cannot be used to show convergence for the system in equation 2.6. However, if equation 3.2 is shown to be consistent and stable under some assumptions, then this would imply the numerical solution to a linearized version of equation 3.2 is convergent. This would provide some indication that the numerical solution to the non-linear system is convergent, at least in the region where the linear approximation is a good approximation. As a supplement to this theoretical analysis, steady-state solutions for different choices of Δt and Δx will be evaluated. If these solutions are equivalent, then this evaluation will provide more evidence that the calculated steady-state solution converges to the theoretical solution.

To show consistency, the truncation error of the difference approximation must go to zero as $\Delta x, \Delta t \rightarrow 0$. Let $T_i^n(\mathbf{Q})$ represent the finite-difference equation approximating

equation 2.6, where $\mathbf{Q}(x, t)$ is the solution. $T_i^n(\mathbf{Q})$ is also called the truncation error given by

$$T_i^n(\mathbf{Q}) = \frac{\mathbf{Q}_i^{n+1} - \mathbf{Q}_i^n}{k} + \frac{1}{A} \left[\frac{(A\Gamma^+\mathbf{Q})_i^n - (A\Gamma^+\mathbf{Q})_{i-1}^n}{h} + \frac{(A\Gamma^-\mathbf{Q})_{i+1}^n - (A\Gamma^-\mathbf{Q})_i^n}{h} \right] - \mathbf{R}_i^n,$$

where $k = \Delta t$ and $h = \Delta x$. Using a Taylor series expansion about the grid point (i, n) and rearranging terms gives

$$\begin{aligned} T_i^n(\mathbf{Q}) &= \left(\frac{\partial \mathbf{Q}}{\partial t} \right)_i^n + \frac{1}{A} \left[\left(\frac{\partial A\Gamma^+\mathbf{Q}}{\partial x} \right)_i^n + \left(\frac{\partial A\Gamma^-\mathbf{Q}}{\partial x} \right)_i^n \right] - \mathbf{R}_i^n. \\ &+ \frac{1}{2}k \left(\frac{\partial^2 \mathbf{Q}}{\partial t^2} \right)_i^n + \frac{1}{A} \left[-\frac{1}{2}h \left(\frac{\partial^2 A\Gamma^+\mathbf{Q}}{\partial x^2} \right)_i^n + \frac{1}{2}h \left(\frac{\partial^2 A\Gamma^-\mathbf{Q}}{\partial x^2} \right)_i^n \right] + O(k^2, h^2). \end{aligned}$$

By using $\left(\frac{\partial \mathbf{Q}}{\partial t} \right)_i^n + \frac{1}{A} \left[\left(\frac{\partial A\Gamma^+\mathbf{Q}}{\partial x} \right)_i^n + \left(\frac{\partial A\Gamma^-\mathbf{Q}}{\partial x} \right)_i^n \right] - \mathbf{R}_i^n = \mathbf{0}$, the principle part of the truncation error becomes

$$T_i^n(\mathbf{Q}) = \frac{1}{2}k \left(\frac{\partial^2 \mathbf{Q}}{\partial t^2} \right)_i^n + \frac{1}{A} \left[-\frac{1}{2}h \left(\frac{\partial^2 A\Gamma^+\mathbf{Q}}{\partial x^2} \right)_i^n + \frac{1}{2}h \left(\frac{\partial^2 A\Gamma^-\mathbf{Q}}{\partial x^2} \right)_i^n \right].$$

As $k \rightarrow 0$ and $h \rightarrow 0$, the truncation error $T_i^n(\mathbf{Q}) \rightarrow 0$. Therefore, the explicit method used in equation 3.2 is consistent.

In order for equation 3.2 to be convergent, it must first be stable. For non-linear equations, stability analysis is not straight-forward. As Beam and Warming state [18]:

While the von Neumann (spatially periodic) stability analysis for a system of difference equations is sometimes tractable, the stability analysis with nonperiodic boundary conditions can be prohibitive and usually one must resort to simplified model equations and/of a heuristic stability analysis.

Since the boundary conditions that will be used are not periodic, a crude analysis can be made under an assumption. This assumption will simplify the non-linear system of equation 2.6 to a linear system. If the matrix $\mathbf{A}$ in equation 2.3 is assumed to be locally constant, then the system can be linearized. Now, a von Neumann analysis can be done on this linearized system. The analysis is done on a certain volume of the nozzle assuming an infinitesimally small thickness dx and constant cross-sectional area A^* (see figure 3.1).

Figure 3.1 - Volume for Stability

Using this volume, equation 2.3 can be linearized as

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{A} \frac{\partial \mathbf{Q}}{\partial x} = \mathbf{0},$$

where $\mathbf{A}$ is assumed to be locally constant. Since this system is hyperbolic, $\mathbf{A}$ can be diagonalized and the vector equation can be uncoupled into three scalar equations [7]. Each scalar equation has the form:

$$\frac{\partial U}{\partial t} + \sigma \frac{\partial U}{\partial x} = 0, \quad (3.3)$$

where σ is some constant associated with the eigenvalues of $\tilde{\mathbf{A}}$. To approximate the Flux Splitting Method, two von Neumann stability analyses must be performed on equation 3.3.

One analysis will use a forward difference for the $\frac{\partial}{\partial x}$ term (if σ correspond to a subsonic case) and the other will use a backward difference (if σ corresponds to a supersonic case). If both analyses prove to be stable, then conclude the method is stable.

Discretizing equation 3.3 gives

$$U_i^{n+1} - U_i^n + \sigma r (U_{i+1}^n - U_i^n) = 0, \quad (3.4)$$

where $r \equiv \frac{\Delta t}{\Delta x}$. Substituting $U_i^n = e^{i\beta h} \xi^n$, as defined for von Neumann analysis [17], gives

$$e^{i\beta h} \xi^{n+1} - e^{i\beta h} \xi^n + \sigma r (e^{i\beta(j+1)h} \xi^n - e^{i\beta h} \xi^n) = 0.$$

Dividing through by $e^{i\beta h} \xi^n$ and solving for ξ results in

$$\xi = 1 + \sigma r - \sigma r e^{i\beta h}. \quad (3.5)$$

Since ξ is the amplification factor for stability, a system will be stable if $|\xi| \leq 1$. From equation 3.5,

$$\begin{aligned} \xi &= (1 + \sigma r - \sigma r \cos \beta h) - i \sin \beta h. \\ |\xi|^2 &= (1 + \sigma r - \sigma r \cos \beta h)^2 + \sin^2 \beta h \\ &= 1 + 2\sigma r (1 - \cos \beta h) (1 + \sigma r). \end{aligned}$$

For stability, $|\xi| \leq 1$ or $|\xi|^2 \leq 1$ must be satisfied. Therefore,

$$2\sigma r (1 - \cos \beta h) (1 + \sigma r) \leq 0$$

is needed for stability. This inequality is true when

$$2\sigma r < 0 \text{ and } (1 + \sigma r) > 0,$$

since $(1 - \cos \beta h) \geq 0$ is always true. Since the eigenvalue is negative for the Γ^- term in equation 2.6, the corresponding σ term in equation 3.4 is also negative. The inequality $2\sigma r < 0$ is always true and $(1 + \sigma r) > 0$ is true when $0 < -\sigma r < 1$. Therefore, equation 3.4 is stable for $0 < -\sigma r < 1$.

Likewise, a similar analysis can be done by using a backward difference for $\frac{\partial}{\partial x}$ in equation 3.3. The analogous equation of 3.5 is

$$\xi = 1 - \sigma r (1 - e^{-i\beta h}).$$

Determining $|\xi|^2$ gives

$$|\xi|^2 = 1 + 2\sigma r(-1 + \sigma r)(1 - \cos\beta h).$$

Again, for stability, $|\xi|^2 \leq 1$ must be satisfied. This property occurs when $2\sigma r > 0$ and $(-1 + \sigma r) < 0$. From equation 2.6, the eigenvalues in the Γ^+ term are positive leading to $\sigma > 0$. Thus, the backward difference case is stable when $0 < \sigma r < 1$.

Since both forward and backward differencing for the $\frac{\partial}{\partial x}$ term lead to conditionally stable methods, the Flux Splitting Method will be stable for the linearized system. Equation 3.2 is both consistent and stable under the assumption stated earlier. In Chapter 4, it is shown that different choices of Δt and Δx lead to the same steady-state solution. Having established consistency, stability and equivalence of steady-state solutions, these facts suggest equation 3.2 will converge to the theoretical solution.

To implement equation 3.2, the initial and boundary conditions need to be defined. The initial conditions need to be specified in order to have the starting values for Q_i^0 , where i ranges over the grid points in the x-direction. For these conditions, three entrance boundary conditions into the nozzle are specified. They are p , ρ and M (the Mach number $\equiv \frac{u}{a}$). These values are the controllable factors in the physical problem. For instance, a wind tunnel can simulate air flow at a designated Mach number, pressure and density. Given the entrance boundary information, the theory of steady compressible flow will give an adequate approximation for the initial conditions along the nozzle.

The calculations for the initial conditions are made as follows. Steady adiabatic compressible flow (meaning the flow does not fluctuate with time) in one-dimension can be described by Bernouli's equation [12]

$$u \, du + \frac{dp}{\rho} = 0. \quad (3.6)$$

Using the definition of M and $\alpha^2 \equiv \left(\frac{\partial p}{\partial \rho} \right)$, equation 3.6 can be rearranged as

$$\frac{dp}{\rho} = -M^2 \frac{du}{u}. \quad (3.7)$$

The continuity equation for flow through a tube is

$$\rho u A = \text{const1}, \quad (3.8)$$

where *const1* is some constant. Taking the total derivative of this equation, it becomes

$$\frac{d\rho}{\rho} + \frac{du}{u} + \frac{dA}{A} = 0. \quad (3.9)$$

Substituting equation 3.7 into equation 3.9 gives the area velocity relation

$$\frac{du}{u} = -\frac{dA/A}{(1-M^2)}.$$

One further substitution of equation 3.7 results in the change in density equation

$$d\rho = \rho M^2 \frac{dA/A}{(1-M^2)}.$$

The right-hand side of this equation can be discretized at node i using a backward difference for dA . Setting this discretized equation equal to $\Delta\rho_i$ gives

$$\Delta\rho_i = \rho_i M_i^2 \frac{(A_i - A_{i-1})/A_i}{1 - M_i^2}. \quad (3.10)$$

Now, ρ_{i+1} can be determined from

$$\rho_{i+1} = \rho_i + \Delta\rho_i. \quad (3.11)$$

The u component is determined from a rearrangement of equation 3.8

$$u = \frac{\text{const1}}{\rho A}.$$

The constant can be calculated using the entrance conditions on u and ρ

$$const1 = u_0 \rho_0 A_0 \quad (3.12)$$

and

$$u_i = \frac{const1}{\rho_i A_i}. \quad (3.13)$$

The last two components to solve for are e and p . By using the equation of state (equation 1.15), e_i can be solved for if p_i is known. The component p can be determined as follows. From the conservation of energy law, the energy equation for steady flow is [12]

$$p_1 v_1 - p_2 v_2 = \left(e_2 + \frac{1}{2} u_2^2 \right) - \left(e_1 + \frac{1}{2} u_1^2 \right),$$

where the subscript 1 denotes the incoming side of figure 1.2 and the subscript 2 refers to the outgoing side. Introducing an enthalpy term, $h = e + pv$, the above energy equation can be rewritten as

$$c_p T + \frac{1}{2} u^2 = const. = c_p T_0,$$

assuming the gas is in equilibrium and is thermally and calorically perfect. Using the expression $a^2 = \gamma RT$ for the speed of sound, the energy equation for a perfect gas is

$$\frac{u^2}{2} + \frac{a^2}{\gamma - 1} = \frac{a_0^2}{\gamma - 1}, \quad (3.14)$$

where a_0 is the entrance speed of sound. Since a_0 depends only on the entrance pressure and density, the right-hand side of equation 3.14 remains constant throughout the length

of the nozzle. Using $a = \sqrt{\frac{\gamma p}{\rho}}$ and the equation of state, equation 1.15, equation 3.14

simplifies to

$$\frac{e+p}{\rho} = \text{const2}.$$

The constant is determined by the entrance values of e , p and ρ giving

$$\text{const2} = \frac{e_0 + p_0}{\rho_0}. \quad (3.15)$$

Substituting equation 3.15 into the equation of state, equation 1.15, the pressure can be determined by

$$p = \frac{\gamma - 1}{\gamma} \left(\rho \times \text{const2} - \frac{\rho u^2}{2} \right),$$

where the term const2 is given in equation 3.15. The discretized equation is

$$p_i = \frac{\gamma - 1}{\gamma} \left(\rho_i \times \text{const2} - \frac{\rho_i u_i^2}{2} \right) \quad (3.16)$$

Once the initial values of p are known, the values for e can be determined by using equation 1.15.

The initial conditions are determined by first calculating the appropriate constants from equations 3.12 and 3.15. Then these constants are used in equations 3.11, 3.13, 3.16 and 1.15 to determine the values for ρ_i , u_i , p_i and e_i , where $2 \leq i \leq \text{node \# of the exit boundary}$. These values become the initial guess for the iterative process. The iterative process will be explained later in this chapter.

Now that the initial conditions are known, the right boundary condition must be known in order to implement equation 3.2. At the exit boundary, an accurate solution reflecting the actual flow must be calculated. Hence, assume the flow is unsteady, i.e. it fluctuates with time. Characteristic theory will be used to determine the exit boundary conditions. In general, this theory leads to more accurate results since the exit boundary is a free boundary. To calculate these conditions, the components of $\mathbf{Q}$ at each node

(excluding the boundary node) for the next time level must be known. The values of $\mathbf{Q}$ at the next time level are computed by using the iteration process. Now, the boundary values can be determined.

To calculate the boundary conditions, the compatibility equations must be determined [6]. It can be shown that

$$\delta\tilde{\mathbf{Q}} = \mathbf{S}^{-1}\delta\mathbf{Q}.$$

The characteristic variable $\mathbf{C}$ is defined from

$$\begin{aligned}\mathbf{A}\delta\mathbf{Q} &= \tilde{\mathbf{S}}\mathbf{A}\mathbf{S}^{-1}\delta\mathbf{Q} \\ &= \mathbf{S}\mathbf{T}_{\tilde{\mathbf{A}}}\mathbf{\Lambda}_{\tilde{\mathbf{A}}}\mathbf{T}_{\tilde{\mathbf{A}}}^{-1}\mathbf{S}^{-1}\delta\mathbf{Q} \\ &= \mathbf{S}\mathbf{T}_{\tilde{\mathbf{A}}}\mathbf{\Lambda}_{\tilde{\mathbf{A}}}\delta\mathbf{C},\end{aligned}$$

$$\text{where } \delta\mathbf{C} \equiv \mathbf{T}_{\tilde{\mathbf{A}}}^{-1}\delta\tilde{\mathbf{Q}} = \begin{bmatrix} \delta p - \frac{1}{a^2}\delta p \\ \frac{\rho a}{2} + \frac{1}{2}\delta p \\ -\frac{\rho a}{2} + \frac{1}{2}\delta p \end{bmatrix}.$$

The Euler equations (for flow through a tube with constant cross-section) can be rewritten in terms of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{Q}}$

$$\frac{\partial\tilde{\mathbf{Q}}}{\partial t} + \tilde{\mathbf{A}} \frac{\partial\tilde{\mathbf{Q}}}{\partial x} = 0. \quad (3.17)$$

By multiplying both sides by $\mathbf{T}_{\tilde{\mathbf{A}}}^{-1}$ and inserting the identity matrix $\mathbf{T}_{\tilde{\mathbf{A}}}\mathbf{T}_{\tilde{\mathbf{A}}}^{-1}$, equation 3.17 becomes

$$\mathbf{T}_{\tilde{\mathbf{A}}}^{-1} \frac{\partial\tilde{\mathbf{Q}}}{\partial t} + \mathbf{T}_{\tilde{\mathbf{A}}}^{-1}\tilde{\mathbf{A}}\mathbf{T}_{\tilde{\mathbf{A}}}\mathbf{T}_{\tilde{\mathbf{A}}}^{-1} \frac{\partial\tilde{\mathbf{Q}}}{\partial x} = 0$$

or

$$\mathbf{T}_A^{-1} \frac{\partial \tilde{\mathbf{Q}}}{\partial t} + \Lambda_A \mathbf{T}_A^{-1} \frac{\partial \tilde{\mathbf{Q}}}{\partial x} = 0. \quad (3.18)$$

Using the definition of $\delta \mathbf{C}$, $\frac{\delta \mathbf{C}}{\delta t} = \mathbf{T}_A^{-1} \frac{\delta \mathbf{Q}}{\delta t} \rightarrow \frac{\partial \mathbf{C}}{\partial t} = \mathbf{T}_A^{-1} \frac{\partial \mathbf{Q}}{\partial t}$ as $\delta t \rightarrow 0$. Likewise,

$\frac{\delta \mathbf{C}}{\delta x} = \mathbf{T}_A^{-1} \frac{\delta \mathbf{Q}}{\delta x} \rightarrow \frac{\partial \mathbf{C}}{\partial x} = \mathbf{T}_A^{-1} \frac{\partial \mathbf{Q}}{\partial x}$. Therefore, equation 3.18 can be written as

$$\frac{\partial \mathbf{C}}{\partial t} + \Lambda_A \frac{\partial \mathbf{C}}{\partial x} = 0.$$

This equation is called the compatibility relation. In component form, these relations are

$$\begin{aligned} \left(\frac{\partial \rho}{\partial t} - \frac{1}{a^2} \frac{\partial p}{\partial t} \right) + u \left(\frac{\partial \rho}{\partial x} - \frac{1}{a^2} \frac{\partial p}{\partial x} \right) &= 0 \\ \left(\rho a \frac{\partial u}{\partial t} + \frac{\partial p}{\partial t} \right) + (u + a) \left(\rho a \frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} \right) &= 0 \\ \left(-\rho a \frac{\partial u}{\partial t} + \frac{\partial p}{\partial t} \right) + (u - a) \left(-\rho a \frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} \right) &= 0. \end{aligned}$$

From these compatibility relations, the exit boundary conditions can now be determined. Define R_1 , R_2 and R_3 as follows, where the subscript I refers to the node number at the exit boundary:

$$R_1 = \delta \rho_I - \frac{1}{a^2} \delta p_I = -\frac{\alpha_1}{1 + \alpha_1} \left((\rho_I^n - \rho_{I-1}^n) - \frac{1}{a^2} (p_I^n - p_{I-1}^n) \right); \quad \alpha_1 = \frac{u_I^n \Delta t}{\Delta x} \quad (3.19)$$

$$R_2 = \delta p_I + (\rho a)_I^n \delta u_I = -\frac{\alpha_2}{1 + \alpha_2} \left((p_I^n - p_{I-1}^n) + (\rho a)_I^n (u_I^n - u_{I-1}^n) \right); \quad \alpha_2 = \frac{(u + a)_I^n \Delta t}{\Delta x} \quad (3.20)$$

$$R_3 = \delta p_I - (\rho a)_I^n \delta u_I = -\frac{\alpha_3}{1 + \alpha_3} \left((p_I^n - p_{I-1}^n) - (\rho a)_I^n (u_I^n - u_{I-1}^n) \right); \quad \alpha_3 = \frac{(u - a)_I^n \Delta t}{\Delta x}. \quad (3.21)$$

The superscript n refers to the current time level. The components of $\mathbf{Q}$ at each node i and time level n are known. In particular, $\mathbf{Q}_I^n$ is known. By using equations 3.19 - 3.21, $\mathbf{Q}_I^{n+1}$ can be determined as follows.

Add equation 3.20 to 3.21 to get

$$R_2 + R_3 = 2\delta p_I.$$

Rearrange this equation to get

$$\delta p_I = \frac{R_2 + R_3}{2}$$

or

$$\delta p_I = \begin{cases} \frac{R_2 + R_3}{2} & \text{if } M_I^n = \left(\frac{u}{a}\right)_I^n > 1 \\ \frac{\partial p_I}{\partial t} \Delta t & \text{if } M_I^n < 1 \end{cases}, \quad (3.22)$$

where $\delta p_I = p_I^{n+1} - p_I^n$. With δp_I known, $\delta \rho_I$ and δu_I can be readily determined.

$$\delta \rho_I = R_1 + \left(\frac{\delta p}{a^2}\right)_I^n \quad (3.23)$$

$$\delta u_I = \frac{R_2 - \delta p}{(\rho a)_I^n}. \quad (3.24)$$

Using the notation $\delta z_I = z_I^{n+1} - z_I^n$, ρ_I^{n+1} and u_I^{n+1} can be calculated. The term e_I^{n+1} can be determined using the equation of state, equation 1.15. Thus, equations 3.22 - 3.24 determine p , ρ , u and e at the exit boundary.

Now that the initial and boundary conditions are known, the iterative process can begin. Figure 3.2 gives an outline of the iterative procedure. In an iterative process, a calculation gives a correction term for the guess at the old time level. This correction is added on to the solution at the old time level to give a more accurate guess at the new time level:

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n + \Delta \mathbf{Q}_i$$

This new guess becomes the old guess and the process of computing the correction term begins again. Thus, the new values will be iterated on to determine the subsequent values. The iteration steps continue until the correction term has met a given criteria (i.e. the error term is within the specifications). The result will be the steady-state solution of the system.

Figure 3.2 - Outline of Iterative Procedure

A more specific look at the iterative procedure is as follows. To start the iteration process, the entrance boundary conditions must be specified. These values are used to determine the initial conditions as given in equation 3.10, 3.11, 3.12, 3.14 and 3.15. These values become the old solution, Q_i^0 with $1 \leq i \leq I$. Now that the old solution is known, use equation 3.2 to determine $Q_i^1 = Q_i^0 + \Delta Q_i$ for $2 \leq i \leq I-1$. The term ΔQ_i is the correction term calculated by the iterative process. The calculations for Q_i^1 are done from left to right for each node 2 to $I-1$. The value for Q_I^1 uses the boundary conditions, equations 3.22 to 3.24. Once all Q_i^1 , $2 \leq i \leq I$, values have been calculated, then the terms Q_i^2 , $2 \leq i \leq I-1$, can be determined by calculating a new correction term and Q_I^2 can be determined from the boundary conditions. Therefore, Q can be calculated at subsequent time levels.

This iterative process terminates once a certain tolerance level (which is specified) has been met. After all values at a new time level are determined, the error is checked to see if it has met the tolerance requirement. Define the residual as

$$\text{error} \equiv \text{norm}(\rho^{n+1} - \rho^n),$$

where the norm used is the infinity-norm. Once the residual has met the specified tolerance, the iterative procedure terminates. The result of the algorithm is an approximation to the steady-state solution to the Euler equations.

CHAPTER 4

SET-UP AND SOLUTION

In this chapter, the results of the Flux Splitting method will be given. The nozzle will be defined in terms of a given cross-sectional area. The entrance conditions will be given and a steady-state solution to equation 2.6 will be obtained. By redefining the time and space steps, it will be concluded that the solution obtained from using the algorithm is the actual steady-state solution. The Flux Splitting algorithm is programmed using MATLAB [19]. MATLAB is installed on an 486-DX PC compatible. The machine epsilon for this computer is 7.0×10^{-17} . The underflow limit is 5.0×10^{-324} and the overflow limit is 1.0×10^{308} .

Figure 4.1 - View of the Nozzle

For the convergent-divergent nozzle, the entrance Mach number of the fluid is 2.3 (to simulate airflow through a jet engine) with a density of 0.2596 and at a pressure of 0.1081. (Note that the entrance pressure and density are normalized values). The cross-sectional area of the nozzle is simulated by $A(x) = 1 + 4(x - 0.5)^2$ where $0 \leq x \leq 1$. The throat of the nozzle occurs at $x = 0.5$, where the area is equal to 1.0 (see figure 4.1). In wind tunnel applications, a plenum chamber is attached to the end of the nozzle. The

pressure in this chamber determines the location of the shock in the nozzle [4]. Usually, a shock is placed after the throat of the nozzle. For this application, a shock-wave is placed at $x = 0.8$.

The computational grid has a uniform distribution of points in the x -direction and a non-uniform distribution of points in the time domain. Forty-one points are distributed evenly along the x -direction giving $\Delta x = 1/40$. (The reason for choosing this space step is in the last paragraph of this chapter). The time step needs to be determined from the stability analysis in the previous chapter. From the last chapter, the flux splitting method for a linearized version of equation 1.11 was shown to be stable given that a certain criteria was met ($0 < |\sigma| < 1$). The time step needs to satisfy the Courant-Friedrichs-Lewy (CFL) condition, $0 < \sigma \frac{\Delta t}{\Delta x} \leq 1$, as stated in Smith's book [17]. (Recall from chapter 3, σ is a constant related to the eigenvalues of $\tilde{A}$ in equation 2.5). This condition depends on the domain of dependence. The CFL condition is satisfied when $\sigma \frac{\Delta t}{\Delta x} \equiv \lambda_{\max} \frac{\Delta t}{\Delta x} = 0.9$, where $\lambda_{\max}$ is the maximum eigenvalue of $\tilde{A}$ as defined in equation 2.5. Setting the time and space ratio equal to 0.9 enables Δt to be as large as possible and still satisfy the CFL condition. Thus, the time step is determined by

$$\Delta t = \frac{0.9 \Delta x}{\lambda_{\max}}, \quad (4.1)$$

which will vary with each iteration.

Using this set-up and calculating the initial conditions as described in Chapter 3, the initial distribution is given in figure 4.2. The iterative process, equations 3.2 and 3.22 - 3.24, begins with this information. The iterations continue until the tolerance has been met. For a steady-state solution, the residual needs to be very small to ensure the solution is not changing from one time step to another. Thus, setting the tolerance for the residual

at 1.0×10^{-15} guarantees a steady-state solution. All calculated solutions will be held to this tolerance. Figure 4.3 shows the steady-state solution after reaching the tolerance. It took 1,519 iterations to reach this converged solution. The total time for convergence was 13.5644 seconds (assuming Δt is in seconds).

Figure 4.2 - Initial Distribution

In Chapter 3, the Flux Splitting method was shown to be consistent and stable. To show the method converged to the analytical solution, the calculated steady-state solution needed to be independent of the choice of Δt and Δx (as long as the CFL condition remained satisfied). First, the Δt term is altered. The steady-state solution with

$\Delta t = \frac{0.6\Delta x}{\lambda_{\max}}$ differs from the steady-state solution using equation 4.1 by a magnitude of

10^{-13} . When $\Delta t = \frac{1.2\Delta x}{\lambda_{\max}}$, the calculated steady-state solution differs from the steady-state solution (using equation 4.1) by a magnitude of 10^{-14} . However, oscillations start to occur in the residual when the residual reaches 10^{-14} . These oscillations occur because the CFL condition has not been satisfied. This Δt is above the threshold for stability. If the time step is increased even further to $\Delta t = \frac{1.4\Delta x}{\lambda_{\max}}$, the oscillations start to occur at 10^{-4} and the residual grows rather than decreases. Therefore, changing Δt does not affect the solution as long as the CFL condition has been met.

Figure 4.3 - Converged Solution

Secondly, the Δx term needs to be altered. Figure 4.4 shows the results of the Mach number for $\Delta x = \frac{1}{20}$, $\frac{1}{40}$, $\frac{1}{80}$ and $\frac{1}{160}$. As Δx becomes smaller, the shock wave becomes more pronounced. Mathematically, the shock is a discontinuity. The smaller step sizes in the x-direction lead to a more accurate approximation of the discontinuity. The solution changes as Δx is decreased. However, for any single choice of Δx , the solution obtained is accurate for the particular choice of Δx .

The calculated solutions from changing Δt are exact to within 10^{-13} given the CFL condition has been met. The calculated solutions are dependent on the choice for Δx . Therefore, the solution obtained using equation 3.2 is the steady-state solution for the particular choice of Δx .

Figure 4.4 - Converged Solution for Mach Number with Different Values for Δx

As seen in figure 4.4, the solution depends on the choice of Δx . The following describes what needs to be taken into consideration when selecting an appropriate value for Δx . When $\Delta x = \frac{1}{20}$, the shock wave in the solution has not completely formed. This choice of Δx does not best represent the actual air flow. When $\Delta x = \frac{1}{160}$, the shock wave is the most pronounced. However, the run time for this choice of Δx is long. It takes approximately five minutes to run one-hundred iterations. This amount of time is too long considering the residual needs to reach 10^{-15} . Therefore, a good compromise between a formed shock and run time is reached when $\Delta x = \frac{1}{40}$.

CHAPTER 5

ACCELERATION METHODS

In this chapter, three acceleration methods used to speed up the convergence of the solution will be discussed and implemented. The results of the accelerating the Flux Splitting method will be analyzed. The analysis will compare the acceleration rates and the extra work used in reaching the converged solution. The most efficient acceleration method will be chosen by using the information of the analysis.

Three different types of acceleration procedures are used. The first procedure is a standard method for linear systems called Relaxation (or SOR) [17]. Although equation 2.3 is non-linear, SOR is used to see if it will accelerate the convergence of the solution. This method is defined as

$$\mathbf{Q}_{\text{improved}} = \omega \mathbf{Q}_{\text{new}} + (1 - \omega) \mathbf{Q}_{\text{old}} \quad (5.1)$$

where $0 \leq \omega \leq 2$ is the relaxation factor. Over-relaxation occurs when $1 < \omega \leq 2$. In this case, more weight is given to the new solution. Under-relaxation occurs when $0 < \omega < 1$ and hence, the old solution is given more weight. The best results occurred when $\omega = 1.25$ was used. This improvement is performed at each iteration.

A comparison of this method with the non-accelerated solution is shown in figure 5.1. The solution using relaxation converged in 1,223 iterations. This procedure requires only 240 extra flops (floating point operations where one multiplication or division and two additions or subtractions are counted as a single flop) at each step. The extra work involved to reach convergence is an added 293,000 flops. This count is minimal when compared with the 27,831 flops required for only one complete iteration step. Although

this is a savings of approximately 300 iterations or 20%, the improvement is not consistent until after 350 iterations.

Figure 5.1 - Relaxation

In general, the SOR method may not be a viable acceleration method for all non-linear systems. However, it does produce favorable results for the particular non-linear system in equation 2.3. The SOR method is cost-effective in terms of acceleration rate versus extra work involved.

The second and third acceleration methods are explicit eigenvalue annihilations. These methods can be used for linear, as well as non-linear, systems. The error at each step is controlled by the largest eigenvalue of the matrix $\mathbf{B}$ in the iteration $x^{n+1} = \mathbf{B}x^n + c$. If x^* is the analytical solution and x^n is the computed solution after n iterations, then the

error is $e^n = x^* - x^n$ [6]. Assume $\mathbf{B}$ is diagonalizable with eigenvalues λ_i and corresponding eigenvectors $\bar{v}_i$. The error can be written as

$$e^n = \sum_{i=1}^m \alpha_i \lambda_i^n \mathbf{v}_i = \lambda_1^n \left(\alpha_1 \mathbf{v}_1 + \sum_{i=2}^m \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^n \mathbf{v}_i \right)$$

where $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_m|$. As the number of iterations increases, the $\left(\frac{\lambda_i}{\lambda_1} \right)^n$ term goes to

zero. Thus, the summation approaches zero giving

$$e^n \cong \lambda_1^n \alpha_1 \mathbf{v}_1, \quad (5.2)$$

as $n \rightarrow \infty$. Since α_1 and $\mathbf{v}_1$ remain constant, λ_1 controls the error in the iterative process. Thus, the computed solution converges when $|\lambda_1| < 1$. If this eigenvalue can be eliminated, then from equation 5.2, the error will be controlled by the next eigenvalue. Hence, the overall error term becomes smaller leading to a faster convergence rate.

The second acceleration method is based on a technique used by A. Cheer and M. Saleem [2]. In this method, the largest eigenvalue is eliminated by a linear combination of the two most recent solution gives the following method

$$\mathbf{Q} = \frac{\mathbf{Q}^{n+1} - \lambda_{\max} \mathbf{Q}^n}{1 - \lambda_{\max}}. \quad (5.3)$$

In order to use this procedure, the largest real eigenvalue of $\mathbf{B}$ must be known in order to avoid complex arithmetic. The matrix $\mathbf{B}$ is an 120×120 matrix consisting of nine 40×40 tri-diagonal blocks. Calculating the eigenvalues of $\mathbf{B}$ at each annihilation step can be quite cumbersome even with MATLAB's built in eigenvalue function. On a random 120×120 matrix, MATLAB uses almost six million flops to calculate the eigenvalues. Therefore, using the estimation

$$\lambda_{\max} = \frac{\|\mathbf{Q}_{n+1}\|}{\|\mathbf{Q}_n\|}, \quad (5.4)$$

which requires approximately 4500 flops, will reduce the amount of work done in calculating the eigenvalue by 99.92%. Once $\lambda_{\max}$ is known, the procedure requires only 300 extra flops. Thus, one annihilation step uses 4800 flops. Figure 5.2 shows the estimated dominant eigenvalue at each iteration step. Since these value are less than one, this annihilation procedure will converge according to equation 5.2.

Figure 5.2 - Dominant Eigenvalue Estimation

Figure 5.3 shows the results of this single eigenvalue annihilation procedure done at iterations 130, 190, 250, 350, 500, 600, 725, 850 and 950. The solution converged to the specified tolerance in 1373 iterations. Although this procedure has a savings of

approximately 150 iterations or 10%, the SOR method converged at a faster rate. But, annihilation for the nine steps required only 43,200 extra flops while the SOR method required 249,800 (or ~570%) more flops. Between these two methods, there is a trade off in convergence time and extra work. Although the SOR method converged at a faster rate, it also requires more work.

Figure 5.3 - Single Eigenvalue Annihilation

The third acceleration method is another explicit eigenvalue annihilation based on D. Jespersen and P. Buning's method [6]. The second method requires the largest eigenvalue to be real. But in many cases, the largest eigenvalue is complex. Hence, two eigenvalues, the dominant eigenvalue and its conjugate, have the largest magnitude. Therefore, two complex conjugate eigenvalues will be eliminated. This method is a linear combination of the three most recent solutions. Again, the error at each step is controlled

by the largest eigenvalue of the system $x^{n+1} = \mathbf{B}x^n + c$. At three consecutive steps, the errors are (assuming $\mathbf{Q}^*$ is the analytic steady-state solution)

$$\mathbf{Q}^* = \mathbf{Q}^n + \alpha \lambda^n \mathbf{v} + \beta \bar{\lambda}^n \bar{\mathbf{v}} + O(|\lambda_3|^n) \quad (5.5)$$

$$\mathbf{Q}^* = \mathbf{Q}^{n+1} + \alpha \lambda^{n+1} \mathbf{v} + \beta \bar{\lambda}^{n+1} \bar{\mathbf{v}} + O(|\lambda_3|^{n+1}) \quad (5.6)$$

$$\mathbf{Q}^* = \mathbf{Q}^{n+2} + \alpha \lambda^{n+2} \mathbf{v} + \beta \bar{\lambda}^{n+2} \bar{\mathbf{v}} + O(|\lambda_3|^{n+2}) \quad (5.7)$$

where λ and its conjugate, $\bar{\lambda}$, are the dominant eigenvalues of $\mathbf{B}$ with $\mathbf{v}$ and $\bar{\mathbf{v}}$ as the corresponding eigenvectors, respectively. The α -term can be eliminated by multiplying equation 5.5 by λ and subtracting equation 5.6 from it. Likewise, multiplying equation 5.6 by λ and subtracting equation 5.7 results in, respectively,

$$\lambda \mathbf{Q}^* - \mathbf{Q}^* = (\lambda \mathbf{Q}^n - \mathbf{Q}^{n+1}) + \beta \bar{\mathbf{v}} (\lambda \bar{\lambda}^n - \bar{\lambda}^{n+1}) + O(|\lambda_3|^n) \quad (5.8)$$

$$\lambda \mathbf{Q}^* - \mathbf{Q}^* = (\lambda \mathbf{Q}^{n+1} - \mathbf{Q}^{n+2}) + \beta \bar{\mathbf{v}} (\lambda \bar{\lambda}^{n+1} - \bar{\lambda}^{n+2}) + O(|\lambda_3|^{n+1}). \quad (5.9)$$

The β -term can be eliminated from these equations by multiplying equation 5.8 by $\bar{\lambda}$ and subtracting equation 5.9. The final equation is

$$\lambda \bar{\lambda} \mathbf{Q}^* - \bar{\lambda} \mathbf{Q}^* - \lambda \mathbf{Q}^* + \mathbf{Q}^* = \lambda \bar{\lambda} \mathbf{Q}^n - (\lambda - \bar{\lambda}) \mathbf{Q}^{n+1} + \mathbf{Q}^{n+2} + O(|\lambda_3|^n)$$

This equation can be rearranged to give

$$\mathbf{Q}^{\text{imp}} = \frac{\mathbf{Q}^{n+2} - 2a\mathbf{Q}^{n+1} + (a^2 + b^2)\mathbf{Q}^n}{1 - 2a + a^2 + b^2}, \quad (5.10)$$

where $\lambda_{\text{max}} = a + ib$ and $\mathbf{Q}^{\text{imp}} = \mathbf{Q}^* - O(|\lambda_3|^n)$. Therefore, complex eigenvalues and their conjugates can be annihilated by using real arithmetic.

In order to use this procedure, the largest eigenvalue of $\mathbf{B}$ must be known. As in the single value annihilation procedure, determining the dominant pair of eigenvalues can be extensive. Another eigenvalue estimation will be employed with the assumption that

the dominant eigenvalue is complex and $\{\mathbf{v}_i\}$ is a complete set of eigenvectors. Start with the change in solution at step n , $\Delta \mathbf{x}^n = \sum_{i=1}^m \beta_i \mathbf{v}_i$, where the solution is a linear combination of the eigenvectors. Assuming $\lambda_1 = \bar{\lambda}_2$ and $|\lambda_1| = |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_m|$, the change in solution at step $n+k-r$ is

$$\Delta \mathbf{x}^{n+k-r} = \lambda_1^{k-r} \beta_1 \mathbf{v}_1 + \lambda_2^{k-r} \beta_2 \mathbf{v}_2 + O(|\lambda_3|^{k-r})$$

for $r = 0, 1$ and 2 . Thus, for any real c and d ,

$$\begin{aligned} \Delta \mathbf{x}^{n+k} + c \Delta \mathbf{x}^{n+k-1} + d \Delta \mathbf{x}^{n+k-2} \\ = \lambda_1^{k-2} \beta_1 (\lambda_1^2 + c \lambda_1 + d) \mathbf{v}_1 + \lambda_2^{k-2} \beta_2 (\lambda_2^2 + c \lambda_2 + d) \mathbf{v}_2 + O(|\lambda_3|^{k-2}). \end{aligned}$$

The coefficients of $\bar{\mathbf{v}}_1$ and $\bar{\mathbf{v}}_2$ equal zero if $c = -2 \operatorname{Re} \lambda_1$ and $d = |\lambda_1|^2$, so that $\lambda^2 + c \lambda + d = (\lambda - \lambda_1)(\lambda - \lambda_2)$. Therefore, the indices $i \neq j$ can be chosen and the following equations can be solved for c and d

$$\begin{aligned} (\Delta \mathbf{x}^n)_i + c (\Delta \mathbf{x}^{n-1})_i + d (\Delta \mathbf{x}^{n-2})_i &= 0 \\ (\Delta \mathbf{x}^n)_j + c (\Delta \mathbf{x}^{n-1})_j + d (\Delta \mathbf{x}^{n-2})_j &= 0. \end{aligned}$$

It can be shown that

$$\begin{aligned} c &= \frac{(\Delta \mathbf{x}^n)_i (\Delta \mathbf{x}^{n-2})_j - (\Delta \mathbf{x}^{n-2})_i (\Delta \mathbf{x}^n)_j}{(\Delta \mathbf{x}^{n-2})_i (\Delta \mathbf{x}^{n-1})_j - (\Delta \mathbf{x}^{n-2})_j (\Delta \mathbf{x}^{n-1})_i} \\ d &= \frac{(\Delta \mathbf{x}^n)_j (\Delta \mathbf{x}^{n-1})_i - (\Delta \mathbf{x}^n)_i (\Delta \mathbf{x}^{n-1})_j}{(\Delta \mathbf{x}^{n-2})_i (\Delta \mathbf{x}^{n-1})_j - (\Delta \mathbf{x}^{n-2})_j (\Delta \mathbf{x}^{n-1})_i}. \end{aligned}$$

Since c and d are known, the largest eigenvalue can be estimated by

$$\lambda_{\max} = \lambda_1 = \bar{\lambda}_2 = -\frac{c}{2} + i \sqrt{d - \frac{c^2}{4}} \quad (5.11)$$

Now that the largest eigenvalue is known, equation 5.10 can be used to annihilate the two largest complex eigenvalues.

Figure 5.4 - Estimated Dominant Eigenvalues
for Two Eigenvalue Annihilation

This eigenvalue annihilation was used at iterations 150, 190, 240, 275, 370, 470, 530, 575, 670 and 700. The estimated dominant eigenvalues at these steps are seen in figure 5.4. Since the magnitude of these eigenvalues is less than 1, the solution will converge by equation 5.2. As seen in figure 5.5, the results of this procedure are quite dramatic. This solution converged to the desired tolerance of 10^{-15} in 1080 iterations. This result is a reduction in the number of iterations by 29%. This method requires an extra 2273 flops for each annihilation step. To reach convergence, an extra 22,700 flops

need to be performed. The two eigenvalue annihilation reduces the number of iterations by a greater percentage than either the SOR or single eigenvalue annihilation. Also, it requires the least amount of extra work.

Figure 5.5 - Two Eigenvalue Annihilation

	Extra Work @ each step	Extra Work for convergence	Reduction of Iterations	% decrease
SOR	240	293,000	300	20%
Single eig.	4800	43,200	150	10%
Two eig.	2270	22,700	439	29%

Table 5.1 - Comparison of the Three Acceleration Methods

In conclusion, each of the acceleration methods produce a steady-state solution which is, at most, within 10^{-13} of the non-accelerated steady-state solution. Each method is produces accurate results. Given the information in table 5.1, the two eigenvalue annihilation method is clearly the better choice for acceleration. However, there are some drawbacks to using this method. One is that the transient effects in the error need to die off before annihilation can be performed. Annihilation done in the early stages of convergence destroys the solution. As Jespersen and Buning state [6] ,

It is clear that $\mathbf{B}$ is not a constant matrix, but if we wait until the later stages of the iteration, $\mathbf{B}$ will be changing very slowly, and we should be able to apply annihilation with a fair degree of success.

(The matrix $\mathbf{B}$ is the matrix of the system $x^{n+1} = \mathbf{B}x^n + c$). As seen in Figure 5.6, the solution is accelerated when the annihilation is done at iterations 50 and 80.

Figure 5.6 - Two Eigenvalue Annihilation in Early Stages

Second, the dominant eigenvalue is only an estimation. From theory, the exact eigenvalue can reduce the error even more substantially. Although MATLAB has a built in function to determine the eigenvalues, the set-up of the 120×120 matrix would be tedious and would require extra storage space. Also, the increase in the flop count would be considerable. The estimation only requires storing three consecutive solution vectors and performing twelve flops as opposed to the 4500 flops required for the single eigenvalue case. Since using the second estimation does reduce the residual considerably, it is by far more efficient to use the estimated eigenvalue than to find the actual eigenvalues. Furthermore, if more efficient estimations could be developed, then this annihilation method would be improved.

Figure 5.7 - Two Eigenvalue Annihilation at
Iterations 150, 210 and 260.

Lastly, the annihilation produces some side effects. As seen in figure 5.5, there are sharp jumps in the residual immediately following the annihilation step. Although this method produces a faster convergence rate, annihilation strategies that do not allow these jumps might be preferred [6]. In some cases the residual does not fall after these jumps occur. As seen in figure 5.7, the residual at iteration 260 does not fall after the two eigenvalues have been annihilated. The annihilation cannot be performed indiscriminately. Placement of the annihilation steps needs to be done with care. Annihilation performed at a step where the tangent to the residual is negative produces better results than where the tangent is positive. One reason for this behavior could be the annihilation tends to "push" the solution in the direction of the tangent.

Despite these drawbacks, the two eigenvalue annihilation method is the most acceptable acceleration method of the three which have been presented here. This method produces the best results and requires the fewest number of flops.

REFERENCES

1. Anderson, D. A., Tannehill, J. C., & Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Washington: McGraw-Hill Book Company (1984).
2. Cheer, A. & Saleem, M., 'Acceleration of Convergence by Shifting the Spectrum of Implicit Finite Difference Operators Associated with the Equations of Gas Dynamics', *International Journal for Numerical Methods in Fluids*, **12**, 443-462 (1991).
3. Eisberg, R. M. & Lerner, L. S., *Physics - Foundations and Applications*, New York: McGraw-Hill Book Company (1981).
4. Fox, R. W. & McDonald, A. T., *Introduction to Fluid Mechanics* (2nd ed.), New York: John Wiley & Sons (1973).
5. Hirose, A. & Lonngren, K. E., *Introduction to Wave Phenomena*, New York: John Wiley & Sons (1985).
6. Jespersen, D. C. & Bunning, P. G., 'Accelerating an Iterative Process by Explicit Annihilation', *SIAM J. Sci. Stat. Comput.*, **6**, 639 - 651 (July 1985).
7. John, F., *Partial Differential Equations* (4th ed.), Springer-Verlag (1982).
8. John, J. E. A., *Gas Dynamics*, Boston: Allyn and Bacon, Inc. (1969).
9. Karamcheti, K., *Principles of Ideal Fluid Aerodynamics*, New York: John Wiley & Sons (1966).
10. Kreyszig, E., *Advanced Engineering Mathematics* (4th ed.), New York: John Wiley & Sons (1979).
11. Landau, L. D. & Lifshitz, E. M., *Fluid Mechanics*, London: Pergamon Press (1959).
12. Liepmann, H. W. & Roshko, A., *Elements of Gasdynamics*, New York: John Wiley & Sons (1957).
13. MacCormack, R. W., 'A Perspective on a Quarter Century of CFD Research', AIAA Paper No. 93-3291-CP (1993).
14. Reynolds, W. C. & Perkins H. C., *Engineering Thermodynamics*, New York: McGraw-Hill Book Company (1977).

15. Richardson, P. F., Dwoyer, D. L., Kutler, P. & Povinelli, L. A., 'The NASA Computational Fluid Dynamics (CFD) Program Building Technology to Solve Future Challenges', AIAA Paper No. 93-3292-CP (1993).
16. Sakell, L., 'The Status of CFD: An Air Force Perspective', AIAA Paper No. 93-3293-CP (1993).
17. Smith, G. D., *Numerical Solution of Partial Differential Equations: Finite Difference Methods* (3rd ed.), Oxford: Clarendon Press (1985).
18. Warming, R. F. & Beam, R. M., 'On the Construction and Application of Implicit Factored Schemes for Conservation Laws', *SIAM-AMS Proceedings*, **11**, 85-129 (1978)
19. MATLAB for Microsoft Windows, Version 4.0, The MathWorks, Inc. (1992)

APPENDIX A

PROGRAM LISTING

```

% This program solves the one-dimensional Euler equations for airflow through a nozzle.
% A Flux Splitting algorithm is used to solve these equations. An explicit method is used
% to solve the equations. MATLAB 4.0 for MS-Windows was used for this program.
%
clear
%
% Set-up grid
%
n = 40;
x = linspace(0,1,n+1);
dx = 1 / n;
time = 0;
%
%Set-up initial data
%
gamma = 1.4;
%
% Entrance conditions
%
M(1) = 2.3;
p(1) = .1081;
rho(1) = .2596;
a = area(x);
u(1) = M(1) * sqrt(gamma*p(1)/rho(1));
e(1) = p(1)/(gamma-1) + rho(1) * u(1)^2/2;
c1 = rho(1) * u(1) * a(1);
c2 = (e(1) + p(1))/ rho(1);
%
% Initial conditions
%
da(1:n) = a(1:n) - area(x(1:n) - dx);
da(n+1) = a(n+1) - a(n);
for i = 1:n
    if i == 36
        drho(i) = rho(i)*(M(i)^2-1)/((gamma-1)*M(i)^2/2+1);
    
```

```

else
    drho(i) = rho(i) * M(i)^2 * da(i)/((1-M(i)^2)*a(i));
end
rho(i+1) = rho(i) + drho(i);
u(i+1) = c1 / (rho(i+1) * a(i+1));
p(i+1) = rho(i+1)*(c2-.5*u(i+1)^2)*(1-1/gamma);
e(i+1) = rho(i+1)*c2-p(i+1);
M(i+1) = sqrt(u(i+1)^2 * rho(i+1) / (gamma*p(i+1)));
end
sa = sqrt(gamma .* p ./ rho);
%
% Store initial values
%
uinit=u;
pinit = p;
rinit = rho;
Minit = M;
einit = e;
nrho = rho;
np = p;
nu = u;
ne = e;
nM = M;
nsa = sa;
oldp = p(n+1);
%
%Initialize all matrices and parameters
%
ntime = 0;
q = zeros(3,n+1);
s = zeros(3);
sinv = zeros(3);
t = zeros(3);
tinv = zeros(3);
poslam = zeros(3);
neglam = zeros(3);
rhs = zeros(3,n+1);
j = 1;
conver = 0;           % Set toggle for convergence.
epsilon = .001;       % Smoothing factor for eigenvalues
w = 1.25;             % Relaxation term
ev = eye(41);
k = 1;               % Counter for eigenvalue estimation

```

```

%
%Start iterating
%
while conver == 0
j
lammax = u(1) + sqrt(gamma * p(1) / rho(1));
q(1,:) = rho;
q(2,:) = rho .* u;
q(3,:) = e;
lam(1,:) = u;
lam(2,:) = u + sa;
lam(3,:) = u - sa;
cfl = max(max(lam));
if (lammax < cfl)
    lammax = cfl;
end
dt = .9 * dx / lammax;          % Compute the time step
time = time + dt;
dqnew = fa(1:3,q,a,da,dt,dx,n); % Determine delta Q
qnew = q(:,1:n) + dqnew';
%*****
%
%                                RELAXATION
%
%qnew = w * qnew + (1 - w) * q(:,1:n);
%
%*****
nrho(2:n) = qnew(1,2:n);
nu(2:n) = qnew(2,2:n) ./ nrho(2:n);
ne(2:n) = qnew(3,2:n);
np(2:n) = (gamma - 1) * (ne(2:n) - .5 * nrho(2:n) .* nu(2:n).^2);
nsa(2:n) = sqrt(gamma * np(2:n) ./ nrho(2:n));
nM(2:n) = nu(2:n) ./ nsa(2:n);
%
% Last node using characteristic theory (Exit Boundary Condition)
%
pdif = p(n+1)-p(n);
sa(n+1) = sqrt(gamma * p(n+1) / rho(n+1));
a1 = u(n+1) * dt / dx;
R1 = - a1 / (1+a1) * (rho(n+1) - rho(n) - pdif / sa(n+1)^2);
a2 = (u(n+1) + sa(n+1)) * dt / dx;
R2 = - a2 / (1+a2) * (pdif + rho(n+1) * sa(n+1) * (u(n+1) - u(n)));
a3 = (u(n+1) - sa(n+1)) * dt / dx;
R4 = - a3 / (1+a3) * (pdif - rho(n+1) * sa(n+1) * (u(n+1) - u(n)));

```

```

if M(n+1) > 1
    np(n+1) = p(n+1) + (R2 + R4) / 2;
else
    np(n+1) = p(n+1);
end
nrho(n+1) = rho(n+1) + R1 + (np(n+1) - p(n+1)) / sa(n+1)^2;
nu(n+1) = u(n+1) + (R2 - np(n+1) + p(n+1)) / (rho(n+1) * sa(n+1));
ne(n+1) = np(n+1) / (gamma - 1) + .5 * nrho(n+1) * u(n+1)^2;
nsa(n+1) = sqrt(gamma * np(n+1) / nrho(n+1));
nM(n+1) = nu(n+1) / nsa(n+1);
%
%Checking for convergence
%
nerr(j) = norm((nrho-rho),inf);
if nerr(j) <= 1e-15      %Checks to see if tolerance has been met
    iter = j;
    conver = 1;          %Toggle to end while loop
else
    j = j + 1;
end
%
%*****
%          BEGIN SINGLE EIGENVALUE ANNIHILATION PRODEDURE
%
lambda(j-1) = norm(qnew) / norm(q);
%if (j-1 == 130) | (j-1 == 1900) | (j-1 == 2500) | (j-1 == 3500) | (j-1 == 5000) |
%    (j-1 == 6000) | (j-1 == 5500) | (j-1 == 6500) | (j-1 == 7250) | (j-1 == 8500) |
%    (j-1 == 9500) | (j-1 == 9500)% | (j-1 == 10250)
% k = 1
%
%Explicit Annihilation
%
% nrho = (nrho - lambda(j-1) * rho) / (1 - lambda(j-1));
% nu = ( nu - lambda(j-1) * u) / (1 - lambda(j-1));
% np = ( np - lambda(j-1) * p) / ( 1 - lambda(j-1));
% ne = ( ne - lambda(j-1) * e) / ( 1 - lambda(j-1));
% nM = ( nM - lambda(j-1) * M) / ( 1 - lambda(j-1));
%end
%          END SINGLE EIGENVALUE ANNIHILATION PROCEDURE
%*****
%*****
%          BEGIN TWO EIGENVALUE ANNIHILATION PROCEDURE
%

```



```

%
% Iteration number where the annihilation is performed
%
test1 = 150;
test2 = 180;
test3 = 210;
test4 = 400;
test5 = 470;
test6 = 575;
test7 = 40000;
test8 = 4800;
test9 = 5500;
test10 = 7300;
%
% Store three consecutive solutions for use in the annihilation procedure
% Save solution for step n-2 in eigenvalue estimation
%
if (j == test1) | (j == test2) | (j == test3) | (j == test4) | (j == test5) | (j == test6) |
    (j == test7) | (j == test8) | (j == test9) | (j == test10)
    q1(1,:) = nrho;
    q1(2,:) = nrho .* nu;
    q1(3,:) = ne;
    D = dqnew(40,1);
    G = dqnew(35,1);
end
%
% Save solution for step n-1 in eigenvalue estimation
%
if (j == test1+1) | (j == test2+1) | (j == test3+1) | (j == test4+1) | (j == test5+1) |
    (j == test6+1) | (j == test7+1) | (j == test8+1) | (j == test9+1) | (j == test10+1)
    q2(1,:) = nrho;
    q2(2,:) = nrho .* nu;
    q2(3,:) = ne;
    B = dqnew(40,1);
    F = dqnew(35,1);
end
%
% Save solution for step n in eigenvalue estimation
%
if (j == test1+2) | (j == test2+2) | (j == test3+2) | (j == test4+2) | (j == test5+2) |
    (j == test6+2) | (j == test7+2) | (j == test8+2) | (j == test9+2) | (j == test10+2)
    q3(1,:) = nrho;
    q3(2,:) = nrho .* nu;

```

```

q3(3,:) = ne;
A = dqnew(40,1);
E = dqnew(35,1);
%
% Estimate the largest eigenvalue
%
c = (A * G - D * E) / (D * F - G * B);
d = (E * B - A * F) / (D * F - G * B);
maxlam(k) = -c / 2 + sqrt(-1) * (sqrt(d - c^2 / 4));
maxlam(k) = 1 / (1 - maxlam(k));
%
% Explicit two eigenvalue annihilation
%
denom = 1 - 2 * real(maxlam(k)) + abs(maxlam(k))^2;
qimp = (q3 - 2 * real(maxlam(k)) * q2 + abs(maxlam(k))^2 * q1) / denom;
%
% Save improved solution in the solution vector
%
nrho(2:n+1) = qimp(1,2:n+1);
nu(2:n+1) = qimp(2,2:n+1) ./ nrho(2:n+1);
ne(2:n+1) = qimp(3,2:n+1);
np(2:n+1) = (gamma - 1) * (ne(2:n+1) - .5 * nrho(2:n+1) .* nu(2:n+1).^2);
nsa(2:n+1) = sqrt(gamma * np(2:n+1) ./ nrho(2:n+1));
nM(2:n+1) = nu(2:n+1) ./ nsa(2:n+1);
nerr(j) = norm((nrho-rho),inf);
j = j + 1;
k = k + 1;
end
%
%
% END TWO EIGENVALUE ANNIHILATION
%*****
%
% Stores new values into the variables
%
p = np;
rho = nrho;
u = nu;
sa = nsa;
M = nM;
e = ne;
%
% End of iteration loop
%
```

```
end  
result = [p' rho' M'];
```

```

%
% This function is called by flux.m. This function calculates the  $\Delta Q$  term from the main
% program.
%
% Inputs: k - row number of solution to be sent to main program
%         qa - input vector
%         ar - area from main program
%         da - delta area from main program
%         dx,dt - space and time difference used in main program
%         n - number of nodes in x-direction
%
% Output: ans - F(qa)
%
function ans = fa(k,qa,ar,da,dt,dx,n)
    gamma = 1.4;
    epsilon = .001;
    s = zeros(3);
    sinv = zeros(3);
    t = zeros(3);
    tinv = zeros(3);
    p = (gamma - 1) * (qa(3,:) - .5 * qa(2,:).^2 ./ qa(1,:));
    for i = 1:n+1
        sa = sqrt (gamma * p(i) / qa(1,i));
        s(1,1) = 1;
        s(2,1) = qa(2,i) / qa(1,i);
        s(2,2) = qa(1,i);
        s(3,1) = .5 * s(2,1) ^ 2;
        s(3,2) = qa(2,i);
        s(3,3) = 1 / (gamma-1);
        sinv(1,1) = 1;
        sinv(2,1) = -s(2,1) / qa(1,i);
        sinv(2,2) = 1 / qa(1,i);
        sinv(3,1) = .5 * s(2,1)^2 * (gamma-1);
        sinv(3,2) = s(2,1) * (1 - gamma);
        sinv(3,3) = gamma - 1;
        t(1,1) = 1;
        t(1,2) = qa(1,i) / (gamma * p(i));
        t(1,3) = t(1,2);
        t(2,2) = 1 / (qa(1,i) * sa);
        t(2,3) = -t(2,2);
        t(3,2) = 1;
        t(3,3) = 1;
        tinv(1,1) = 1;

```

```

tinv(1,3) = - qa(1,i) / (gamma * p(i));
tinv(2,2) = .5 * qa(1,i) * sa;
tinv(2,3) = .5;
tinv(3,2) = -tinv(2,2);
tinv(3,3) = .5;
lam(1) = s(2,1);
poslam(1,1) = .5 * (lam(1) + sqrt(lam(1)^2 + epsilon));
lam(2) = s(2,1) + sa;
poslam(2,2) = .5 * (lam(2) + sqrt(lam(2)^2 + epsilon));
lam(3) = s(2,1) - sa;
if (lam(3) > 0)
    poslam(3,3) = .5 * (lam(3) + sqrt(lam(3)^2 + epsilon));
    neglam(3,3) = 0;
else
    poslam(3,3) = 0;
    neglam(3,3) = .5 * (lam(3) - sqrt(lam(3)^2 + epsilon));
end
%
% Form fluxes
%
fplus = s * t * poslam * tinv * sinv * qa(:,i) * ar(i);
fminus = s * t * neglam * tinv * sinv * qa(:,i) * ar(i);
afp(i,:) = fplus';
afm(i,:) = fminus';
end
delt = dt ./ (ar(2:n) * dx);
rhs = zeros(3,n);
rhs(2,:) = dt .* p(1:n) .* da(1:n) ./ (ar(1:n) * dx);
for i = 2:n
    delaafp(i-1,:) = afp(i,:) - afp(i-1,:);
    delaafm(i-1,:) = afm(i+1,:) - afm(i,:);
    ans(:,i) = -delt(i-1) * (delaafp(i-1,:) + delaafm(i-1,:)) + rhs(:,i);
end
ans = ans(k,:);

```

```
%  
% This function is called by flux.m. This function calculates the area of a nozzle.  
%  
% Input: x - distance along the nozzle  
%  
function [sol] = area(x)  
n = length(x);  
sol = 1 + 4 * ((x - .5).^2);
```

APPENDIX B

NOMENCLATURE

- A** Jacobian
- $\tilde{\mathbf{A}}$ Matrix with eigenvalue u , $u + a$ and $u - a$
- A Cross-sectional area of the nozzle
- a Local speed of sound
- e total energy
- M Mach number
- p Pressure
- Q** Vector of conserved variables
- $\tilde{\mathbf{Q}}$ Vector of primitive variables
- T** Matrix containing eigenvectors of $\tilde{\mathbf{A}}$
- t Time
- u Velocity in the x-direction
- x Distance along the nozzle
- γ Ratio of specific heats, equal to 1.4
- λ Eigenvalues of the Euler equations
- ρ Density