1993

Evaluation of multigrid acceleration for a coupled, strongly implicit procedure for the Navier-Stokes equations

Robert L. Cupples
Iowa State University

Evaluation of multigrid acceleration for a coupled, strongly implicit procedure for the Navier-Stokes equations

by

Robert L. Cupples

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Mechanical Engineering
Major: Mechanical Engineering

Approved:

_____
In Charge of Major Work

_____
For the Major Department

_____
For the Graduate College

Iowa State University
Ames, Iowa
1993

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

a	combination of metric, viscous, and/or conduction terms
a	area
c	speed of sound
c_p	specific heat at constant pressure
c_v	specific heat at constant volume
$dtrat$	scaling factor for the pseudo-time steps
e	internal energy per unit mass
$\mathbf{e}$	algebraic error vector
$\mathbf{f}$	general right-hand side vector
$\vec{g}$	vector notation for gravity
h	fine grid
k	thermal conductivity
k	pseudo-time iteration index
n	time (or general) iteration index
p	pressure
p_o	reference pressure
p_g	gauge pressure
$\vec{q}$	vector notation for heat conduction

$\dot{q}$	heat generation
r	radial direction
$\mathbf{r}$	residual vector
t	time
u	velocity in the x direction
u_i	index notation for velocity
$\mathbf{u}$	exact solution vector
$\tilde{\mathbf{u}}$	current solution vector
v	velocity in the y direction
$\mathbf{v}$	exact correction vector
$\tilde{\mathbf{v}}$	current correction vector
x	spatial coordinate
x_i	index notation for coordinates
y	spatial coordinate
$\mathbf{A}$	coefficient matrix of a general linear system
$A^1_{i,j}$ - $A^9_{i,j}$	elements of the coefficient matrix
C_1 C_2	Sutherland constants
CS	correction scheme
E	inviscid flux in the x direction
$\tilde{E}$	inviscid flux in the ξ direction
E_v	viscous flux in the x direction
$\tilde{E}_v$	viscous flux in the ξ direction
E_t	total energy per unit volume
F	inviscid flux in the y direction

$\tilde{F}$	inviscid flux in the η direction
F_v	viscous flux in the y direction
$\tilde{F}_v$	viscous flux in the η direction
FAS	full approximation scheme
GS	Gauss-Seidel
H	coarse grid
H	total enthalpy
I	transfer operator
J	transformation Jacobian
$\mathbf{L}$	coefficient matrix of a general linear system
M	Mach number
$\mathbf{N}$	nonlinear operator
Q	external energy
Q	Cartesian vector of conservative variables
$\hat{Q}$	transformed vector of conservative variables
$\tilde{Q}$	vector of unknowns
R	gas constant
Re	Reynolds number
SIP	strongly implicit procedure
T	temperature
U	contravariant velocity component
$\vec{V}$	vector notation for velocity

Greek Symbols

α	thermal diffusivity
α β	parameters in the Roberts grid transformation
β	scaling factor in the preconditioning matrix
Γ	preconditioning matrix
γ	Ratio of constant specific heats
Δ	increment of change
δ	Kronecker delta function
ϵ	convergence criterion
η	transformed coordinate
θ	circumferential direction
λ	eigenvalue
μ	dynamic viscosity
ν_1 ν_2	multigrid V-cycle control parameters
ξ	transformed coordinate
ρ	density
σ	explicit smoothing term
σ_1 σ_2	parameters controlling the explicit smoothing
τ	pseudo time
τ_{xx}	shear stress
τ_{xy}	shear stress
τ_{yy}	shear stress
Φ	dissipation function
ϕ	general dependent variable

Subscripts

h	value defined on the fine grid
i	index in the x or ξ direction
j	index in the y or η direction
max	maximum value
new	updated value
old	provisional value
ref	reference value
x	derivative with respect to x
y	derivative with respect to y
H	value defined on the coarse grid
η	derivative with respect to η
ξ	derivative with respect to ξ

Superscripts

h	value defined on the fine grid
H	value defined on the coarse grid
$*$	nondimensional value

ACKNOWLEDGMENTS

This thesis is a product of the ongoing project "Numerical Simulation of Flows in Combustion Systems" funded by the Aerothermochemistry Branch of the National Aeronautics and Space Administration's Lewis Research Center through the Graduate Student Researchers Program under grant NGT-50633.

The author graciously thanks the Aerothermochemistry Branch for its funding and in particular Dr. Kuo-Huey Chen for his time, patience, and previous work.

1. INTRODUCTION

Many attribute the development and first use of multigrid techniques in their present form to Achi Brandt of the Weizmann Institute in the early 70's with the "Multi-Level Adaptive Technique (MLAT)". Brandt [1] claims that the earliest application he knows of is a two-level scheme known as Southwell's acceleration of relaxation by "group relaxation" [2] and that the first to describe a recursive scheme with more than two levels was Fedorenko [3]. Apparently, the earliest works did not exhibit a complete understanding of the multigrid method and were rather crude and inefficient compared with Brandt's techniques of the present. Since the early work of Brandt, many researchers have made contributions through the application and further development of multigrid methods to a wide variety of problems including not only the field of fluid dynamics but also in the areas of semiconductor device simulation, image reconstruction, statistics, general relativity, and linear programming to name a few.

In this thesis, the author will present some of his work toward the application of multigrid to a two-dimensional Navier-Stokes solver for chemically reacting flows. The work began with application of multigrid to the two-dimensional Laplace's equation with as many as four grids presented in Chapter 3 which is preceded by an introduction to the multigrid method in Chapter 2. Laplace's equation was solved

using both Gauss-Seidel iteration and the strongly implicit procedure.

The main purpose of the research on Laplace's equation was to learn some of the properties of the multigrid procedure under various conditions so as to guide the development of an efficient multigrid procedure for the Navier-Stokes equations. More recent work involves a coupled, strongly implicit Navier-Stokes solver with two levels presented in Chapter 4. Results were obtained mainly for the flow in a driven cavity, but more limited results are also reported for the developing channel. The author believes this work is unique in that few results have been reported on the use of multigrid with either the coupled strongly implicit procedure or an all-Mach number formulation.

2. THE MULTIGRID METHOD

There are two major ideas behind multigrid methods. The first concept can be seen within the name itself. The word multigrid implies that multiple grids are being used, as is the case. A successive series of coarser grids are used based upon the grid desired to model a problem. This assumes for the purposes of this study that the finest grid is a fixed, two-dimensional, structured grid and that a coarser grid can be formed by removing every second grid line while leaving the grid lines along the boundary intact. Any number of multiple grids can be used as long as this rule for generating the next coarser grid can be followed. For purposes of simplifying the discussion of multigrid methods, a multigrid system of only two grids will be assumed throughout the rest of this chapter, but extension to further levels is rather straightforward.

The second main concept utilized by the multigrid strategy is embodied in how the equations are represented on the coarse grid. The equations solved on the coarse grid are not in the same form as the equations being solved on the fine grid. Instead, they are rewritten such that they take into account the amount by which the fine grid equations are not satisfied. This is known as the residual form of the equations. The equations in residual form are not solved for the variables of interest on the fine grid, but for the amount by which the solution variables need to change by in order

to satisfy the fine grid equations.

Multigrid uses a series of coarser grids to temporarily approximate the solution to a set of equations. The reason for using coarser grids is more than the obvious reason: fewer points implies fewer equations to solve, but there is also an additional reason which requires some explanation. In general, problems being solved with fewer grid points also require fewer iterations.

There are two viewpoints on why coarse grids solve problems in fewer iterations. Both can be seen by examining how a solution converges. Denote the difference between a candidate solution to a problem and the exact solution as the "error" or "error function". This error function can be represented using a Fourier series: a collection of sine waves of different frequencies and amplitudes whose superposition approximates the error function. The higher frequency components are quickly removed by a given iterative process because the waves extend for only a few nodes and most discretizations are particularly well suited for local "smoothing". However, the lower frequency waves extend over many more grid points and require more iterations to be dissipated. This trend can be noticed in several problems by examining the convergence history. Many problems converge rapidly in the first few iterations and then slow down. This is due to the higher frequency sine waves being removed rapidly while the lower frequency waves remain behind.

A second point of view is to examine the rate at which information propagates. As an example, think of a thin rod insulated along the sides approximated as a one-dimensional problem. The rod is 100 C everywhere except at the end points where the temperature is fixed at 0 C. The steady-state temperature distribution is desired. Obviously, the entire rod will eventually become 0 C at steady state, and the problem

does not require a numerical solution, but it will serve as a good example.

The difference expression for Laplace's equation for this one-dimensional problem can be written as $T_i = 1/2\left(T_{i+1} + T_{i-1}\right)$. At first, the initial distribution satisfies this equation except near the boundaries. If Gauss-Seidel is used to smooth the error function, changes will only occur near the boundaries and work their way toward the center approximately one grid point per iteration. Thus, the temperature at the edges changes rapidly after the first few iterations leaving an error function untouched at the center of the rod until information propagates there. After several iterations, an intermediate error function will be rather smooth, nearly satisfying the equations being solved, and thus slowing the iteration process. Information needs to travel from the endpoints to the center, and the more points from the ends to the center, the more iterations needed for this information to travel. On the other hand, if the rod is modeled with only one interior point, the problem can be solved in a single iteration.

Therefore, if every other grid point were dropped from the grid as shown in Figure 2.1, the error function would be easier to smooth since information would travel about twice as far per iteration, requiring fewer iterations to solve the problem.

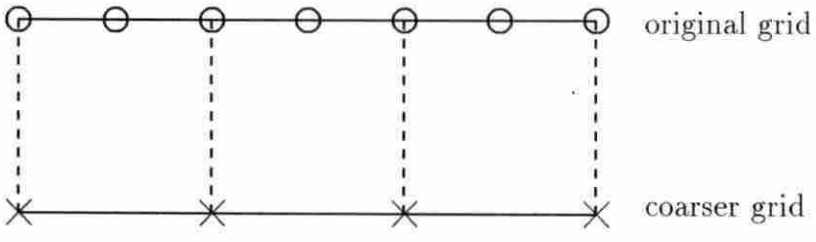

Figure 2.1: One-dimensional rod example

After looking at this simple example, a coarser grid (even a single interior point) could have been used in the first place, bypassing the need for multigrid. However, many problems can not be resolved adequately by a single point. They require that some higher degree of resolution be maintained depending upon the problem and the desired accuracy in order to be modeled properly. This brings back the second main concept: retaining the fine grid resolution on a coarser grid.

Let

$$\mathbf{A}\mathbf{u} = \mathbf{f} \tag{2.1}$$

be a system of linear equations with $\mathbf{u}$ denoting the exact solution to the system. The exact solution is not known but an approximation to $\mathbf{u}$ is and will be labeled $\tilde{\mathbf{u}}$. The difference between $\mathbf{u}$ and $\tilde{\mathbf{u}}$ is the algebraic error, $\mathbf{e}$:

$$\mathbf{e} = \mathbf{u} - \tilde{\mathbf{u}}$$

Unfortunately, neither $\mathbf{u}$ nor $\mathbf{e}$ are known and therefore $\mathbf{e}$ is limited in its usefulness. However a computable rating of how well $\tilde{\mathbf{u}}$ satisfies the system of equations can be formed by what will be known as the *residual* or *residualfunction* defined as:

$$\mathbf{r} = \mathbf{f} - \mathbf{A}\tilde{\mathbf{u}} \tag{2.2}$$

The residual is a measure of the extent to which the current approximation $\tilde{\mathbf{u}}$ satisfies the original equation. Notice that $\mathbf{r} = 0$, if and only if, $\mathbf{u} = \tilde{\mathbf{u}}$ and thus $\mathbf{e} = 0$. Equation (2.2) can be rewritten as

$$\mathbf{A}\tilde{\mathbf{u}} = \mathbf{f} - \mathbf{r} \tag{2.3}$$

If equation (2.3) is subtracted from equation (2.1), then an expression is formed relating the residual to the error:

$$\mathbf{A}\mathbf{e} = \mathbf{r} \tag{2.4}$$

This equation is known as the residual equation and plays an important part in multigrid methods. The equation shows that $\mathbf{e}$ satisfies the same equation that $\mathbf{u}$ satisfies when $\mathbf{f}$ is replaced with $\mathbf{r}$.

The residual equation (2.4) can be used to calculate $\mathbf{e}$ and thus improve the approximation $\tilde{\mathbf{u}}$. This is accomplished by calculating the residual using the current fine grid solution, transferring the residual to the coarse grid in a process termed *restriction* (discussed in a later section), and solving the residual equation on the coarse grid. Thus, the residual equation acts as a fine grid version of the coarse grid equation. If equation (2.1) were being solved on the coarse grid instead, the coarse grid problem would merely be the fine grid problem with a poorer resolution. The residual helps to link the coarse grid to the fine grid by acting as a source term representing the amount by which the fine grid solution needs to be adjusted to satisfy the fine grid equations.

When the time comes to apply the correction computed on the coarse grid to the fine grid, the correction $\mathbf{e}$ is added to $\tilde{\mathbf{u}}$ on the fine grid in a process termed *prolongation* (also discussed in a later section). After the fine grid solution is updated, the procedure is not yet finished. The fine grid equation is usually iterated upon a few times to smooth any errors introduced by the multigrid process, and the process repeats itself in a cyclic fashion until the fine grid equation has converged.

In summary, multigrid methods are desirable because fewer equations need to be solved per iteration on the coarse grid, the coarse grid representation of the error function is easier to smooth, and the residual equation on the coarse grid provides a connection to the fine grid problem. Two multigrid strategies, the correction scheme and the full approximation scheme, will be illustrated. After that, explanation will

be provided as to how the problem is transferred from the fine grid to the coarse grid (restriction), and then how the problem is later transferred back to the fine grid (prolongation).

2.1 Correction Scheme

There are two common multigrid strategies: the correction scheme and the full approximation scheme. The correction scheme (CS) is easier to apply but is not appropriate for nonlinear equations. On the other hand, the full approximation scheme (FAS) is well suited for nonlinear equations, but requires additional programming effort. According to Brandt [4], the two schemes provide equivalent results within roundoff error when applied to linear equations. Therefore, in this study, the CS method was applied to linear equations and the FAS method was applied to nonlinear equations. In this section, only the CS method will be described.

Consider a linear problem $\mathbf{L}^h\mathbf{u}^h = \mathbf{f}^h$ where the superscript h denotes the fine grid upon which the problem is represented. Let $\mathbf{u}^h$ be the exact solution on h. While $\mathbf{u}^h$ is not known, its approximation, labeled $\tilde{\mathbf{u}}^h$, is. Likewise, $\mathbf{L}^h$ and $\mathbf{f}^h$ are $\mathbf{L}$ and $\mathbf{f}$ defined on h. A representation of this problem on a coarser grid denoted by H is desired in order to quickly smooth errors and provide a better approximation to the solution. The variable being sought on the fine grid by using the coarse grid is the correction $\mathbf{v}^h$ which is related to $\mathbf{u}^h$ and $\tilde{\mathbf{u}}^h$ by $\mathbf{v}^h = \mathbf{u}^h - \tilde{\mathbf{u}}^h$. The correction is found by solving the residual equation on the coarse grid which is

$$\mathbf{L}^H\mathbf{v}^H = I_h^H\mathbf{r}^h \tag{2.5}$$

where $\mathbf{L}^H$ is $\mathbf{L}$ computed on the coarse grid. The residual $\mathbf{r}^h$ is calculated on the

fine grid by

$$\mathbf{r}^h = \mathbf{f}^h - \mathbf{L}^h \tilde{\mathbf{u}}^h$$

and transferred to the coarse grid by the restriction operator I_h^H.

After the iteration process has finished with equation (2.5), its approximate solution $\tilde{\mathbf{v}}^H$ is prolongated to the fine grid and serves as a correction to the fine-grid solution:

$$\tilde{\mathbf{u}}_{new}^h = \tilde{\mathbf{u}}_{old}^h + I_H^h \tilde{\mathbf{v}}^H \tag{2.6}$$

where I_H^h represents a prolongation operator acting upon $\tilde{\mathbf{v}}^H$. After the corrections have been prolongated, the fine grid solution is iterated upon for a few iterations to smooth any errors introduced by restriction and prolongation. This cycle is repeated until the fine grid problem has converged.

2.2 Full Approximation Scheme

The full approximation scheme (FAS) presented here is suitable for nonlinear, unsteady problems such as the Navier-Stokes equations and can be found in such papers as Jameson [5] and Smith [6]. The procedure starts with a system of nonlinear equations

$$\mathbf{N}^h \left(\mathbf{u}^h \right) = \mathbf{f}^h \tag{2.7}$$

where $\mathbf{N}^h$ is a nonlinear operator acting upon the solution. As with the previous section, h indicates a quantity represented on the fine grid.

The first step is to linearize the equation with respect to $\mathbf{u}$. This can be accomplished with the result

$$\frac{\partial \mathbf{N}}{\partial \mathbf{u}} \Delta \mathbf{u}^h = \mathbf{f}^h - \mathbf{N}^h \left(\tilde{\mathbf{u}}^h \right) \tag{2.8}$$

where the solution is updated by

$$\tilde{\mathbf{u}}^h_{new} = \tilde{\mathbf{u}}^h_{old} + \Delta\mathbf{u}^h$$

The term $\partial\mathbf{N}/\partial\mathbf{u}$ is nothing more than a Jacobian matrix while $\tilde{\mathbf{u}}^h$ represents a provisional solution on the fine grid. Equation (2.8) is the linearized version of equation (2.7) and will be solved on the fine grid. As this system converges ($\Delta\mathbf{u}^h \rightarrow 0$), the nonlinear equation (2.7) is satisfied. Note that the right-hand side of the above equation is the definition of the residual for CS which will also be used here for FAS. It is important to point out that this residual has the same properties as the residual for CS did in that it approaches zero as the original problem is satisfied.

The coarse grid equation is written as

$$\frac{\partial\mathbf{N}}{\partial\mathbf{u}}\Delta\mathbf{u}^H = \mathbf{r}^H\left(\tilde{\mathbf{u}}^H\right) - \mathbf{r}^H\left(\hat{I}^H_h\tilde{\mathbf{u}}^h\right) + I^H_h\mathbf{r}^h\left(\tilde{\mathbf{u}}^h\right) \tag{2.9}$$

where the terms from left to right on the right-hand side are as follows:

- The first term is the coarse grid residual function evaluated using solution variables defined on the coarse grid.

$$\mathbf{r}^H\left(\tilde{\mathbf{u}}^H\right) = \mathbf{f}^H - \mathbf{N}^H\left(\tilde{\mathbf{u}}^H\right)$$

- The second term is the coarse grid residual function evaluated using solution variables restricted from the fine grid to the coarse grid and therefore remains fixed on the coarse grid. Note that the restriction operator shown has a caret above it. This denotes that the restriction operator for the residual and the restriction operator for the solution variables need not be the same.

$$\mathbf{r}^H\left(\hat{I}^H_h\tilde{\mathbf{u}}^h\right) = \mathbf{f}^H - \mathbf{N}^H\left(\hat{I}^H_h\tilde{\mathbf{u}}^h\right)$$

- The third term is the restricted fine grid residual function and therefore also remains fixed on the coarse grid.

$$I_h^H \mathbf{r}^h \left(\tilde{\mathbf{u}}^h\right) = I_h^H \left(\mathbf{f}^h - \mathbf{N}^h \left(\tilde{\mathbf{u}}^h\right)\right)$$

It should be noted that the first and second terms cancel each other out at the first coarse grid iteration leaving the restricted fine grid residual as the initial driving force. If the fine grid residual is zero then both the fine and coarse grid equations are satisfied. Upon convergence, the right-hand side of the equation (2.9) is zero. Therefore, terms one and two "absorb" the restricted fine grid residual. In other words, the restricted fine grid residual becomes the difference between the coarse grid residual evaluated with the restricted solution (which remains fixed) and the coarse grid residual evaluated with the final solution. Thus on the coarse grid, the fine grid solution is corrected by the fine grid residual.

The coarse grid equation is iterated upon until some criteria is met and the problem is then transferred back to the fine grid. It should be pointed out that as the coarse grid equation is being solved, it is necessary to update $\tilde{\mathbf{u}}^H$ every iteration in order to evaluate new Jacobians on the left-hand side. The $\Delta \mathbf{u}^H$ for the FAS scheme is the set of changes computed and applied after each coarse grid iteration, whereas for the CS scheme, $\tilde{v}^H$ was the cumulative change computed on the coarse grid which was finally applied to correct the fine grid solution at the completion of a set of coarse grid iterations.

Having obtained an approximate solution to $\mathbf{u}^H$, the corrected fine grid solution becomes

$$\tilde{\mathbf{u}}_{new}^h = \tilde{\mathbf{u}}_{old}^h + I_H^h \left(\tilde{\mathbf{u}}^H - I_h^H \tilde{\mathbf{u}}_{old}^h\right) \tag{2.10}$$

Equation 2.10 is the parallel of Equation 2.6. To use directly

$$\tilde{\mathbf{u}}^h_{new} = I^h_H \tilde{\mathbf{u}}^H$$

is inadvisable since it would introduce the interpolation errors of the full solution $\tilde{\mathbf{u}}^H$ instead of the smaller magnitude correction.

2.3 Restriction

In order to solve the coarse grid problem with either scheme, the residual computed on the fine grid needs to be represented on the coarse grid. In addition, the solution variables themselves also need to be transferred to the coarse grid when using the FAS method. This transfer is called restriction and is commonly done in one of two ways: direct injection or weighting. Since the coarse grid was generated by removing every other line from the fine grid, each point on the coarse grid shares a position in common with a fine grid point. Since all of the coarse grid points share their positions with points on the fine grid, one might conclude that a residual (or solution variable) at a point on the coarse grid might as well be equal to the residual calculated at the point on the fine grid with the same location. This is known as direct injection (Figure 2.2). The value of the residual at a coarse grid point is the value of the residual at the corresponding fine grid point.

A few comments on Figure 2.2 should be made. The circles in the figure represent coarse grid points while the crosses represent fine grid points. The arrows indicate the path by which the residuals are transferred from one grid to another. Note that some points have both circles and crosses on them. This is to emphasize that for each of these points on a given grid, there is a corresponding point on the other grid with

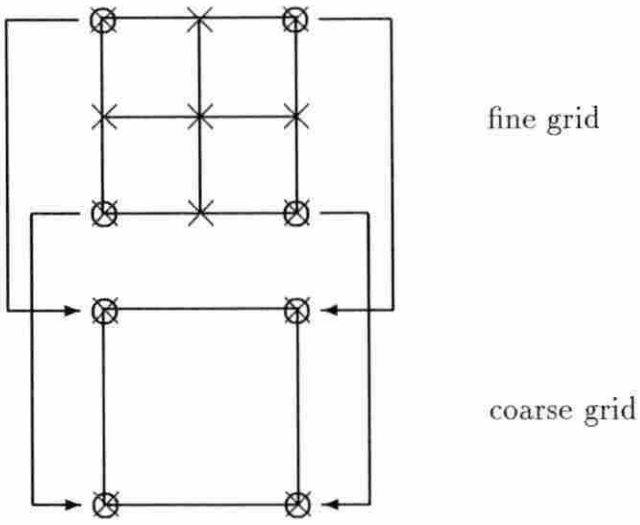

Figure 2.2: Restriction by direct injection

the same location in the domain being modeled. This same labeling methodology will be used in figures throughout the thesis.

An alternative to direct injection (illustrated in Figure 2.3) is to weight the value of the residual calculated at the shared point with the values of the residual calculated at the neighboring fine grid points before transferring the residual directly to the coarse grid. Weighting operators suggested for this type of transfer vary somewhat from investigator to investigator, but all seem to share a common objective: to average or smooth the representation of the residual function on the coarse grid. If direct injection is used, and the value of the residual at the shared grid point is significantly higher or lower than the residuals at the immediately surrounding grid points, then the shared grid point will not represent the other points very well on the coarse grid. The weighting assures that the residual information at the fine grid points which are dropped in the transfer will be represented on the coarse grid along

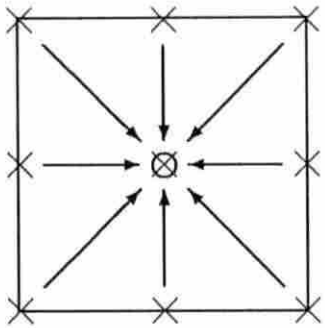

Figure 2.3: Restriction by weighting

with the residuals calculated at the shared grid points.

Two weighting strategies using a full 9-point stencil will be used in this thesis. The first is an area weighting scheme based on a uniform Cartesian grid. Figure 2.4 shows a sample computational cell surrounding a shared grid point subdivided into several compartments. Each compartment is associated with a point. In order to compute an average for the entire cell, the residuals at the points are averaged by weighting each of them with the relative area of its associated compartment as labeled on the figure.

Alternatively in the literature, this is often represented by the matrix shown below where each array element is the weighting of a point with a location in Figure 2.4 corresponding to the location of the array element.

$$I_h^H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The optional fraction in front of the matrix denotes that the number 16 is the weight of the entire averaging. This restriction will be referred to throughout the rest of the

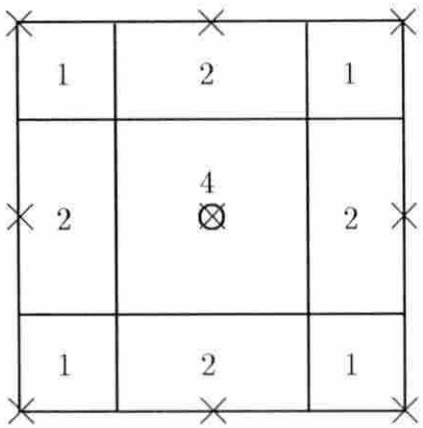

Figure 2.4: The 1/16 restriction

thesis as "1/16 averaging" or "1/16 restriction". This restriction operator works very well even for grids which aren't uniform or Cartesian as will be shown in the next chapter.

The second strategy is a weighting method which is based upon a similar idea, though it is applicable to a generalized grid. Instead of assuming the relative area of each compartment, they are computed directly. This weighting has an accuracy similar to that of the 1/16 operator when a relatively uniform mesh is used, yet adapts to areas where the mesh spacing varies. This weighting will be known as "area weighted" restriction and is shown below.

$$I_h^H = \begin{bmatrix} a_{i-1,j+1} & a_{i,j+1} & a_{i+1,j+1} \\ a_{i-1,j} & a_{i,j} & a_{i+1,j} \\ a_{i-1,j-1} & a_{i,j-1} & a_{i+1,j-1} \end{bmatrix}$$

The a's denote area while the indices are used to emphasize that each area is evaluated at the associated subdivision. Note that for a mesh with uniform cell areas, the operator becomes the 1/16 operator.

2.4 Prolongation

After the coarse grid problem has converged or some other decision has been made to move the problem back to the fine grid, the solution on the fine grid needs to be updated with the information calculated on the coarse grid. This is known as prolongation. The prolongation method can be categorized in terms of three different types of transfers depending upon the relationship between a fine grid point and the points with locations in common between the fine and coarse grids. These three categories are illustrated in Figure 2.5 and will be explained one by one.

The first type of restriction is the most straightforward of the three: it is restriction by direct injection in the opposite direction. The correction solved for at a coarse grid point is the correction applied to the fine grid point which shares the same location (Figure 2.6).

The second type of restriction involves the fine grid points that lie along a grid line directly between two points which are shared with the coarse grid as illustrated in Figure 2.7. These points are prolongated by either averaging the corrections at the two points or by interpolating between them.

The process of generating the coarse grid dropped every other grid line. Intersections of grid lines which were dropped represent points which lie inside boxes whose corners were selected to be coarse grid points (Figure 2.8). These points inside of the boxes are the points which the first two prolongation categories missed. The

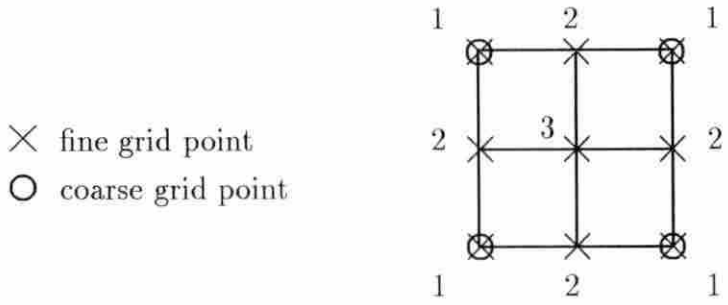

Figure 2.5: The set of points for the three types of prolongation

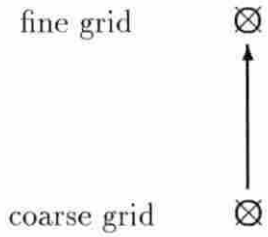

Figure 2.6: Direct prolongation

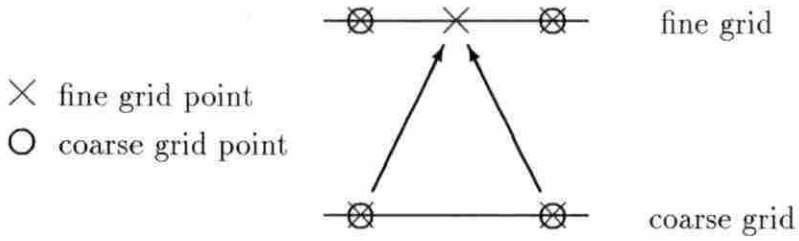

Figure 2.7: Prolongation by linear interpolation

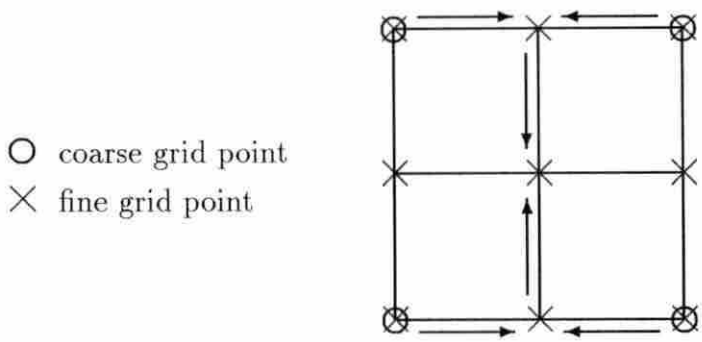

Figure 2.8: Prolongation by bilinear interpolation

corrections to the solution variables at these points are found either by averaging the corrections at the four corners or by bilinear interpolation. When bilinear interpolation is used, the correction for the points are interpolated linearly from the corrections that were already calculated by linear interpolation for the centers of two opposite sides (Figure 2.8).

For consistency, the second and third categories will both use either simple averaging or interpolation. Using interpolation should be more accurate, producing quicker convergence except when applied on a grid with uniform spacing. In this case, the interpolation becomes simple averaging. However, simple averaging does work very well over a wide range of stretched grids as will be seen in the next chapter. This author believes that this is because any errors in prolongation are well suited for being smoothed quickly due to the local nature of each error. Thus, except for rapidly varying solutions or highly stretched grids, it may not matter which method of prolongation is used.

2.5 Cycling Strategy

Everything needed has been introduced about the implementation of multigrid except when it is appropriate to move from one grid to another. Such a wide variety of strategies are possible, that discussing all of them could not be accomplished in a few pages. However, some thoughts on the subject will be mentioned. The fine grid should be iterated upon as little as possible in comparison to the coarse grid. After updating the solution on the fine grid, it is generally desirable to spend a small number of iterations (10 or less) to remove any new high frequency components of the error function which may have been produced through the multigrid cycling process. In general, the more efficient the equation solver, the fewer the number of iterations needed on the fine grid each cycle.

Several cycling strategies were tried with varying degrees of success. One, which was simple, performed very well compared to all of the others. This thesis will deal only with that particular strategy which is known as the V-cycle.

This description of the V-cycle comes from Brandt [4]. The V-cycle, usually denoted as $V(\nu_1, \nu_2)$, starts on the finest grid with ν_1 iterations and then transfers the problem to the next coarser grid. This repeats until the coarsest grid is reached. On the coarsest grid, the equations are usually either solved completely or $\nu_1 + \nu_2$ iterations are performed. The problem then transfers back through progressively finer grids with ν_2 iterations being made on each grid including the finest grid giving a total of $\nu_1 + \nu_2$ iterations on the finest grid for each cycle. This process repeats itself until the fine grid problem is solved. The V-cycle receives its name from its zig-zag motion of switching from grid to grid (Figure 2.9). Solving the coarsest grid equation completely was used in Chapter 2 since it was found to produce solutions

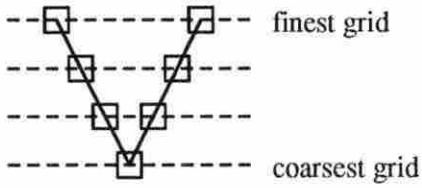

Figure 2.9: The V-cycle

in fewer equivalent fine grid iterations for the sample problem solved with Laplace's equation.

This concludes the introduction to multigrid methods. The next chapter will focus on applying multigrid methods to the two-dimensional Laplace's equation.

3. LAPLACE'S EQUATION

3.1 Formulation

Laplace's equation, also known as the potential equation, can be used to describe steady-state heat diffusion in a solid assumed to have constant thermal conductivity and no heat generation and will be used as a model equation to demonstrate the basic behavior of a multigrid algorithm before moving on to the more complicated Navier-Stokes equations in the next chapter. In this section, a mathematical formulation of Laplace's equation convenient for obtaining a solution in a general physical domain will be derived.

3.1.1 Physical Coordinates

The heat diffusion equation for a two-dimensional, isotropic solid in Cartesian coordinates is shown below:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\dot{q}=\rho c_v\frac{\partial T}{\partial t} \tag{3.1}$$

It is to be solved for the temperature field $T(x,y,t)$. The physical properties ρ, density; c_v, specific heat at constant volume; and k, thermal conductivity may all be functions of temperature. The heat generation $\dot{q}$ can be a function of space, time,

and even temperature. For constant k, equation (3.1) takes the following form

$$\alpha\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)+\frac{\dot{q}}{\rho c_v}=\frac{\partial T}{\partial t}$$

where $\alpha = k/\rho c_v$ is the thermal diffusivity. If no heat generation exists and only the steady-state condition is of interest then the equation becomes Laplace's equation:

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0 \tag{3.2}$$

This equation is an elliptic partial differential equation whose solution has been determined analytically for some geometries and boundary conditions using Fourier series. (A discussion of these can be found in advanced heat transfer texts.) Due to its simplicity and applicability to a wide variety of problems, Laplace's equation was chosen for developing and debugging a simple multigrid algorithm.

3.1.2 Transformed Coordinates

In this study, a generalized coordinate transformation will be employed. If the grid is stationary, then the following generalized coordinate transformation can be used:

$$\xi=\xi(x,y),\quad \eta=\eta(x,y),\quad \zeta=z$$

The conservative form of the transformed Laplacian operator is given by

$$\nabla^2 F=\frac{1}{\sqrt{g}}\sum_{m=1}^{3}\sum_{n=1}^{3}\left(\sqrt{g}\,g^{mn}F_{\xi^n}\right)_{\xi^m} \tag{3.3}$$

where

$$\sqrt{g}=\mathbf{a}_1\cdot(\mathbf{a}_2\times\mathbf{a}_3)$$

$$\mathbf{a}_n=\mathbf{r}_{\xi^n},\quad n=1,2,3$$

$$g^{mn} = \nabla\xi^m \cdot \nabla\xi^n; \quad m = 1,2,3, \quad n = 1,2,3$$

Some explanation of the terms in the above equation may be needed. The term $\mathbf{r}$ is a position vector of a node using the original Cartesian coordinate system while $\mathbf{r}_{\xi^n}$ is the derivative of $\mathbf{r}$ with respect to the transformed coordinates ($\xi^1 = \xi$, $\xi^2 = \eta$, and $\xi^3 = \zeta$ for the problem at hand). The term $\sqrt{g}$ represents J, the Jacobian of the transformation. Finally, $\nabla\xi^l$ is the divergence of the position vector in the transformed coordinates with respect to the original coordinates and can be calculated from the following formula:

$$\nabla\xi^l = \frac{\mathbf{a}_m \times \mathbf{a}_n}{\sqrt{g}}; \quad l = 1,2,3, \quad l,m,n \ \ cyclic$$

Evaluating the transformation equation (3.3) with temperature T, produces the following:

$$\left[\left(\frac{x_\eta^2 + y_\eta^2}{J}T_\xi\right) + \left(\frac{-y_\eta y_\xi - x_\eta x_\xi}{J}T_\eta\right)\right]_\xi + \left[\left(\frac{-y_\eta y_\xi - x_\eta x_\xi}{J}T_\xi\right) + \left(\frac{x_\xi^2 + y_\xi^2}{J}T_\eta\right)\right]_\eta = 0 \tag{3.4}$$

The application of the coordinate transformation described above has resulted in an equation in conservative form which will be discretized in a later section.

3.2 Numerical Solution Algorithm

In this section, the numerical solution algorithm will be described. The discussion will cover the discretization method, the grid generation scheme, metric term evaluation, the use of Gauss-Seidel iteration and the modified strongly implicit procedure, boundary conditions, convergence criterion, and the solution procedure. These topics will be discussed one by one.

3.2.1 Discretization Method

The governing equation written in a generalized coordinate system for two-dimensional heat conduction, equation (3.4), contains two types of terms: spatial second derivative terms and mixed spatial second derivative terms. The discretization method for each type of the derivative terms is described below except for the metric terms which will be discussed in a later section. Note that these have been simplified by $\Delta\xi = \Delta\eta = 1$.

- *Spatial second derivative terms*

  These terms were differenced similar to the following second-order central difference formulas:

$$\frac{\partial^2 T}{\partial \xi^2} = T_{i+1} - 2T + T_{i-1}$$

$$\frac{\partial^2 T}{\partial \eta^2} = T_{j+1} - 2T + T_{j-1}$$

  However, the difference method incorporated information at "half" nodes due to the a's which represent combinations of metric terms:

$$\begin{aligned} \frac{\partial}{\partial \xi}\left(a\frac{\partial T}{\partial \xi}\right) &= \left(a\frac{\partial T}{\partial \xi}\right)_{i+1/2} - \left(a\frac{\partial T}{\partial \xi}\right)_{i-1/2} \\ &= a_{i+1/2}\left(\frac{\partial T}{\partial \xi}\right)_{i+1/2} - a_{i-1/2}\left(\frac{\partial T}{\partial \xi}\right)_{i-1/2} \end{aligned}$$

  The first derivative terms at the half nodal points were evaluated in the following fashion:

$$\frac{\partial T}{\partial \xi}_{i+1/2} = T_{i+1} - T_i$$

$$\frac{\partial T}{\partial \xi}_{i-1/2} = T_i - T_{i-1}$$

  Expressions for the terms in the η direction were evaluated in a similar fashion.

- *Mixed spatial second derivative terms*

  These terms were differenced in a manner similar to the following second-order central difference formula:

$$\frac{\partial}{\partial \xi}\left(\frac{\partial T}{\partial \eta}\right) = \frac{1}{4}\left(T_{i+1,j+1} - T_{i+1,j-1} - T_{i-1,j+1} - T_{i-1,j-1}\right)$$

  First, the outside derivative was discretized:

$$\frac{\partial}{\partial \xi}\left(a\frac{\partial T}{\partial \eta}\right) = \frac{1}{2}\left[\left(a\frac{\partial T}{\partial \eta}\right)_{i+1} - \left(a\frac{\partial T}{\partial \eta}\right)_{i-1}\right]$$

  Then the inside derivative was discretized:

$$= \frac{1}{4}\left[a_{i+1}\left(T_{i+1,j+1} - T_{i+1,j-1}\right) - a_{i-1}\left(T_{i-1,j+1} - T_{i-1,j-1}\right)\right]$$

$$= \frac{1}{4}\left[a_{i+1}T_{i+1,j+1} - a_{i+1}T_{i+1,j-1} - a_{i-1}T_{i-1,j+1} + a_{i-1}T_{i-1,j-1}\right]$$

  Note the similarity between the final expression and the first expression. If the physical and computational domains are the same, then the final expression becomes the first expression. The difference expression for the other derivative where ξ and η are reversed was evaluated in a similar way.

3.2.2 Grid Generation Scheme

The only geometry examined with Laplace's equation in this thesis is a one radian wide "wedge" cross-section of a hollow pipe. Alternatively, this can be thought of as an area in polar coordinates (r, θ) bounded by $r = r_1, r_2$ and $\theta = \theta_1, \theta_2$ where $\theta_2 = \theta_1 + 1$ radian. The first method whereby the mesh was generated was with grid lines that were equally spaced with constant radius or constant angle and were therefore simple to produce as can be seen by the sample grid in Figure 3.1.

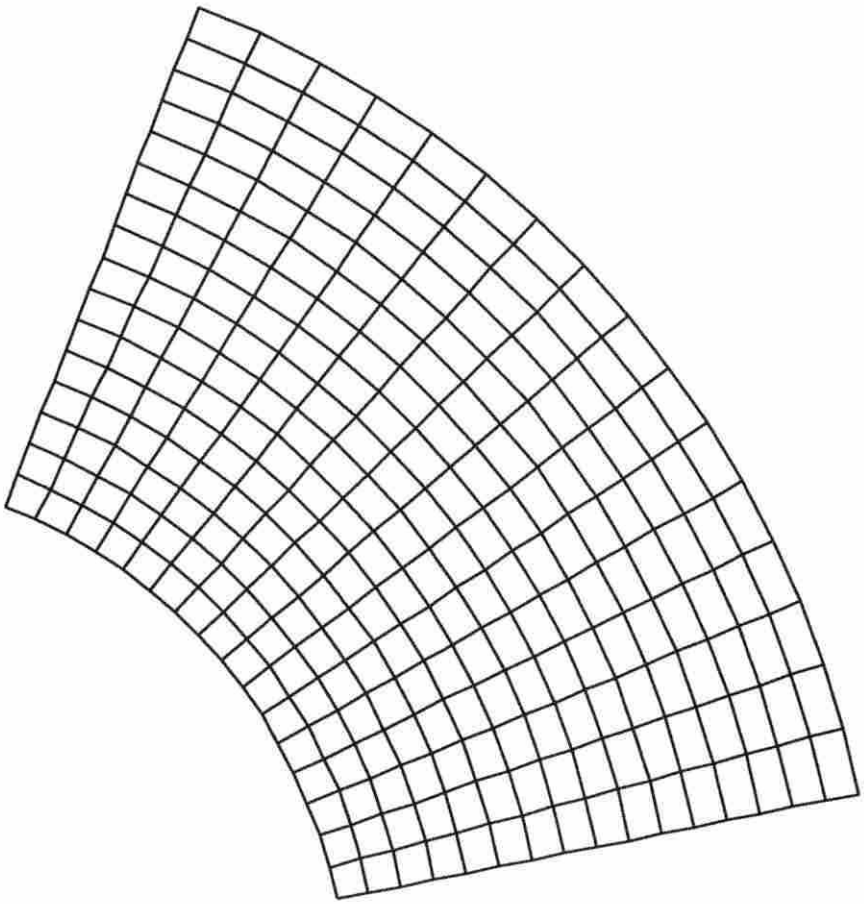

Figure 3.1: A sample 17x17 uniform grid

Additionally, many other grids were used with cell areas that varied with radius. The rate at which the cell areas vary will be referred to as the "cell area ratio". A cell area ratio of 1.0 refers to a grid composed entirely of cells with equal areas as shown in Figure 3.2. On the other hand, a cell area ratio of any other value, say 1.15, refers to a grid where each row of cells along the same radius has an area that is 1.15 times the area of the previous row of cells as one moves with increasing radius. In this way, multigrid performance can be examined on a variety of stretched grids. Several figures showing grids generated with the range of cell area ratios used in the results can be found in Appendix B.

3.2.3 Metric Term Evaluation

The transformed coordinates for the discretization were chosen conveniently as the polar coordinates, $\xi = r$ and $\eta = \theta$. Therefore:

$$\begin{aligned} r = r_1 &\Rightarrow \xi = 0 \\ r = r_2 &\Rightarrow \xi = \xi_{max} \\ \theta = \theta_1 &\Rightarrow \eta = 0 \\ \theta = \theta_2 &\Rightarrow \eta = \eta_{max} \end{aligned}$$

Laplace's equation could have been transformed directly to cylindrical coordinates producing analytical expressions for the metric terms. However, a generalized development allows for easier changes to alternate geometries, if so desired.

All metric terms were evaluated by a second-order central difference formula in accordance with the discretization scheme described in a previous section. For

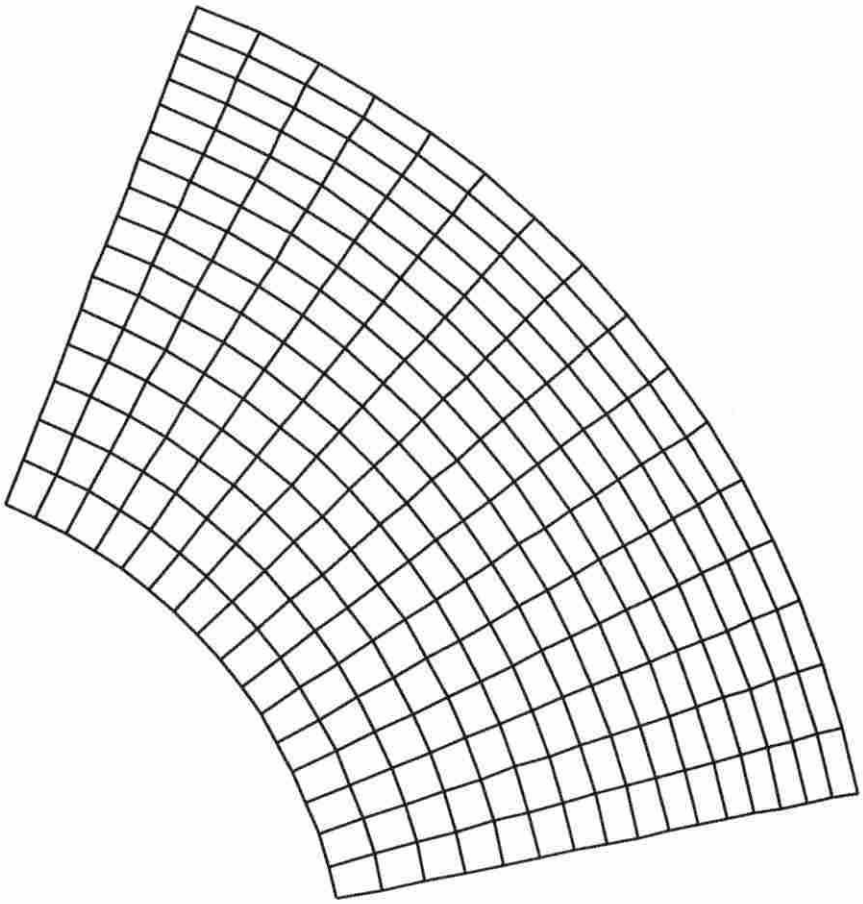

Figure 3.2: A sample 17x17 grid with equal cell areas

example x_ξ at node i, j was evaluated with the expression

$$\frac{\partial x}{\partial \xi} = \frac{x_{i+1/2} - x_{i-1/2}}{\xi_{i+1/2} - \xi_{i-1/2}} \tag{3.5}$$

Note that the expression above requires information at "half" node locations. The evaluation of these terms was simplified by the grid generation scheme. A grid was generated which was actually twice as "dense" as the grid used to model the physical domain in order to evaluate these terms. However, metric terms evaluated at half node points did not require information at the half nodes. For example, x_ξ at node $i + 1/2, j$ was evaluated with the expression

$$\frac{\partial x}{\partial \xi}_{i+1/2} = \frac{x_{i+1} - x_i}{\xi_{i+1} - \xi_i} \tag{3.6}$$

For the symmetric boundaries (discussed in a later section), the interior equations were solved on the boundaries and thus required the evaluation of metric terms on these boundaries. Also due to the simplicity of the grid generation scheme, it was quite convenient to generate a grid with additional points outside the actual physical domain and calculate the metric terms on the boundaries of the physical domain in the same fashion as the interior metric terms.

3.2.4 Gauss-Seidel Iteration

This section will describe how the Gauss-Seidel (GS) iteration scheme was applied to the system of algebraic equations. Although Gauss-Seidel iteration was not used to solve the Navier-Stokes equations in this study, using the procedure with Laplace's equation provided an additional check that the multigrid solver was operating properly.

The well-known concept behind Gauss-Seidel iteration is to solve each equation one at a time for a single unknown where newly computed values are used as soon as they are determined during an iteration. Thus, what was needed was to rearrange each equation in terms of a single unknown variable with no two equations being solved for the same variable. Each equation was discretized about a single point which will commonly be referred to as i, j. The unknown temperature at this point was solved for in each equation which provided the best arrangement for diagonal dominance.

After the governing equation had been discretized, coefficients existed for the temperature at nine nodes in a stencil including and surrounding the i, j node. These coefficients will be labeled here and throughout the thesis as A^1, A^2, A^3, ... , A^9 as shown in Figure 3.3. Expressions for these coefficients can be found in Appendix A.

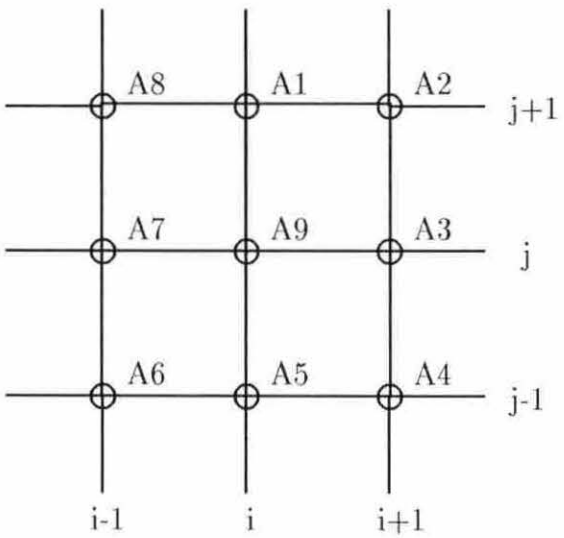

Figure 3.3: Computational cell

The governing equation can be written at node i, j as

$$\begin{aligned} A^1 T_{j+1} + A^2 T_{i+1,j+1} + A^3 T_{i+1} + A^4 T_{i+1,j-1} + A^5 T_{j-1} \\ + A^6 T_{i-1,j-1} + A^7 T_{i-1} + A^8 T_{i-1,j+1} + A^9 T_{i,j} &= 0 \end{aligned} \quad (3.7)$$

The terms with coefficients A^1 through A^8 were moved to the right-hand side and the equation divided by A^9 to produce

$$\begin{aligned} T_{i,j} = &-\left\{A^1 T_{j+1} + A^2 T_{i+1,j+1} + A^3 T_{i+1} + A^4 T_{i+1,j-1} \right. \\ &\left. + A^5 T_{j-1} + A^6 T_{i-1,j-1} + A^7 T_{i-1} + A^8 T_{i-1,j+1}\right\} / A^9 \end{aligned} \quad (3.8)$$

which is in the desired solution form.

3.2.5 Modified Strongly Implicit Procedure

This section will describe the application of a five-point version of the modified strongly implicit procedure (which will be referred to as just the strongly implicit procedure or SIP in this thesis though it has been referred to as MSIP in the literature) to the governing equation. The SIP solver is an incomplete LU method which will not be explained in detail here, but a description of the original strongly implicit procedure can be found by Stone [7], which later became the modified strongly implicit procedure by Schneider and Zedan [8], and can also be found in Anderson et al. [9]

The discretized equation was rewritten into delta form. The previously discretized equation (3.7) can be summarized in the following form:

$$\mathbf{AT} = 0$$

where $\mathbf{T}$ is a column vector of the temperatures at the grid points and $\mathbf{A}$ is a matrix of the coefficients of $\mathbf{T}$. For each node there are nine, generally nonzero, coefficients on the left-hand side of the equation. Since a five-point version of the strongly implicit procedure was used, only five of the nine unknowns (those with coefficients A^1, A^3, A^5, A^7, and A^9) were treated implicitly in each scalar equation. The rest were lagged to the right-hand side. In order to write the equation in delta form, $T^n + \Delta T^{n+1}$ was substituted for T at the five points treated implicitly while T^n was substituted for T at the four points treated explicitly where n refers to the iteration level. This gives

$$\mathbf{A}_e\mathbf{T}^n + \mathbf{A}_i\left(\mathbf{T}^n + \Delta\mathbf{T}^{n+1}\right) = 0$$

which can be rearranged to form:

$$\mathbf{A}_i\Delta\mathbf{T}^{n+1} = -\mathbf{A}\mathbf{T}^n$$

where

$$\mathbf{A} = \mathbf{A}_e + \mathbf{A}_i$$

The solution was then updated by computing $\mathbf{T}^{n+1} = \Delta\mathbf{T}^{n+1} + \mathbf{T}^n$ after each iteration. In this fashion the four lagged points were treated explicitly, while the other five were treated implicitly at each node.

Of course, the four lagged points are treated implicitly for scalar equations written at some of the other nodes. Also, the coefficients for the explicit points are those of the cross derivatives which disappear for an orthogonal grid which is the case for the results to be presented. Since these cross-derivative terms were still being computed numerically, they may have had some nonzero magnitude, but they would have been very small in comparison to the other coefficients. Therefore, despite the

treatment of these four points at each node, the solution procedure was still strongly implicit.

3.2.6 Boundary Conditions

This section describes the boundary conditions which were used when Laplace's equation was being solved. For this study, only Dirichlet and symmetric boundaries were considered. The boundary conditions are described as follows:

- *Dirichlet*

  When Gauss-Seidel iteration was used as the solver, T was simply fixed at the boundary. However, SIP solved equations written in delta form. Therefore, when SIP was used, T was fixed at the boundary and the boundary equation was written as $\Delta T = 0$.

- *Symmetric*

  The example problem to be discussed in this chapter involves a one-radian wide cross-section of a hollow cylinder with constant temperature on each of the inside and outside radii. This problem serves as a model problem, and an analytical form of the solution can be found in several heat transfer texts. Before solving the problem, a boundary condition needed to be selected for the constant angle boundaries of the domain. An important thing to note about the analytical solution to this problem is that it varies only with radius. This fits the description of symmetry for a constant-angle boundary in that the solution can be reflected about any line of constant angle and correctly reproduce the solution. Thus, this type of boundary was selected for implementation on the constant-angle boundaries of the domain. The interior equation was solved on

the boundary, but the equation was rewritten such that the temperatures at the row of points outside the domain were set equal to the temperatures of the first interior row of points, thus, retaining the implicit character of the scheme.

3.2.7 Convergence Criterion

For the calculations in this chapter, the convergence criterion was based upon a maximum nondimensional change in temperature. This criterion is as follows:

$$\frac{max\,(\Delta T)}{T_{max}} \leq \epsilon = 10^{-10}$$

where $max\,(\Delta T)$ is the maximum change in temperature computed at an iteration, and T_{max} is the maximum temperature along the boundary (which is also the maximum temperature at any node).

3.2.8 Solution Procedure

The solution procedure for solving Laplace's equation without multigrid can be summarized as follows:

1. Set the initial guess.
2. Generate the mesh and metric terms.
3. Construct the left-hand coefficient matrix **A**.
4. Construct the right-hand side vector. (SIP only)
5. Call the Gauss-Seidel or the SIP solver and update the solution, $T\,(x, y)$.

6. Go to step 3 and repeat until either the steady-state solution or the specified maximum number of iterations is reached.

After adding multigrid, the solution procedure becomes the following:

1. Set the initial guess on the finest grid.
2. Generate the meshes and metric terms for all of the grids.
3. Start on the finest grid.
4. Perform ν_1 iterations. If this is the finest grid and the solution has converged, then stop.
5. Transfer the problem to the next coarser grid: Restrict the residual, and set an initial guess of $\Delta T = 0$ for the correction.
6. If this is the coarsest grid, solve completely. Otherwise go to step 4.
7. Prolongate the corrections to the next finer grid.
8. Perform ν_2 iterations. If this is the finest grid and the solution has converged, then stop. If this is the finest grid and the solution has not converged then go to step 4. Otherwise, return to step 7.

It should be noted that each iteration in the multigrid solution procedure is composed of steps 3 through 5 of the single grid solution procedure.

3.3 Multigrid

When multigrid was used, Laplace's equation was solved using the correction scheme in conjunction with the V-cycle strategy which were both presented in the

previous chapter. Unless otherwise stated in the results, this section outlines the basic details of the application of the multigrid scheme.

When restricting, residuals at interior points were calculated using the 1/16 restriction operator described in the previous chapter while the residuals on the boundaries were computed using direct injection. This was done to maintain the boundary conditions. For example, at a Dirichlet boundary, the residual is always zero, because the boundary condition is always satisfied since the temperature is known there. Using a weighted average for the residual at these boundaries will produce a nonzero residual for these points on the coarse grid. If this happens, the solution to the coarse problem will produce nonzero corrections to these boundary points; thus, destroying the boundary. The prolongation operator used direct prolongation and simple averaging which was also discussed in the previous chapter.

3.4 Results

An application using multigrid methods with Laplace's equation is presented and discussed in this section. The problem solved is a one-radian wide cross-section of a hollow cylinder presented earlier. The radius varies from one to two units. The temperature on the inside radius was fixed at 50 while the temperature on the outside radius was 100. The initial temperature for all other points was 75.

The cycling strategy used was the V-cycle with, perhaps unusual, settings of $\nu_1 = 0$, $\nu_2 = 3$ when Gauss-Seidel iteration was used as the smoother and $\nu_1 = 0$, $\nu_2 = 1$ when SIP was used as the smoother. This means that the problem was transferred to the coarsest grid immediately at the very beginning of the cycle. The coarsest grid problem was solved completely, and ν_2 iterations were made on each of

the successively finer grids including the finest grid. These settings for the parameters were found to be nearly optimal when four levels were used. When fewer levels were used, parameters with higher values produced better results. However, the purpose of these results is to illustrate the general characteristics of multigrid performance rather than what multigrid can do with a high degree of optimization.

The first calculations made were those comparing the effects of varying the number of grids (1 to 4) and the sizes of a uniform grid (17x17, 33x33, 65x65, and 129x129). Since each grid was composed of approximately one-fourth as many points as the previously finer grid, an iteration on each grid was considered to be worth one-fourth of the computational effort of an iteration on the previously finer grid. In that way, the computational effort of an iteration on each coarser grid was considered to be worth 1/4, 1/16/, and 1/64 of the computational effort of an iteration on the finest grid respectively. In the same fashion, a term will be defined for expressing computational effort called a "work unit" (wu) or an "equivalent fine grid iteration". An iteration on the finest grid is worth one work unit of computational effort while iterations on each coarser grid is 1/4, 1/16, and 1/64 of a work unit in computational effort respectively. Tables 3.1 and 3.2 with accompanying figures show the total number of work units needed to reach convergence using SIP and Gauss-Seidel iteration respectively. Tables 3.3 and 3.4 look at the ratio of total work units used without using multigrid compared to those using multigrid. These were calculated directly from the data used to created the two previous tables.

Three trends can be seen from Tables 3.3 and 3.4. First of all, the factor that multigrid improves convergence by improves with increasing number of levels. This is to be expected since each time a level is added, information can propagate roughly

Table 3.1: SIP equivalent fine grid iterations using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
1	92.00	324.00	1168.00	4220.00
2	40.75	97.75	276.50	834.25
3	15.81	20.38	34.75	80.25
4	13.41	13.95	15.19	18.98

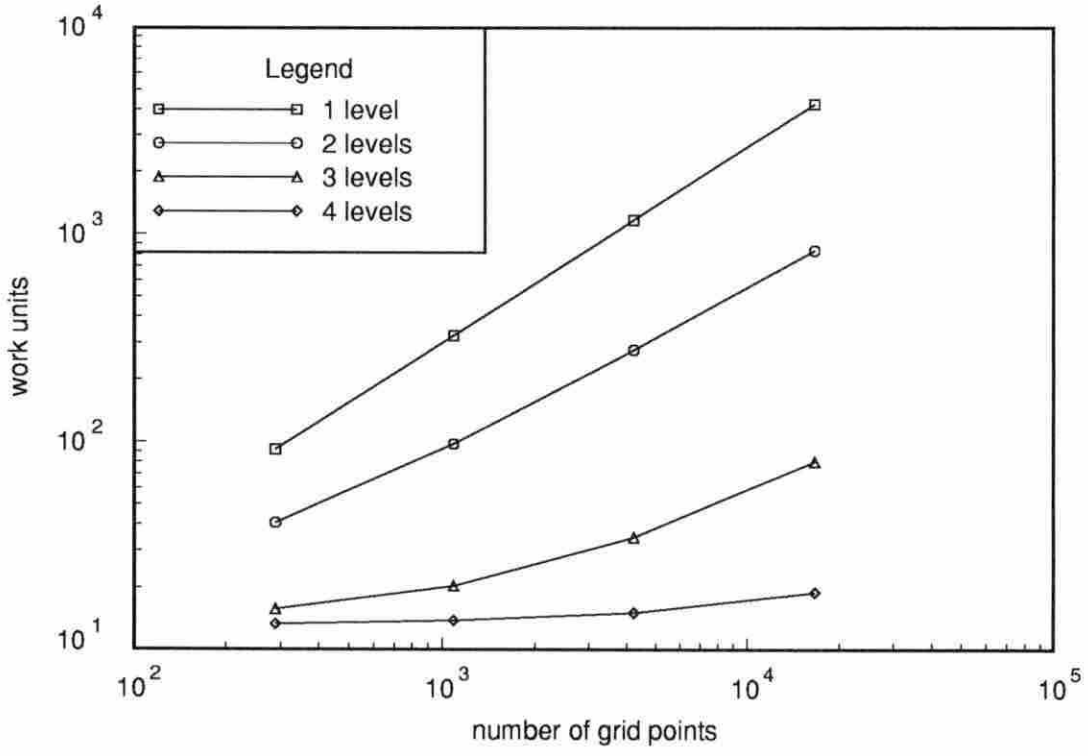

Figure 3.4: SIP equivalent fine grid iterations using uniform grids

Table 3.2: Gauss-Seidel iteration equivalent fine grid iterations using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
1	646.00	2327.00	8343.00	29690.00
2	175.50	532.50	1635.25	5030.00
3	46.50	80.00	178.81	495.06
4	35.42	41.02	49.38	77.53

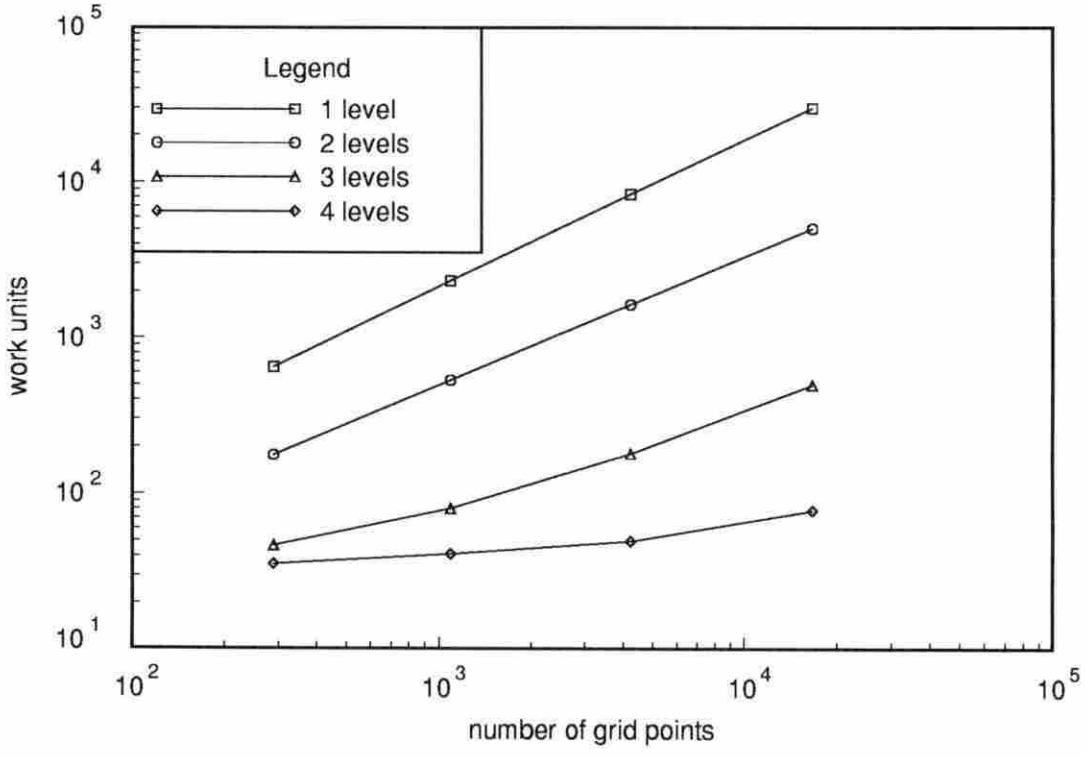

Figure 3.5: Gauss-Seidel iteration equivalent fine grid iterations using uniform grids

Table 3.3: SIP work unit improvement using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
2	2.26	3.31	4.22	5.06
3	5.82	15.90	33.61	52.59
4	6.86	23.22	76.91	222.29

Table 3.4: Gauss-Seidel iteration work unit improvement using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
2	3.68	4.37	5.10	5.90
3	13.89	29.09	46.66	59.97
4	18.24	56.73	168.97	382.94

a factor of 2 faster on the coarsest grid.

Secondly, the factor of improvement increases with increasing grid size. This is due to the total number of points which information needs to travel through to reach the center of the domain. The fewer the number of points from the boundaries to the center, the less that the rate of convergence can be improved. Combining these first two observations, as the number of levels increases, the number of equivalent fine grid iterations becomes less dependent upon the number of grid points.

Finally, Gauss-Seidel is improved more by multigrid than the strongly implicit procedure is. This is because SIP is a more efficient solver in that all equations are solved simultaneously over the entire domain rather than one at a time with each being influenced by a few local points. Similar results when comparing multigrid performance with different equation solvers were found by Miller and Schmidt [10].

Examining the number of work units provides only an estimate of the amount by which a code is improved. A more desirable rating of improvement may be the

cpu time expended to solve the cases presented. Tables 3.5 and 3.6 along with their accompanying plots list the number of cpu seconds used to solve the same cases presented in Tables 3.1 and 3.2 while Tables 3.7 and 3.8 show the ratio of improvement in cpu time of the multigrid solution procedure over the single grid case. The code which performed these cases is written in FORTRAN and was compiled with the flags +es -R8 -K +OP3 on an HP9000/730 workstation running HP-UX 8.07. In all cases, the program was compiled with array dimensions of the maximum case solved (129x129) and required some minor corrections and recompiling in order to change the number of levels. A handful of cases were run more than once and the cpu time for these cases showed a variation of about +/- 5% of their average value. Some variation is to be expected since workstations are capable of multitasking which affects cache memory, and this particular machine was often shared simultaneously by other researchers and other tasks performed by the author.

Interestingly, as the number of levels increased, the advantage of using SIP over Gauss-Seidel iteration in terms of cpu time expended drastically decreased and nearly disappeared. This would seem to indicate that using Gauss-Seidel iteration may be more or just as attractive to use as a smoother compared to SIP when four or more levels are used when solving Laplace's equation. This especially true when taking into account the simplicity and the low amount of memory required of by Gauss-Seidel iteration compared to SIP.

Additional cases were performed to examine the effects of grid stretching along with restriction and prolongation operators. Tables 3.9 through 3.12 (each with an accompanying plot) show the number of work units required to converge on a 17x17 grid when the prolongation operator, restriction operator, and grid are varied. All

Table 3.5: SIP cpu seconds using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
1	0.26	1.97	26.25	399.76
2	0.27	1.37	9.62	94.40
3	0.20	0.85	4.48	22.00
4	0.22	0.82	4.07	17.28

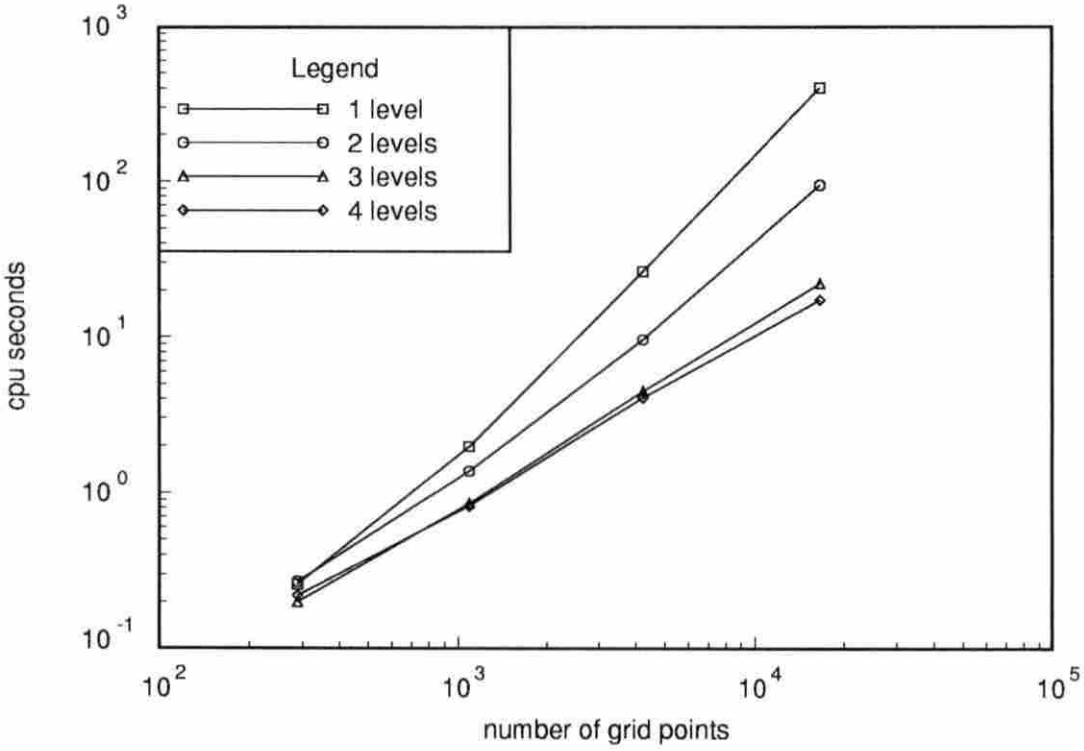

Figure 3.6: SIP cpu time using uniform grids

Table 3.6: Gauss-Seidel iteration cpu seconds using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
1	0.80	9.61	176.44	2499.29
2	0.47	3.14	35.87	445.60
3	0.26	1.23	7.28	55.58
4	0.27	0.92	4.79	21.21

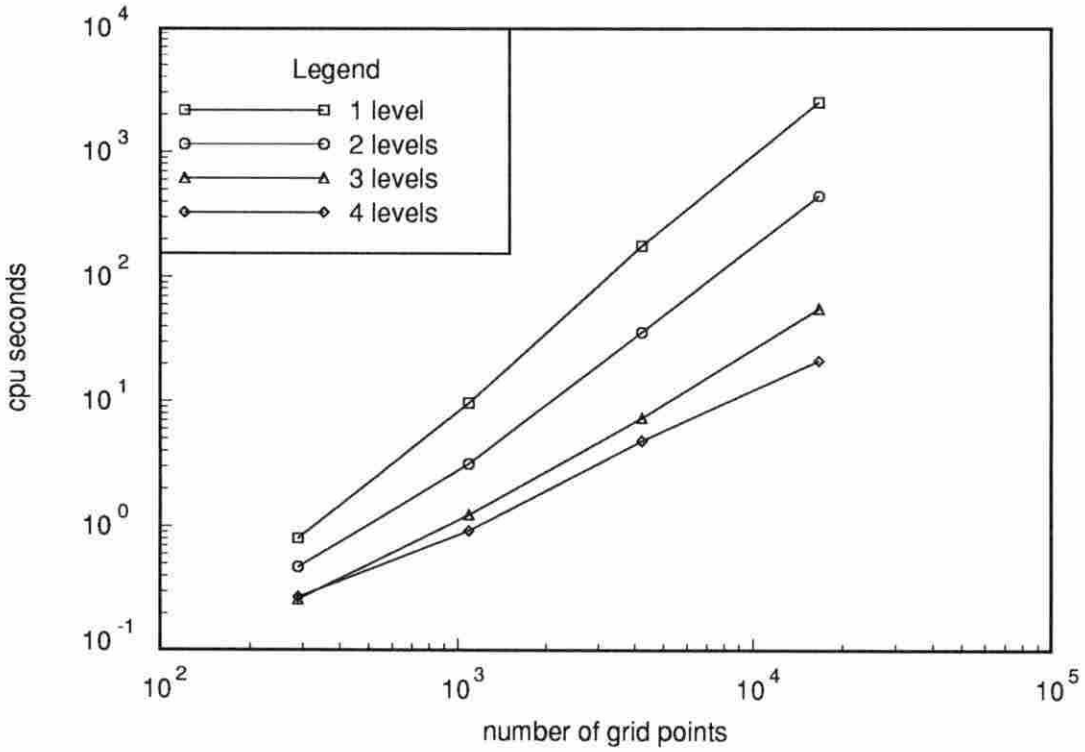

Figure 3.7: Gauss-Seidel iteration cpu time using uniform grids

Table 3.7: SIP cpu time improvement using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
2	0.96	1.44	2.73	4.23
3	1.30	2.32	5.86	18.17
4	1.18	2.40	6.45	23.13

Table 3.8: Gauss-Seidel iteration cpu time improvement using uniform grids

# levels	17 x 17	33 x 33	65 x 65	129 x 129
2	1.70	3.06	4.92	5.61
3	3.08	7.81	24.24	44.97
4	2.96	10.45	36.84	117.84

other details of problem are the same as for the first set of results. The results shown in Tables 3.9 and 3.10 were generated with two levels while the results in Tables 3.11 and 3.12 were generated with four levels. For comparison purposes, Table 3.13 shows how the single grid problem performed as the mesh was stretched. The operators compared are 1/16 weighting restriction, area weighted restriction, simple averaging prolongation, and interpolation prolongation. All of which were discussed in the previous chapter.

The term "cell area ratio" in the tables refers to how the grid was generated. Each row of cells, moving away from the inside radius, were generated with an area which was the cell area ratio times the area of the previous row. In that way, a cell area ratio of 1.0 refers to a grid where every cell in the mesh has the same area. Also, a cell ratio of say 1.1 specifies that the cells in each row moving away from the inside radius each have an area 1.1 times the area of the cells in the previous row. Each grid shares the same inside boundary radius of one unit and the same outside

Table 3.9: Transfer operator comparison using SIP and 2 levels

cell area ratio	1/16 restriction		area restriction	
	simple averaging prolongation (work units)	interpolation prolongation (work units)	simple averaging prolongation (work units)	interpolation prolongation (work units)
0.80	36.00	34.75	33.25	32.25
0.85	40.75	38.75	37.25	36.75
0.90	45.00	45.75	44.25	42.50
0.95	48.50	48.25	48.75	48.50
0.96	49.50	49.50	49.00	49.25
0.97	48.25	48.50	48.00	48.50
0.98	48.25	48.25	48.00	48.25
0.99	47.75	47.75	47.75	47.75
1.00	45.50	45.25	45.50	45.25
1.01	44.25	44.25	45.25	45.00
1.02	41.75	41.75	43.50	43.25
1.03	36.50	37.00	41.50	41.50
1.04	39.50	39.50	39.00	39.00
1.05	40.00	40.00	34.25	34.00
1.10	35.75	35.00	34.50	34.50
1.15	30.00	29.50	29.00	29.00
1.20	26.50	26.00	25.50	25.00
1.25	22.00	23.00	22.25	22.25

boundary radius of two units while the cell area ratio was varied from 0.80 to 1.25 (see Appendix B to examine several of the actual grids used).

The figures accompanying the four tables are plotted as the ratio of work units to the work units needed to solve the 1/16 restriction used with simple averaging prolongation case. Therefore, the 1/16 restriction used with simple averaging prolongation case is plotted as a straight line at y = 1.00. Those cases performing better plotted below this line and vice versa.

The interpolation prolongation with the area weighted restriction generally per-

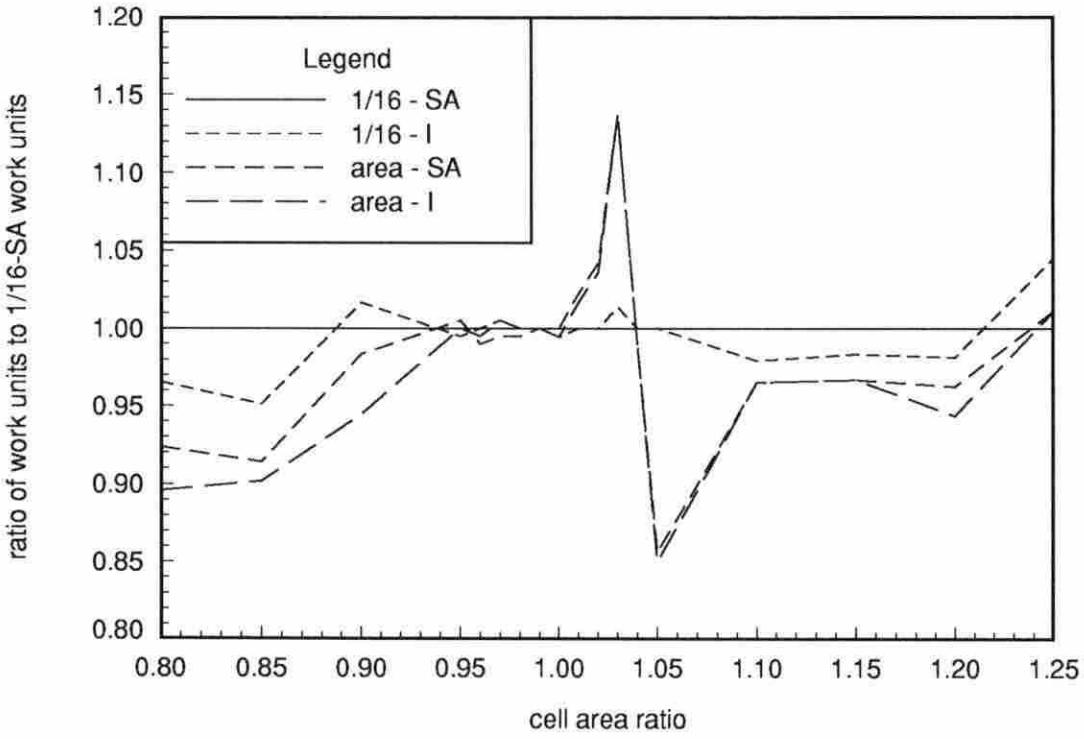

Figure 3.8: Transfer operator comparison using SIP and 2 levels

Table 3.10: Transfer operator comparison using GS iteration and 2 levels

cell area ratio	1/16 restriction		area restriction	
	simple averaging prolongation (work units)	interpolation prolongation (work units)	simple averaging prolongation (work units)	interpolation prolongation (work units)
0.80	477.50	434.25	436.50	402.25
0.85	526.50	485.50	486.00	448.50
0.90	508.75	477.00	479.75	452.75
0.95	401.25	387.50	387.50	375.50
0.96	372.00	361.00	360.25	350.75
0.97	341.25	332.00	332.50	325.00
0.98	310.25	304.75	303.50	298.25
0.99	280.00	276.00	277.50	273.50
1.00	251.75	249.75	251.75	249.75
1.01	225.00	223.25	228.75	228.00
1.02	195.75	195.75	209.50	209.50
1.03	177.00	177.00	194.00	193.75
1.04	190.75	190.75	179.00	177.00
1.05	194.00	193.50	156.75	156.75
1.10	214.00	209.75	181.00	182.75
1.15	260.75	254.50	222.25	221.00
1.20	303.25	292.75	264.75	260.25
1.25	343.25	327.00	302.50	293.25

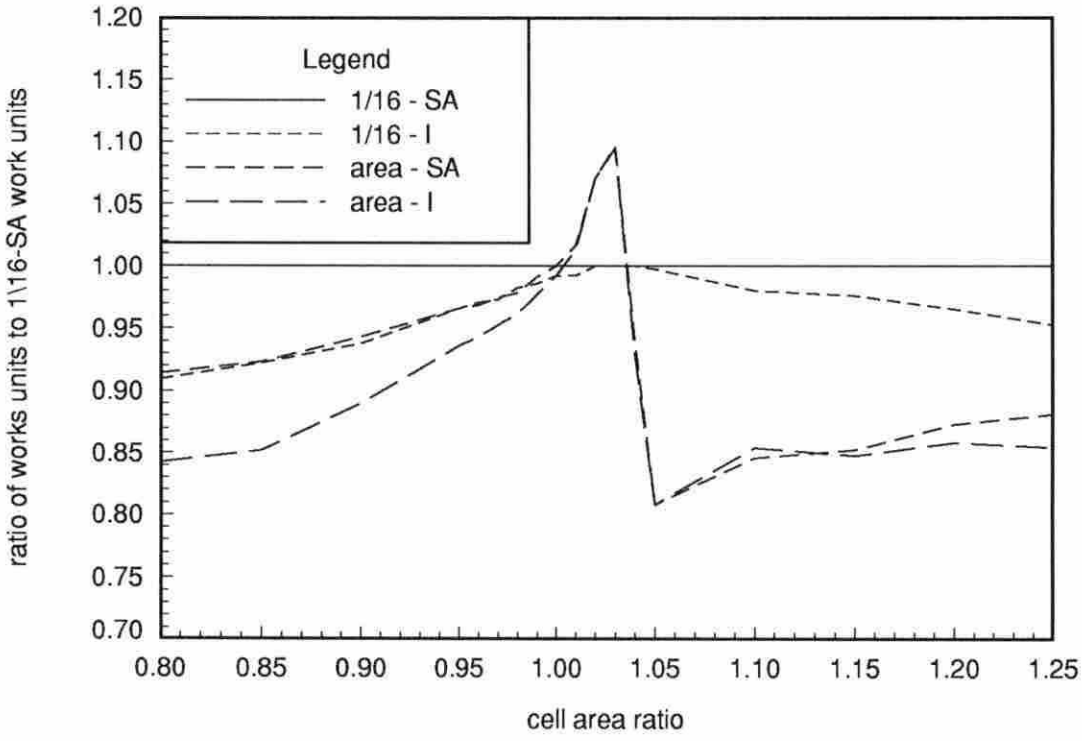

Figure 3.9: Transfer operator comparison using GS iteration and 2 levels

Table 3.11: Transfer operator comparison using SIP and 4 levels

cell area ratio	1/16 restriction		area restriction	
	simple averaging prolongation (work units)	interpolation prolongation (work units)	simple averaging prolongation (work units)	interpolation prolongation (work units)
0.80	17.4375	17.4219	14.7344	14.7500
0.85	18.7813	17.4375	16.0938	16.0938
0.90	18.7656	17.4375	17.4219	16.0938
0.95	16.0938	16.0938	16.0938	16.0938
0.96	16.0938	16.0938	16.0938	16.0938
0.97	14.7656	14.7656	14.7656	14.7656
0.98	14.7656	14.7500	14.7656	14.7500
0.99	14.7656	14.7500	14.7656	14.7500
1.00	13.4219	13.4219	13.4219	13.4219
1.01	13.4219	13.4219	13.4219	13.4219
1.02	13.4219	13.4219	13.4219	13.4219
1.03	13.4063	13.4063	13.4219	13.4219
1.04	13.4063	13.4063	13.4063	13.4063
1.05	13.4063	13.4063	13.4063	13.4063
1.10	13.4219	13.4063	13.4063	13.4063
1.15	13.4063	13.4063	13.4063	13.4063
1.20	13.4063	13.4063	13.4063	13.4063
1.25	12.0781	13.4063	13.4063	13.4063

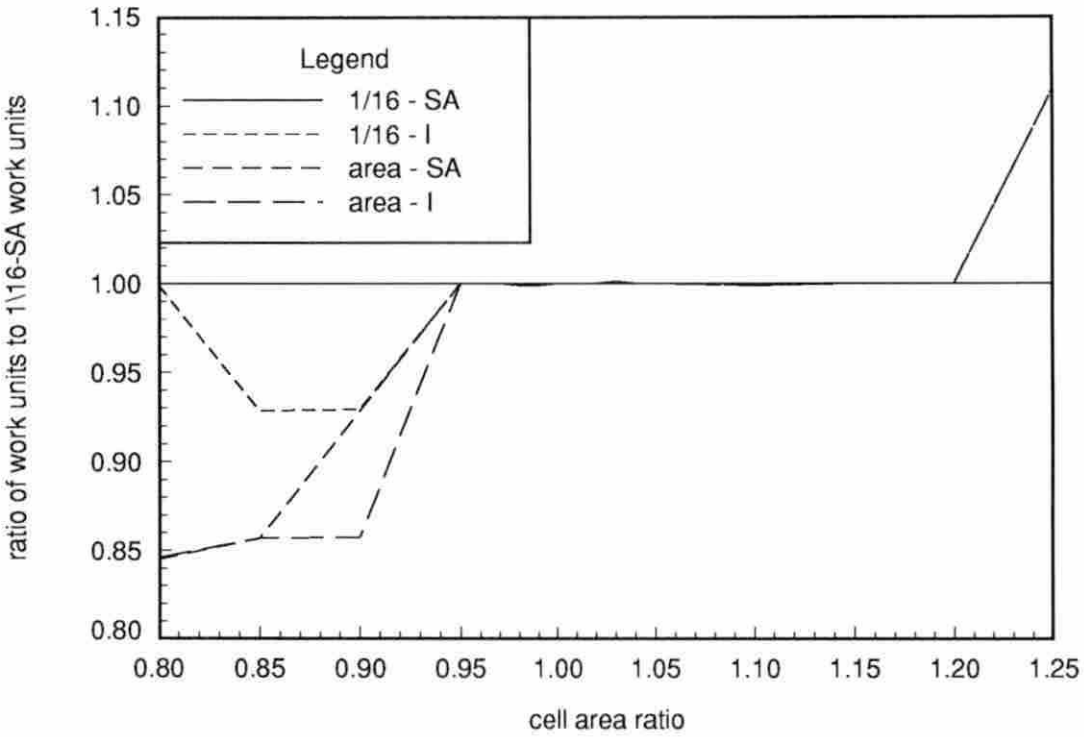

Figure 3.10: Transfer operator comparison using SIP and 4 levels

Table 3.12: Transfer operator comparison using GS iteration and 4 levels

cell area ratio	1/16 restriction		area restriction	
	simple averaging prolongation (work units)	interpolation prolongation (work units)	simple averaging prolongation (work units)	interpolation prolongation (work units)
0.80	198.1094	167.2813	167.0781	142.2500
0.85	167.5156	147.5781	151.4375	135.5313
0.90	127.9375	115.9844	121.8438	110.9219
0.95	83.3125	78.2969	80.3125	75.2969
0.96	74.3906	70.3594	72.3594	68.3594
0.97	66.4063	63.4219	64.4063	62.3594
0.98	59.4375	56.4531	58.4063	55.4219
0.99	52.4531	50.4531	51.4375	50.4375
1.00	46.4531	44.4688	46.4531	44.4688
1.01	40.4375	40.4063	40.4844	40.4844
1.02	35.4063	35.4063	38.3906	38.3906
1.03	35.5156	35.5000	35.3906	35.3594
1.04	35.5313	35.5313	34.4844	34.4531
1.05	35.5938	35.5625	35.5156	35.5156
1.10	42.5625	42.5938	42.5156	40.5313
1.15	64.5781	66.2969	58.3281	59.2031
1.20	88.5781	90.2813	75.1094	78.0000
1.25	120.5938	106.4063	87.8750	86.8750

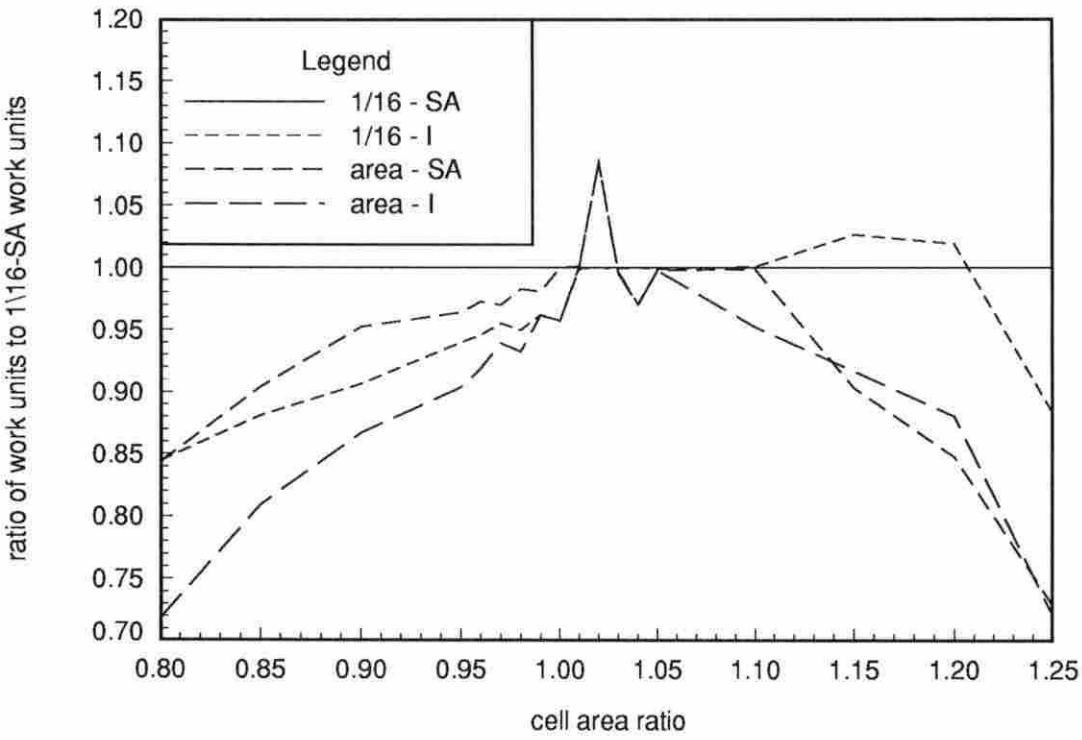

Figure 3.11: Transfer operator comparison using GS iteration and 4 levels

Table 3.13: Total iterations on a single stretched grid

cell area ratio	GS iterations	SIP iterations
0.80	342	35
0.85	424	46
0.90	529	64
0.95	628	85
0.96	643	88
0.97	655	92
0.98	663	94
0.99	668	96
1.00	669	96
1.01	665	96
1.02	656	94
1.03	643	90
1.04	623	84
1.05	593	79
1.10	558	73
1.15	498	55
1.20	433	41
1.25	383	32

formed best in most cases, especially in the more extreme cases of stretching. The case of 1/16 restriction and interpolation prolongation and the case of area restriction and simple averaging prolongation varied about equally for second best.

Only near a cell area ratio of about 1.01 to 1.03 did 1/16 restriction with simple averaging prolongation perform much better than the others which was somewhat of a surprise. Note that on a grid with a cell ratio of 1.0, area restriction becomes 1/16 restriction, and on a uniform grid, interpolation prolongation becomes simple averaging prolongation. Therefore, this case was expected to give its best performance in this range of cell area ratios, but was not expected to perform better than the other cases. In order to verify that the transfer operators were coded correctly, the two restriction operators were checked to see if they used the same number of work units when the cell area ratio was 1.0 and the two prolongation operators were checked to see if they used the same number of work units when a uniform grid was used. In all cases, both checks matched exactly, though data for the uniform grid are not shown in the tables.

4. NAVIER-STOKES EQUATIONS

The application of multigrid to a Navier-Stokes code will be discussed in this chapter. The code to be presented is a modified prerelease version of ALLSPD, a code being developed by the Aerothermochemistry Branch of the NASA Lewis Research Center for chemically reacting flows in aeropropulsion systems. However, the version of the code employed in this study did not contain the terms necessary for chemically reacting flow nor turbulence. It should be noted that the focus of this study is to examine the multigrid method in conjunction with this Navier-Stokes solver and not the Navier-Stokes solver itself. Therefore, the presentation of the solver may seem rather condensed. Additional discussion of the solver can be found in J. S. Shuen et al. [11].

4.1 Formulation

In this section, the mathematical equations which model fluid flow, the Navier-Stokes equations, are presented. They state the conservation of mass, momentum, and energy. The usual form of these equations for a Newtonian fluid can be expressed as follows:

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \vec{V} = 0 \tag{4.1}$$

$$\rho\frac{D\vec{V}}{Dt} = \vec{g} - \nabla p + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right] \tag{4.2}$$

$$\rho\frac{De}{Dt} + p\nabla\cdot\vec{V} = \frac{\partial Q}{\partial t} - \nabla\cdot\vec{q} + \Phi \tag{4.3}$$

where ρ is the density, $\vec{V}$ is the velocity vector, p is the hydrostatic pressure, e is the internal energy per unit mass, $\vec{g}$ is the body force, and δ_{ij} is the Kronecker delta function:

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

The quantity $\frac{\partial Q}{\partial t}$ represents heat energy production by external agencies, $\vec{q}$ is the heat conduction, and Φ is the dissipation function. Fourier's law of heat conduction will be assumed to apply so

$$\vec{q} = -k\nabla T$$

The dissipation function for a Newtonian fluid in a Cartesian coordinate system is

$$\Phi = \mu[2(u_x^2 + v_y^2 + w_z^2) + (v_x + u_y)^2 + (w_y + v_z)^2 + (u_z + w_x)^2 - \frac{2}{3}(u_x + v_y + w_z)^2]$$

The ideal gas equation of state and an equation for viscosity are used to close the system. Equations (4.1)-(4.3) are the general form of the governing equations written in Cartesian coordinates.

In the following section, the appropriate form of the governing equations for two-dimensional compressible flows will be derived followed by a section on applying an all-Mach-number formulation to the system of equations.

4.1.1 Governing Equation

The two-dimensional Navier-Stokes equations are obtained from equations (4.1)-(4.3) by dropping terms in the third dimension. They are recast in a strong conservation law form as follows:

$$\frac{\partial Q}{\partial t} + \frac{\partial (E - E_v)}{\partial x} + \frac{\partial (F - F_v)}{\partial y} = 0 \tag{4.4}$$

where

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E_t \end{pmatrix}$$

$$E = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho E_t + p)\, u \end{pmatrix} \qquad E_v = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix}$$

$$F = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho E_t + p)\, v \end{pmatrix} \qquad F_v = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y \end{pmatrix}$$

where q_x and q_y are the x and y components of the heat conduction vector $\vec{q}$ respectively, E_t is the total energy per unit volume which is defined as:

$$E_t = e + \frac{1}{2}\left(u^2 + v^2\right)$$

and the τ_{xx}, τ_{yy}, and τ_{xy} are the shear stresses which are defined as follows:

$$\begin{aligned}
\tau_{xx} &= \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \\
\tau_{yy} &= \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right) \\
\tau_{xy} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)
\end{aligned}$$

where μ for air is determined by the Sutherland formula shown below:

$$\mu = \frac{C_1 T^{3/2}}{T + C_2} \tag{4.5}$$

The Sutherland constants C_1 and C_2 are:

$$C_1 = 1.458 \times 10^{-6} \;\; kg/(ms\sqrt{K}), \quad C_2 = 110.4 \;\; K$$

The body force term $\vec{g}$ has been neglected in equation (4.4) since body forces were not considered in this study.

4.1.2 All-Mach-Number Formulation

The computation of low speed flows is difficult for the most widely used compressible flow algorithms. The fluid velocity for flows of interest in combustion applications is much smaller than the acoustic speed, yet the density variation is too significant to permit the use of incompressible solvers. In aeropropulsion systems, compressibility effects are significant due to heat release. Numerical algorithms developed for compressible flows encounter difficulties when applied to low Mach number flows due to two well-recognized reasons.

First, the two-dimensional Euler equations have the eigenvalues u, u, $u + c$, and $u - c$ (in the x-direction). As the Mach number becomes small, these eigenvalues differ by orders of magnitude, making the system very stiff.

The second problem of the compressible equations at low Mach numbers can be examined by nondimensionalizing the inviscid momentum equation. The four basic reference quantities chosen for this nondimensionalization will be u_{ref} velocity, ρ_{ref} density, p_{ref} pressure, and L_{ref} length. A reference time and speed of sound can be defined as $t_{ref} = L_{ref}/v_{ref}$ and $c_{ref} = \sqrt{\gamma p_{ref}/\rho_{ref}}$ respectively. After nondimensionalization, the two-dimensional inviscid x-momentum equation can be written in the following form:

$$\frac{\partial\left(\rho^* u^*\right)}{\partial t^*} + \frac{\partial\left(\rho^* u^{*2}\right)}{\partial x^*} + \frac{\partial\left(\rho^* u^* v^*\right)}{\partial y^*} + \frac{\partial\left(p^*/\gamma M_{ref}^2\right)}{\partial x^*} = 0$$

where M_{ref} is the reference Mach number and the superscript $*$ denotes a nondimensional quantity. The magnitude of the variables in the time and convective terms in the equation remain in the order of unity as the Mach number approaches zero; however, the expression being differenced in the numerator of the pressure gradient term becomes very large. When differenced, this results in large roundoff error which can prevent the calculation of low Mach number flows from converging.

In order to avoid the roundoff problem due in the pressure gradient, the pressure was decomposed into a reference pressure and a gauge pressure

$$p(x, y, t) = p_o + p_g(x, y, t)$$

Since p_o is constant, its contribution to the pressure gradient terms in the momentum equations is zero leaving only the derivative with respect to the gauge pressure. To maximize the benefit of this decomposition, p_o should consist of the majority of p. If p_o is chosen properly, the magnitude of the expression in the numerator of the pressure gradient term in the nondimensional momentum equation becomes of order

unity as the Mach number approaches zero, therefore removing the large roundoff due to this term.

Even though the pressure decomposition procedure removes the roundoff problems associated with the pressure gradient term at low Mach numbers, it also creates further problems in that p_g does not appear in the dependent variable Q (equation 4.4), and thus it is impossible to express the vectors E, F, E_v, and F_v as functions of Q alone. Pressure is not one of the dependent variables of conventional compressible flow algorithms but is calculated from the dependent variables and the equation of state. To avoid contamination of the pressure field by roundoff error, it is necessary to let p_g be one of the dependent variables.

The other major difficulty to be overcome at low Mach numbers is the stiffness of the system of equations mentioned earlier. The stiffness is due to the fact that the solution scheme attempts to honor two characteristic speeds (or times), one associated with the convective velocities and the other associated with the acoustic speed. This difficulty is overcome by preconditioning which introduces pseudo-time terms and clusters the characteristic speeds for the new problem in pseudo time very close to the convective speed.

The equations are transformed into a curvilinear coordinate system and returned to conservative form in order to have better conservative properties and the ability to capture shock waves. The resulting equations are

$$\Gamma\frac{\partial\hat{Q}}{\partial\tau}+\frac{\partial\tilde{Q}}{\partial t}+\frac{\partial\left(\tilde{E}-\tilde{E}_v\right)}{\partial\xi}+\frac{\partial\left(\tilde{F}-\tilde{F}_v\right)}{\partial\eta}=0 \tag{4.6}$$

where the preconditioning matrix Γ and the primitive variable vector $\hat{Q}$ are given as

$$\Gamma = \begin{bmatrix} \frac{1}{\beta} & 0 & 0 & 0 \\ \frac{u}{\beta} & \rho & 0 & 0 \\ \frac{v}{\beta} & 0 & \rho & 0 \\ \frac{H}{\beta} & 0 & 0 & \rho \end{bmatrix} \qquad \hat{Q} = \frac{1}{J} \begin{pmatrix} pg \\ u \\ v \\ H \end{pmatrix}$$

where H is the total enthalpy of the gas:

$$H = h + \frac{1}{2}\left(u^2 + v^2\right)$$

The vectors $\tilde{Q}$, $\tilde{E}$, $\tilde{F}$, $\tilde{E}_v$, and $\tilde{F}_v$ are defined as

$$\tilde{Q} = \frac{Q}{J}$$

$$\tilde{E} = \frac{1}{J}\left(\xi_x E + \xi_y F\right)$$

$$\tilde{F} = \frac{1}{J}\left(\eta_x E + \eta_y F\right)$$

$$\tilde{E_v} = \frac{1}{J}\left(\xi_x E_v + \xi_y F_v\right)$$

$$\tilde{F_v} = \frac{1}{J}\left(\eta_x E_v + \eta_y F_v\right)$$

In the above expressions ξ and η are the generalized spatial coordinates. The ξ_x, ξ_y, η_x, and η_y are the metric terms and the J is the transformation Jacobian. Note that the transformations above do not transform time because the grids in this study will be stationary. The vectors Q, E, F, E_v, and F_v are closely related to those presented in equation (4.4) with the exception that the pressure in the momentum equations

has been replaced by the gauge pressure as shown below:

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E_t \end{pmatrix}$$

$$E = \begin{pmatrix} \rho u \\ \rho u^2 + p_g \\ \rho uv \\ (\rho E_t + p)\, u \end{pmatrix} \qquad E_v = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix}$$

$$F = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p_g \\ (\rho E_t + p)\, v \end{pmatrix} \qquad F_v = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y \end{pmatrix}$$

One of the advantages of the dual time-stepping method is that the convergence of the iterative process is determined by the eigenvalue characteristics of the pseudo-time space and not by the eigenvalues of the original stiff system. The eigenvalues (in the ξ-direction) in the pseudo-time space can be obtained from the matrix $\Gamma^{-1}A$, where A is the Jacobian $\partial\tilde{E}/\partial\hat{Q}$. The eigenvalues (in the ξ-direction) for a real gas are

$$\lambda = \frac{1}{2}\left[U\left(1+\frac{\beta}{c^2}\right) \pm \sqrt{U^2\left(1-\frac{\beta}{c^2}\right)^2 + 4\beta\left(\alpha_1^2+\alpha_2^2\right)}\right], \quad U, \quad U \qquad (4.7)$$

where $\alpha_1 = \xi_x$, $\alpha_2 = \xi_y$, U is the contravariant velocity component defined as $U = \alpha_1 u + \alpha_2 v$, and c is the speed of sound. To obtain well-conditioned eigenvalues, the scaling factor β is taken to be

$$\beta = max\left(u^2 + v^2, v_{ref}^2\right)$$

where v_{ref} is the reference velocity used to calculate the Reynolds number making all the eigenvalues of the same order of magnitude.

A local time-stepping method is used for the pseudo-time steps. The local time steps are determined by:

$$\Delta\tau_{i,j} = \frac{dtrat}{\sqrt{\lambda_\xi^2 + \lambda_\eta^2}}$$

where λ_ξ and λ_η are the first eigenvalue listed in equation (4.7) in the ξ-direction and η-direction respectively evaluated with information at the local node i, j. The term $dtrat$ is simply a user selected constant used as a means of scaling the local pseudo-time steps and was manually optimized to the nearest 0.5 to produce the fewest number of work units. Optimized values typically ranged from 4 to 12.

4.2 Numerical Solution Algorithm

In this section, the numerical solution algorithm will be described. The discussion will cover the dual time-stepping integration procedure, the linearization method, explicit smoothing, the discretization method, the grid generation scheme, metric term evaluation, boundary conditions and implementation, convergence criterion, and the solution procedure. These topics will be described one by one. This section will not cover, however, the multigrid algorithm.

4.2.1 Dual Time-Stepping Integration Procedure

The ALLSPD code uses a dual time-stepping integration method to obtain time-accurate solutions for time-evolving problems. An implicit iterative procedure is used for asymptotic time-marching in pseudo time. The solution converged in pseudo time corresponds to a time-accurate solution in physical time. One advantage of the dual time-stepping method is that the convergence of the iterative process is determined by the eigenvalue characteristics on the pseudo-time space and not by the original stiff eigenvalues.

A general dual-time marching procedure is used for equation (4.6) by using a three-level backward differencing for physical time, an Euler implicit differencing for pseudo time, and a central differencing for space.

4.2.2 Linearization Method

The system of equations can be written in the following form:

$$\Gamma\frac{\hat{Q}^{k+1}-\hat{Q}^{k}}{\Delta\tau}+\frac{\tilde{Q}^{k+1}-\tilde{Q}^{n}}{\Delta t}+\frac{\partial\left(\tilde{E}-\tilde{E}_v\right)}{\partial\xi}^{k+1}+\frac{\partial\left(\tilde{F}-\tilde{F}_v\right)}{\partial\eta}^{k+1}=0 \qquad (4.8)$$

If only steady-state flows are of interest then this can be accomplished by setting $\Delta t = \infty$ and marching only in pseudo time or by performing a single iteration at each time step.

The terms at the $k+1$ time level need to be linearized for the construction of an implicit time marching scheme. The inviscid Jacobians used in the linearization are defined as

$$T=\frac{\partial\tilde{Q}}{\partial\hat{Q}}, \quad A=\frac{\partial\tilde{E}}{\partial\hat{Q}}, \quad B=\frac{\partial\tilde{F}}{\partial\hat{Q}}$$

where

$$T = \begin{pmatrix} \frac{1}{RT} & \frac{\rho u}{c_p T} & \frac{\rho v}{c_p T} & -\frac{\rho}{c_p T} \\ \frac{u}{RT} & \rho\left(1+\frac{u^2}{c_p T}\right) & \frac{\rho uv}{c_p T} & -\frac{\rho u}{c_p T} \\ \frac{v}{RT} & \frac{\rho uv}{c_p T} & \rho\left(1+\frac{v^2}{c_p T}\right) & -\frac{\rho v}{c_p T} \\ \frac{H}{RT}-1 & \frac{\rho uH}{c_p T} & \frac{\rho vH}{c_p T} & \rho\left(1-\frac{H}{c_p T}\right) \end{pmatrix}$$

$$A = \begin{pmatrix} \frac{U}{RT} & \rho\alpha_1 & \rho\alpha_2 & -\frac{\rho U}{c_p T} \\ \alpha_1+\frac{uU}{RT} & \rho\left(U+u\alpha_1\right) & \rho u\alpha_2 & -\frac{\rho uU}{c_p T} \\ \alpha_2+\frac{vU}{RT} & \rho v\alpha_1 & \rho\left(U+v\alpha_2\right) & -\frac{\rho vU}{c_p T} \\ \frac{UH}{RT} & \rho H\left(\alpha_1+\frac{uU}{c_p T}\right) & \rho H\left(\alpha_2+\frac{vU}{c_p T}\right) & \rho U\Phi \end{pmatrix}$$

and

$$U = \alpha_1 u + \alpha_2 v, \quad \alpha_1 = \xi_x, \quad \alpha_2 = \xi_y, \quad \Phi = 1 - \frac{H}{c_p T}$$

The Jacobian matrix B is obtained by letting $\alpha_1 = \eta_x$, $\alpha_2 = \eta_y$.

If the definition of the differential operator for the viscous terms is

$$L_v\left(\hat{Q}\right) = \frac{\partial \tilde{E}_v}{\partial \xi} + \frac{\partial \tilde{F}_v}{\partial \eta}$$

then the viscous terms can be linearized in the following manner

$$L_v\left(\hat{Q}\right)^{k+1} = L_v\left(\hat{Q}\right)^k + \left(\frac{\partial}{\partial \xi} R_{\xi\xi} \frac{\partial}{\partial \xi} + \frac{\partial}{\partial \eta} R_{\eta\eta} \frac{\partial}{\partial \eta}\right) \Delta\hat{Q}$$

After linearization, the equation becomes

$$\left\{\Gamma + \frac{\Delta\tau}{\Delta t} T + \Delta\tau\left(\frac{\partial A}{\partial \xi} - \frac{\partial}{\partial \xi} R_{\xi\xi} \frac{\partial}{\partial \xi}\right) + \Delta\tau\left(\frac{\partial B}{\partial \eta} - \frac{\partial}{\partial \eta} R_{\eta\eta} \frac{\partial}{\partial \eta}\right)\right\}^k \Delta\hat{Q}$$

$$= -\Delta\tau\left(R\right)^k \tag{4.9}$$

where

$$R^k = \frac{\tilde{Q}^k - \tilde{Q}^n}{\Delta t} + \frac{\partial\left(\tilde{E} - \tilde{E}_v\right)^k}{\partial\xi} + \frac{\partial\left(\tilde{F} - \tilde{F}_v\right)^k}{\partial\eta}$$

When the equation converges in pseudo time at the $n+1$ physical time level, the right-hand side goes to zero and $\tilde{Q}^{n+1} = \tilde{Q}\left(\hat{Q}^{k+1}\right)$ provides the time accurate solution.

The discretization of the derivative terms will be described in a later section. The final equation (4.9) is solved by a five-point coupled version of the modified strongly implicit procedure. The strongly implicit procedure was first presented by Stone [7], later modified by Schneider and Zedan [8], and extended to a coupled system by Chen and Pletcher [12].

4.2.3 Explicit Smoother

A second-order explicit term of the form shown below was added to the right-hand side of the equations in order to smooth the variables:

$$\sigma = \sigma_x \Gamma \frac{\partial^2 \tilde{Q}}{\partial \xi^2} + \sigma_y \Gamma \frac{\partial^2 \tilde{Q}}{\partial \eta^2}$$

where σ_x and σ_y are parameters to control the amount of smoothing desired and Γ is the preconditioning matrix. In all cases, a small amount of smoothing ($\sigma_x = \sigma_y = 0.01$) was used for consistency.

4.2.4 Discretization Method

The governing equations written in a generalized nonorthogonal coordinate system for two-dimensional compressible flows, equation (4.9) contain four types of

terms: time derivative terms, spatial first derivative terms, spatial second derivative terms, and spatial cross derivative terms. The discretization method for each type of derivative term is described below. It should be noted that the following discretization method, in general, was applied to the interior points only. The boundary equations were discretized in their own unique way and will be described in the section entitled "the boundary conditions and implementation".

1. *Time derivative terms*

   Letting ϕ be a general dependent variable, a first-order accurate forward difference was used such as:

   $$\left(\frac{\partial \phi}{\partial t}\right)^{n+1} = \frac{1}{\Delta t}\left(\phi^{n+1} - \phi^{n}\right)$$

   Note: Unless otherwise noted, no subscript for the dependent variable, ϕ, means that it is evaluated at the center point, i.e., (i, j). Furthermore, only those subscripts that are incremented for the (i, j) level will be shown. For example, $\phi_{i+1,j}^{n+1}$ will be written as ϕ_{i+1}^{n+1} .

2. *Spatial first derivative terms*

   A second-order central difference formula was used for these terms such as:

   $$\left(\frac{\partial \phi}{\partial \xi}\right)^{n+1} = \frac{1}{2}\left(\phi_{i+1}^{n+1} - \phi_{i-1}^{n+1}\right)$$

   $$\left(\frac{\partial \phi}{\partial \eta}\right)^{n+1} = \frac{1}{2}\left(\phi_{j+1}^{n+1} - \phi_{j-1}^{n+1}\right)$$

   where $\Delta\xi = \Delta\eta = 1$ is assumed.

3. *Spatial second derivative terms*

   These terms were evaluated in a fashion similar to the second-order central

difference formula:

$$\left(\frac{\partial^2\phi}{\partial\xi^2}\right)^{n+1} = \left(\phi_{i+1}^{n+1} - 2\phi^{n+1} + \phi_{i-1}^{n+1}\right)$$

$$\left(\frac{\partial^2\phi}{\partial\eta^2}\right)^{n+1} = \left(\phi_{j+1}^{n+1} - 2\phi^{n+1} + \phi_{j-1}^{n+1}\right)$$

However, these derivatives in the equations often appeared as and were evaluated in the following form:

$$\frac{\partial}{\partial\xi}\left(a\frac{\partial\phi}{\partial\xi}\right)^{n+1} = \left(a\frac{\partial\phi}{\partial\xi}\right)_{i+1/2}^{n+1} - \left(a\frac{\partial\phi}{\partial\xi}\right)_{i-1/2}^{n+1}$$

where $i + 1/2$ indicates a location halfway between i and $i + 1$; and $i - 1/2$ denotes a location halfway between $i - 1$ and i; and a represents a combination of metric terms and viscosity in the viscous terms in the momentum equations and the coefficient to the conduction terms in the energy equation. The values of $a_{i+1/2}$ and $a_{i-1/2}$ were determined as follows:

$$a_{i+1/2} = \frac{1}{2}\left(a_i + a_{i+1}\right)$$

$$a_{i-1/2} = \frac{1}{2}\left(a_i + a_{i-1}\right)$$

The first derivative terms at the half nodal points were evaluated as follows:

$$\left(\frac{\partial\phi}{\partial\xi}\right)_{i+1/2}^{n+1} = \phi_{i+1}^{n+1} - \phi_i^{n+1}$$

$$\left(\frac{\partial\phi}{\partial\xi}\right)_{i-1/2}^{n+1} = \phi_i^{n+1} - \phi_{i-1}^{n+1}$$

The expressions for the terms in the η direction were evaluated in a similar manner.

4. *Spatial cross derivative terms*

These terms were evaluated in a fashion similar to the second-order central difference formula:

$$\frac{\partial}{\partial \xi}\left(\frac{\partial \phi}{\partial \eta}\right)^{n+1} = \frac{1}{4}\left(\phi^{n+1}_{i+1,j+1} - \phi^{n+1}_{i+1,j-1} - \phi^{n+1}_{i-1,j+1} + \phi^{n+1}_{i-1,j-1}\right)$$

As with the previous set of derivative terms, these often appeared in a different form in the equations which was accommodated for:

$$\frac{\partial}{\partial \xi}\left(a\frac{\partial \phi}{\partial \eta}\right)^{n+1} = \frac{1}{2}\left[a_{i+1}\left(\frac{\partial \phi}{\partial \eta}\right)^{n+1}_{i+1} - a_{i-1}\left(\frac{\partial \phi}{\partial \eta}\right)^{n+1}_{i-1}\right]$$

where the first derivative terms were evaluated as:

$$\left(\frac{\partial \phi}{\partial \eta}\right)^{n+1}_{i+1} = \frac{1}{2}\left(\phi^{n+1}_{i+1,j+1} - \phi^{n+1}_{i+1,j-1}\right)$$

$$\left(\frac{\partial \phi}{\partial \eta}\right)^{n+1}_{i-1} = \frac{1}{2}\left(\phi^{n+1}_{i-1,j+1} - \phi^{n+1}_{i-1,j-1}\right)$$

4.2.5 Grid Generation Scheme

For all geometries considered in this study the grids were generated by algebraic means. Grid clustering was achieved near boundaries when needed by using the following Roberts transformation:

$$y^* = \alpha + (1-\alpha)\frac{\ln\left(\frac{\beta+[y(2\alpha+1)/h]-2\alpha}{\beta-[y(2\alpha+1)/h]+2\alpha}\right)}{\ln\left[\frac{\beta+1}{\beta-1}\right]} \tag{4.10}$$

where y^* is a coordinate in the transformed domain, y is a coordinate in the physical domain and h is the maximum values in the y direction.

For this transformation, if $\alpha = 0$ the mesh will be refined near $y = h$ only, whereas, if $\alpha = 0.5$ the mesh will be refined equally near $y = 0$ and $y = h$. In the

actual computation the inverse transformation of equation (4.10) is more useful. The inverse transformation is:

$$y = h\frac{(\beta+2\alpha)\left[\frac{\beta+1}{\beta-1}\right]^{\left[\left(y^*-\alpha\right)/(1-\alpha)\right]} - \beta + 2\alpha}{(2\alpha+1)\left\{1+\left[\frac{\beta+1}{\beta-1}\right]^{[(y^*-\alpha)/(1-\alpha)]}\right\}} \tag{4.11}$$

In equations (4.10) and (4.11), β is a stretching parameter for controlling the grid clustering. Its value should be greater than one. As β increases, the grid distribution becomes uniform.

Sometimes a row of points outside of the domain were needed. These were formed by extrapolation as shown in the examples below:

$$\begin{aligned} x_{0,j} &= 2x_{1,j} - x_{2,j} \\ x_{imax,j} &= 2x_{imax-1,j} - x_{imax-2,j} \\ y_{i,0} &= 2y_{i,1} - y_{i,2} \\ y_{i,jmax} &= 2y_{i,jmax-1} - y_{i,jmax-2} \end{aligned}$$

These are equivalent to either specifying that the second derivative of x with respect to ξ for lines of constant ξ; or the second derivative of y with respect to η for lines of constant η is zero on the boundaries of the domain.

4.2.6 Metric Term Evaluation

All metric terms were evaluated by a second-order central difference formula in accordance with the discretization scheme described earlier and shown below:

$$x_\xi = \frac{1}{2}\left(x_{i+1} - x_{i-1}\right)$$

$$
\begin{aligned}
x_\eta &= \frac{1}{2}\left(x_{j+1} - x_{j-1}\right) \\
y_\xi &= \frac{1}{2}\left(y_{i+1} - y_{i-1}\right) \\
y_\eta &= \frac{1}{2}\left(y_{j+1} - y_{j-1}\right)
\end{aligned}
$$

where $\Delta\xi = 1$ and $\Delta\eta = 1$ have been assumed. From these, the transformation Jacobian was calculated:

$$J = \frac{1}{x_\xi y_\eta - x_\eta y_\xi}$$

4.2.7 Boundary Conditions and Implementation

All boundary conditions were treated implicitly. For the two-dimensional compressible flow results presented in this study, several different types of boundaries were involved. They include inflow, outflow, symmetry, wall, and moving wall boundaries.

For the two-dimensional compressible flow calculations considered in this study, the boundary conditions are described as follows:

1. *Inflow boundary*

   The variables u, v, and T were specified at this boundary. However, pressure was extrapolated from interior points.

2. *Outflow boundary*

   Atmospheric pressure was specified. Extrapolations were used to obtain values for other variables.

3. *Symmetry boundary*

   If a line of symmetry exists in a flow, it is common practice to solve the problem for only one-half of the domain using a symmetry boundary condition. The

governing equations were written on the line of symmetry. All variables at the points outside the domain were obtained by the symmetry condition for u, p, and T and the antisymmetry condition for v.

4. *Wall boundary*

   Instead of writing the governing equations at this boundary, noslip conditions were used for the velocity components. For pressure and temperature, zero gradients normal to the wall were specified.

5. *Moving wall boundary*

   The moving wall boundary was treated the same as the wall boundary with one exception: the fluid was imparted the velocity of the wall in order to satisfy the noslip condition for the velocity components.

As mentioned previously, extrapolation was used to obtain variables at the extra points outside the computational domain. At the inlet, the second-order Lagrangian extrapolation formula was used:

$$\phi_0 = \frac{(x_0 - x_2)(x_0 - x_3)}{(x_1 - x_2)(x_1 - x_3)}\phi_1 + \frac{(x_0 - x_1)(x_0 - x_3)}{(x_2 - x_1)(x_2 - x_3)}\phi_2 + \frac{(x_0 - x_1)(x_0 - x_2)}{(x_3 - x_1)(x_3 - x_2)}\phi_3$$

In the above extrapolation ϕ_i represents a dependent variable and x_i indicates the spatial location of the variable ϕ_i. At the outlet, however, the flow is expected to be fully developed and thus does not change as rapidly with the streamwise direction of the flow compared to the inlet. Therefore, only a one point extrapolation was used at the outlet.

Often for internal steady flow calculations, the treatment of the pressure boundary condition at inflow and outflow deserves special attention. The pressure level calculated at the inflow boundary must be adjusted as the calculation process proceeds

if the specified inflow Reynolds number is to be maintained. The same adjustment must be applied to the pressure everywhere. This pressure adjustment procedure maintains a constant and predetermined mass flow rate. Without this adjustment, the Reynolds number of the final converged solution may drift from the desired value. Although a subroutine in ALLSPD is available to adjust this, it was not used, because the drift was found to be insignificant.

4.2.8 Convergence Criterion and Solution Procedure

For the calculations in this study, the convergence criterion was based upon the norm of all variables. This criterion is as follows:

$$\frac{\sum_{\substack{i=1\\i=imax}}^{\substack{j=1\\j=jmax}} \sum_{n=1}^{n=nq} \left| \frac{q^{k+1}-q^{k}}{q_{ref}^{k+1}} \right|^2}{4 \times imax \times jmax} \leq \epsilon = 10^{-7} \tag{4.12}$$

where k is the iteration level, n the variable index, nq the number of equations, $imax$ the number of grid points in the ξ direction, $jmax$ the number of grid points in η direction, and q_{ref} is a reference value.

The solution procedure for solving the Navier-Stokes equations on a single grid can be summarized as follows:

1. Set the initial conditions, generate the grid and associated metric terms.
2. Construct the coefficient matrix and the right-hand side vector.
3. Call the SIP solver to update the solution $\left(p_g, u, v, H\right)$.
4. Go to step 2 and repeat until steady-state solution or the specified maximum number of iterations is reached.

4.3 Multigrid

Since this part of the study involves only two levels and it was the wish of the author to devise a scheme that worked adequately without adjusting any more "knobs" for each individual case than those found in the original program, the multigrid scheme which was applied to the Navier-Stokes equations is rather simplistic. Nonetheless, several candidate cycles were evaluated before selecting the scheme which was adopted. The scheme starts by performing ten iterations on the fine grid before transferring to the coarse grid. On the coarse grid, a single iteration is performed and the convergence criterion evaluated. Iterations then continue on the coarse grid until the convergence criterion has decreased a factor of ten from the first iteration. The problem is then transferred back to the fine grid where the cycle repeats itself.

4.4 Results

The results to be shown for the most part will consist of the model case of a driven cavity. The Reynolds number, grid size, and grid generation scheme were all varied in order to compare the computational effort of multigrid to that of the single grid. These are then be followed by the single example of the model case of developing flow between two parallel flat plates in order to show that the driven cavity multigrid results are not unique to that problem alone. The working fluid in both problems was air at a reference temperature of 298.15 K.

In all cases, the code was compiled on a Cray YMP8/8128 with FORTRAN compiler flags -Wf "-a static" -Zv which took advantage of vectorization. A coupled

SIP solver was used to solve the system of equations which was written with special care towards vectorization. The original code vectorized very well as did the new code when working with a single grid. However, when the new code was used with multigrid, it was found not to vectorize as well. Since no attention was given to vectorization in this study, the author was not surprised by this result. More details will be given to this matter at the end of the driven cavity results.

4.4.1 Driven Cavity

Results of multigrid's ability to accelerate the driven cavity problem are presented here. The driven cavity consists of a square box with a moving lid which is suddenly started (Figure 4.1). The initial condition is a stationary fluid (air). The Reynolds number is based upon a reference length which is the width of the box and a reference velocity which is the velocity of the lid. The Reynolds number was varied across 100, 400, 1000, and 3200. The grid sizes examined were 65x65 and 129x129. Both a uniform and a stretched grid were used. The parameters used for generating the stretched grid (discussed in a previous section) were $\alpha = 0.5$ and $\beta = 1.03$. The stretched 65x65 grid used is shown in Figure 4.2.

Figure 4.3 shows the u velocity along the vertical centerline of the driven cavity for the computations on the single 129x129 uniform grid. The symbols represent the results from Ghia et al. [13] on a uniform 129x129 grid while the lines are that of the present study. The correlation between the two is very good across all Reynolds numbers examined (100, 400, 1000).

Likewise, Figure 4.4 shows the v velocity along the horizontal centerline of the driven cavity on the same grid also compared to results of Ghia et al. Again, agree-

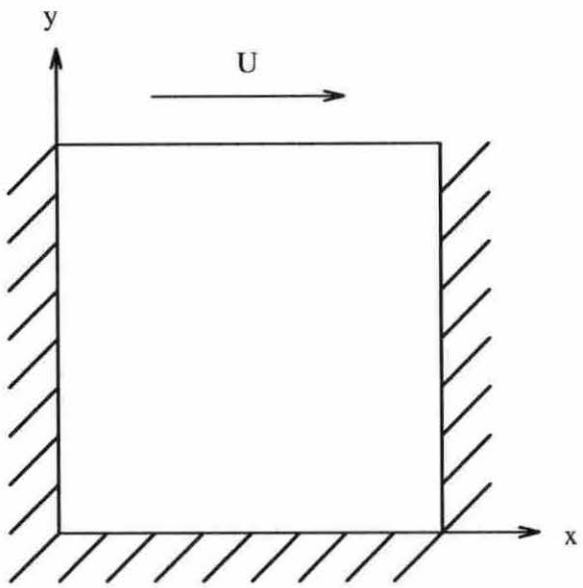

Figure 4.1: Geometry of the driven cavity

ment is very good for all the Reynolds numbers examined.

Table 4.1 shows the number of work units and cpu seconds necessary to solve the driven cavity problem at Reynolds numbers of 100, 400, and 1000 on a single 129x129 uniform grid. Table 4.2 shows the same results when multigrid was used. The column labeled *dtrat* refers to the variable (discussed in a previous section) which scaled the magnitude of the local pseudo-time steps. In all cases, *dtrat* was optimized to the nearest 0.5 for the least number of work units and was the same on both grids.

Table 4.3 shows the ratio of improvement in both the number of work units and cpu time of the multigrid case compared to the single grid case. The ratio of work unit and cpu time improvement decreased from 4.7 to 2.6 and from 4.2 to 2.3 respectively as the Reynolds number increased. These improvements are similar in

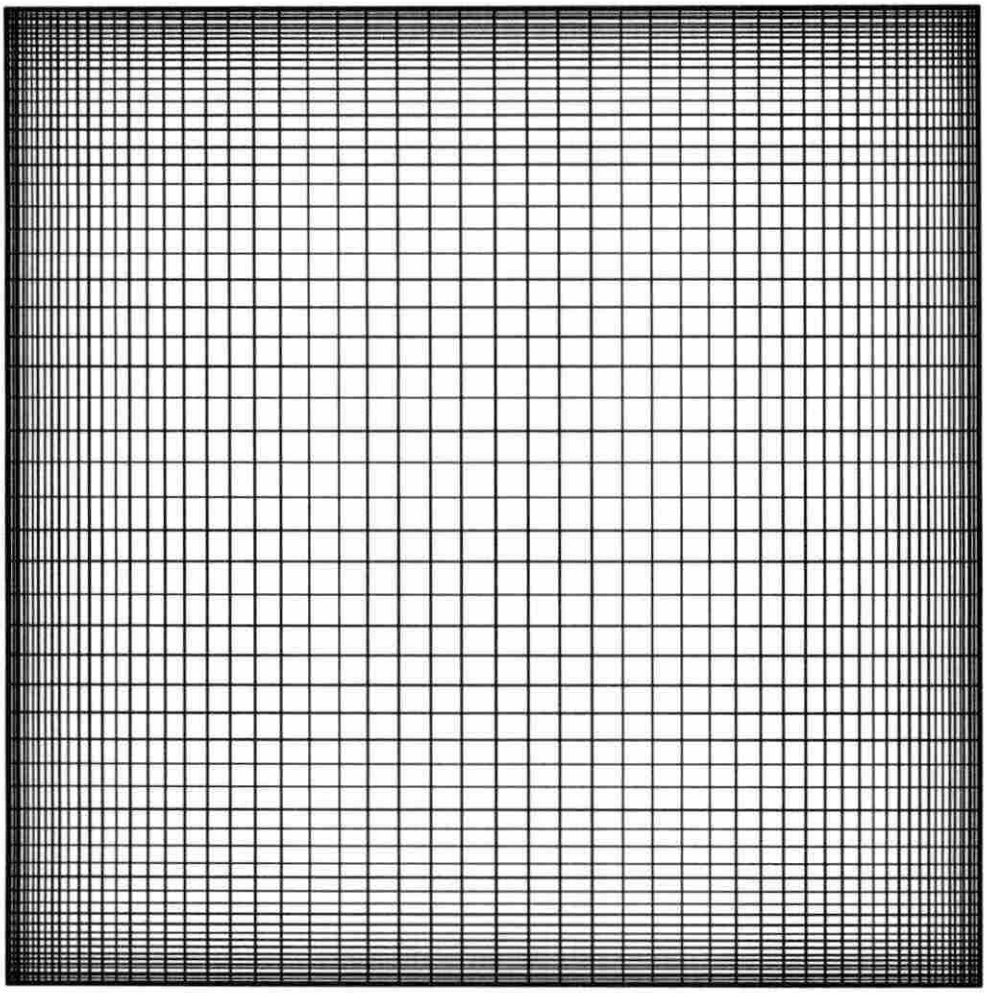

Figure 4.2: A sample 65x65 stretched grid used for the driven cavity

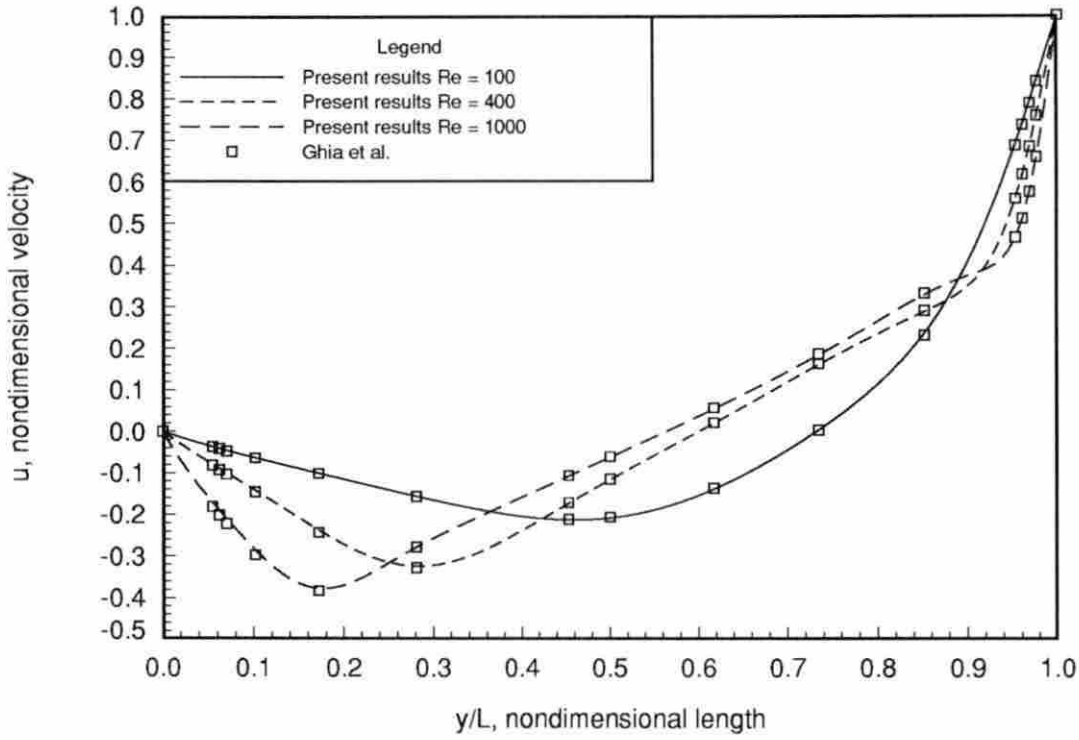

Figure 4.3: The nondimensional u velocity component along the vertical centerline of the driven cavity computed on a single 129x129 uniform grid

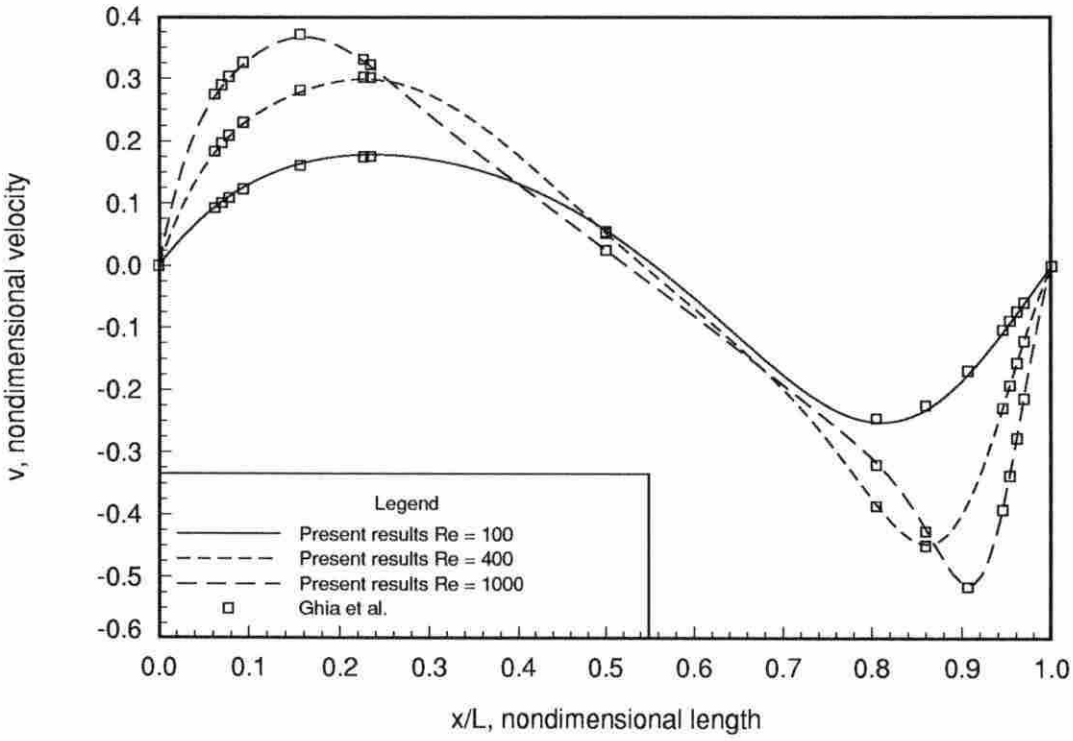

Figure 4.4: The nondimensional v velocity component along the horizontal centerline of the driven cavity computed on a single 129x129 uniform grid

Table 4.1: Single 129x129 uniform grid comparison

Re	dtrat	work units	cpu seconds
100	4.5	1937	1004.1
400	9.5	1108	571.7
1000	7.0	2312	1203.1

Table 4.2: Multigrid 129x129 uniform grid comparison

Re	dtrat	work units	cpu seconds
100	5.5	408.50	236.9
400	11.5	350.25	198.7
1000	7.0	898.25	533.3

size to those found when Laplace's equation was solved using two levels.

Next, the results using the same size, but stretched, grid are examined. Figure (4.5) shows the u velocity along the the vertical centerline while Figure (4.6) shows the v velocity along the horizontal centerline for the driven cavity cases which were calculated for Reynolds numbers of 100, 400, 1000, and 3200 using this grid. The correlation of the profiles to others is very good at Reynolds numbers of 100, 400, and 100; and good at 3200.

Table 4.4 shows the number of work units and cpu seconds required to solve the single grid case for the range of Reynolds numbers examined while Table 4.5 shows

Table 4.3: Driven cavity improvement on 129x129 uniform grids

Reynolds number	work units	cpu time
100	4.74	4.24
400	3.16	2.88
1000	2.57	2.26

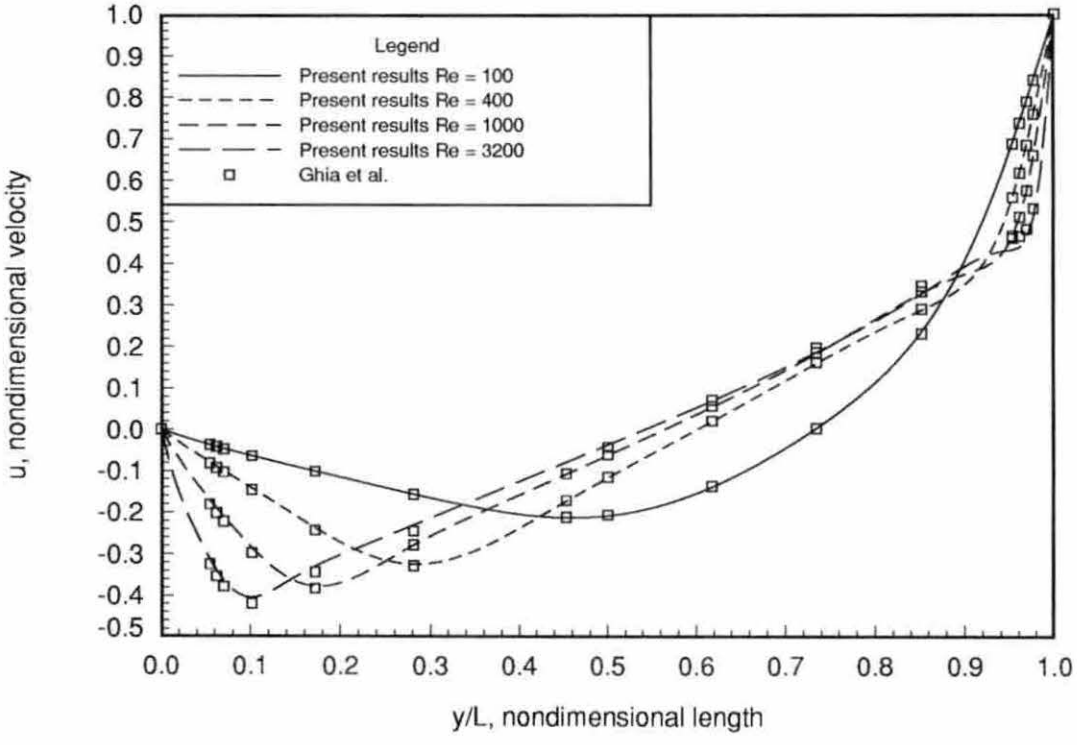

Figure 4.5: The nondimensional u velocity component along the vertical centerline of the driven cavity computed on a single 129x129 stretched grid

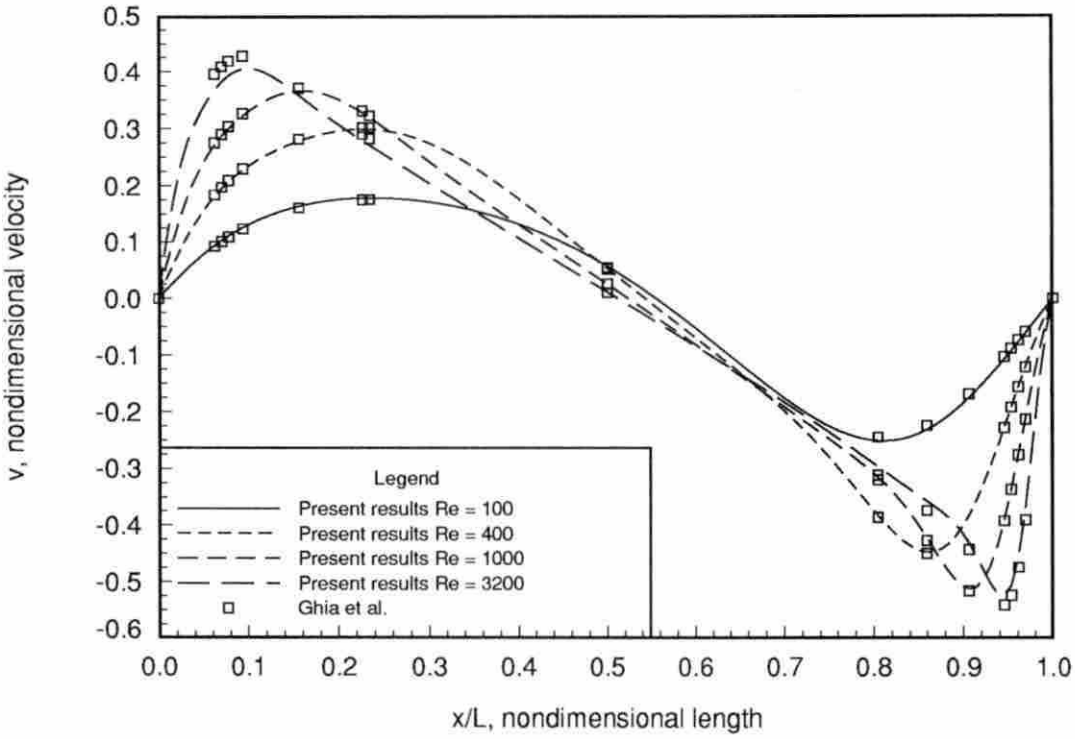

Figure 4.6: The nondimensional v velocity component along the horizontal centerline of the driven cavity computed on a single 129x129 stretched grid

Table 4.4: Single 129x129 stretched grid comparison

Re	dtrat	work units	cpu seconds
100	6.5	725	375.8
400	8.5	885	460.9
1000	7.5	1243	646.7
3200	5.5	4576	2382.7

Table 4.5: Multigrid 129x129 stretched grid comparison

Re	dtrat	work units	cpu seconds
100	6.0	302.50	172.9
400	8.0	325.25	191.8
1000	6.0	521.75	316.0
3200	4.5	1202.50	726.1

the same when multigrid was used.

Table 4.6 shows the ratio of improvement in both the number of work units and cpu time of the multigrid case compared to the single grid case. The improvement in the number of work units varied from 2.4 to 3.8 while the improvement in cpu time varied from about 2.2 to 3.3. These improvements are not as large as they were with the uniform grid. However, the single stretched grid case took far fewer work units to converge when compared to the uniform grid case at each Reynolds number.

Table 4.6: Driven cavity improvement on 129x129 stretched grids

Reynolds number	work units	cpu time
100	2.40	2.17
400	2.72	2.40
1000	2.38	2.05
3200	3.81	3.28

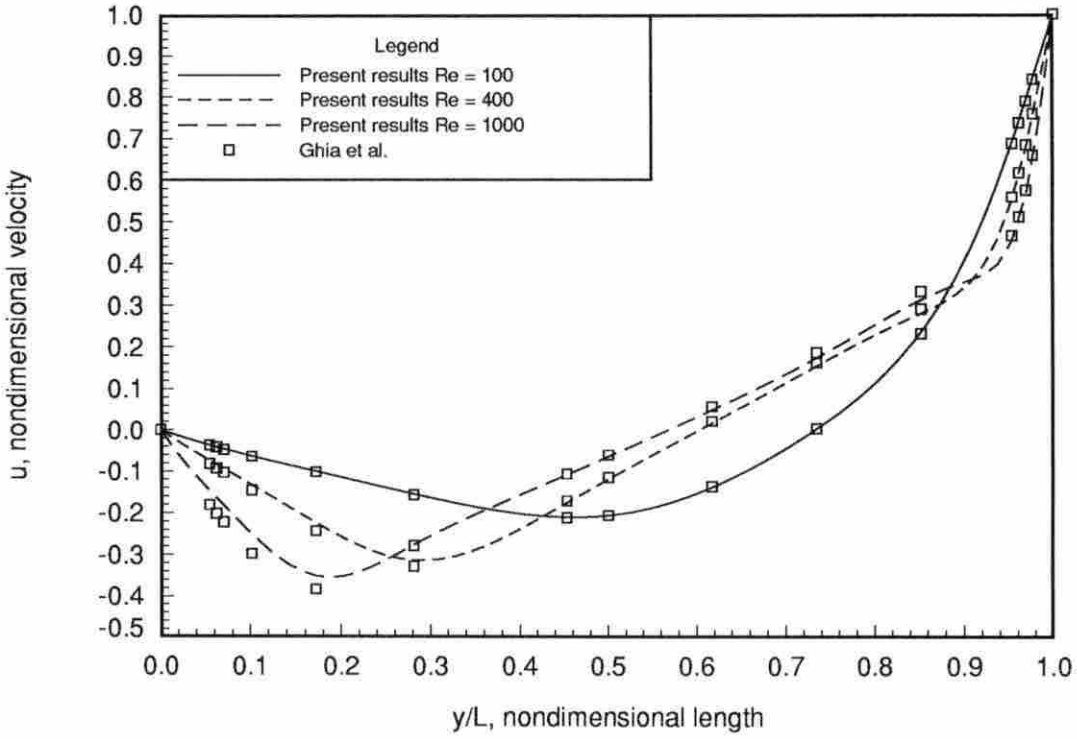

Figure 4.7: The nondimensional u velocity component along the vertical centerline of the driven cavity computed on a single 65x65 uniform grid

Continuing the trend, the results using the 65x65 uniform grid are examined. Figure (4.7) shows the u velocity along the vertical centerline while Figure (4.8) shows the v velocity along the horizontal centerline for the driven cavity cases which were calculated for Reynolds numbers of 100, 400, and 1000. Even though a smaller grid was used, the results still correlate quite well, except perhaps at $Re = 1000$.

Table 4.7 shows the number of work units and cpu seconds required to solve the single grid case for the range of Reynolds numbers examined while Table 4.8 shows the same when multigrid was used.

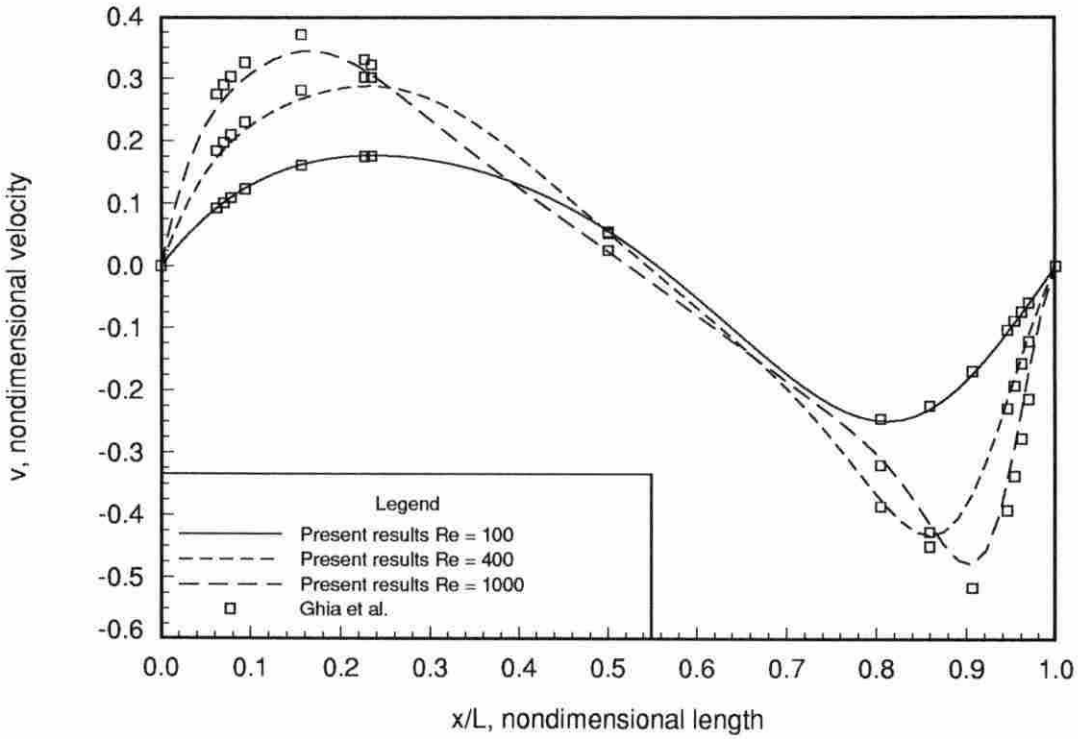

Figure 4.8: The nondimensional v velocity component along the horizontal centerline of the driven cavity computed on a single 65x65 uniform grid

Table 4.7: Single 65x65 uniform grid comparison

Re	dtrat	work units	cpu seconds
100	5.5	638	94.6
400	8.0	566	84.3
1000	6.5	1265	189.8

Table 4.8: Multigrid 65x65 uniform grid comparison

Re	dtrat	work units	cpu seconds
100	6.5	168.75	29.2
400	7.0	268.00	48.8
1000	5.5	760.50	140.4

Table 4.9 shows the ratio of improvement in both the number of work units and cpu time of the multigrid case compared to the single grid case. The improvement in the number of work units varied from 1.7 to 3.8 while the improvement in cpu time varied from about 1.4 to 3.2. These improvements are not as large as they were with the larger grid. However, this is expected since the same trend was found when Laplace's equation was solved: the improvement tends to decrease with a decreasing number of grid points.

Finally, the results using the 65x65 stretched grid are examined. Figure (4.9) shows the u velocity along the vertical centerline while Figure (4.10) shows the v

Table 4.9: Driven cavity improvement on 65x65 uniform grids

Reynolds number	iterations	cpu time
100	3.78	3.24
400	2.11	1.73
1000	1.66	1.35

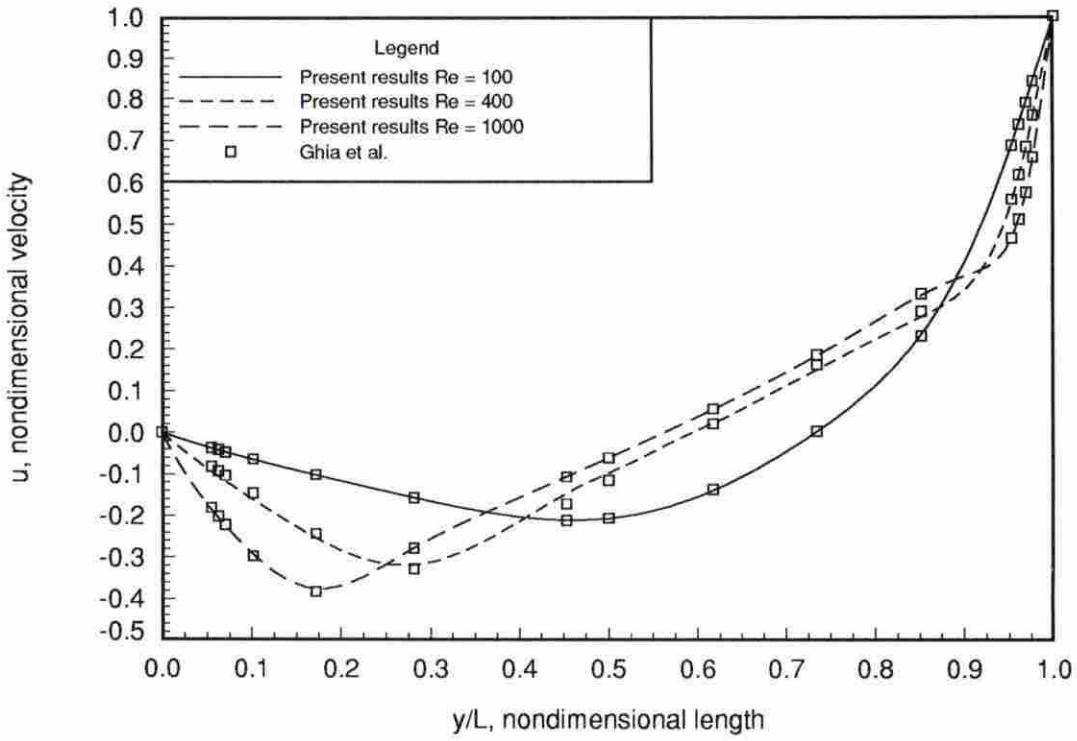

Figure 4.9: The nondimensional u velocity component along the vertical centerline of the driven cavity computed on a single 65x65 stretched grid

velocity along the horizontal centerline for the driven cavity cases which were calculated for Reynolds numbers of 100, 400, and 1000. The results correlate quite well, except at perhaps $Re = 400$.

Table 4.10 shows the number of work units and cpu seconds required to solve the single grid case for the range of Reynolds numbers examined (100, 400, 1000) while Table 4.11 shows the same when multigrid was used.

Table 4.12 shows the ratio of improvement in both the number of work units and cpu time of the multigrid case compared to the single grid case. The improvement in

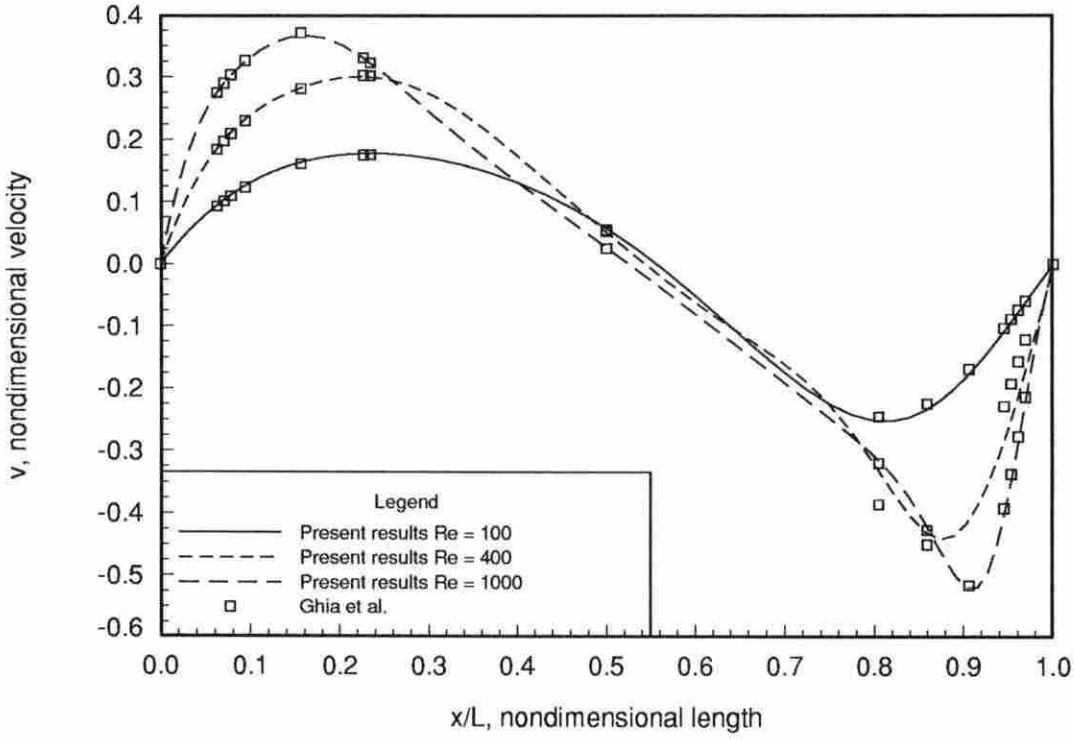

Figure 4.10: The nondimensional v velocity component along the horizontal centerline of the driven cavity computed on a single 65x65 stretched grid

Table 4.10: Single 65x65 stretched grid comparison

Re	dtrat	work units	cpu seconds
100	8.5	263	38.6
400	9.5	365	53.4
1000	6.5	815	121.7

Table 4.11: Multigrid 65x65 stretched grid comparison

Re	dtrat	work units	cpu seconds
100	7.5	150.25	25.3
400	6.5	205.25	38.1
1000	5.0	413.75	77.3

the number of work units varied from 1.8 to 2.0 while the improvement in cpu time varied from about 1.4 to 1.6. Again, these improvements are not as large as they were with the larger grid, but are expected since the same trend was found when Laplace's equation was solved: the improvement tends to decrease with a decreasing number of grid points.

As had been mentioned previously, the multigrid cases did not vectorize as well as the single grid cases. This should not be thought of as a negative aspect of multigrid since much careful work was taken by others to vectorize the original code and virtually none, as of yet, to vectorize the new additions to the code. At this time, the

Table 4.12: Driven cavity improvement on 65x65 stretched grids

Reynolds number	work units	cpu time
100	1.75	1.53
400	1.78	1.40
1000	1.97	1.57

affect of thoughtful vectorization on the multigrid procedure can not be determined. On the one hand, careful attention to vectorization could significantly increase the improvement over the single grid problem. On the other hand, vectorization on the coarsest of grids may be rather impractical due to small array sizes.

In order to check the effects of vectorization on the multigrid procedure, two tests were employed. The first was to perform some calculations with a version of the code compiled on the Cray YMP in a manner which forced the compiler to not take advantage of vectorization. This comparison showed that ratio of improvement in cpu time was lower when vectorization was taken advantage of. This means that the multigrid case did not vectorize as well as the single grid case. Therefore, examining the ratio of improvement in cpu time between the single grid and the multigrid cases while taking advantage of vectorization was not a completely fair measurement since the multigrid aspect of the code was not written for vectorization.

The second check involved determining what the cpu improvement would have been if the driven cavity had been solved without vectorization. Since using a vector machine for this would be a waste of resources, a scalar workstation was employed. All of the 65x65 comparisons between the single and multigrid cases were repeated on an HP 9000/730 workstation using +es -R8 -K +O3 as FORTRAN compiler options. It was found that the improvement in cpu time correlated extremely well with the improvement in work units. At worst, the difference between the two numbers was +/- 0.03. This is an improvement over what the Cray had computed and is encouraging news in that the improvement in cpu time for use on workstations may be indicative of the potential of careful vectorization of the multigrid procedure.

Surprisingly, the improvements found compare quite well to those found when

Laplace's equation was solved using two grids. There had been some worry before the study began that multigrid performance might severely degrade when applied to equations which were not elliptic. This may yet happen when the number of levels is increased, but current results are encouraging. Next, a very common test case, flow between two parallel plates, will be considered in order to check that these observations about multigrid are not unique to the driven cavity alone.

4.4.2 Developing Flow in a Channel

In order to show that the multigrid characteristics observed with the driven cavity were not completely problem dependent, a calculation was made for the model problem of a constant area, two-dimensional channel inlet without heat transfer (Figure 4.11). A uniform flow enters the channel. It is well known that an incompressible flow will eventually become fully developed with a parabolic velocity distribution. The centerline velocity will reach a value 1.5 times the inlet velocity. A very low Mach number flow will be examined which adequately approximates the incompressible flow even though a compressible formulation is used.

Due to the symmetric nature of this channel flow, the solution was only computed in the upper half of the channel. Only a Reynolds number of 75 at a Mach number of 0.0025 was examined. The Reynolds number was based upon the inlet velocity, and the half-height of the channel. The calculation was performed on a 61x21 grid, 30 half-heights in length. Figure 4.12 compares the centerline velocity to that of TenPas and Pletcher [14]. Correlation between the two is very good.

Table 4.13 shows the number of work units and cpu seconds (on the previously mentioned scalar workstation) required to solve the problem. The improvements

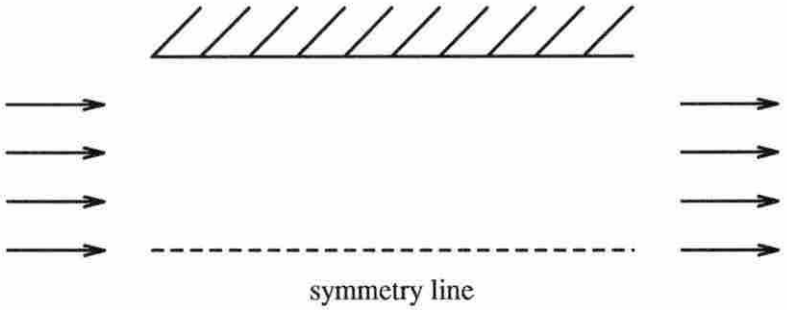

Figure 4.11: Developing channel

is about 1.74 in works units and 1.67 in cpu time. Even though the improvement does not seem as high as the improvement seen with the driven cavity, the channel calculation did not require as much computational effort or as many points to model as did the driven cavity. From the results of the previous chapter, a decrease in improvement for this problem would seem likely.

Table 4.13: Developing channel improvement

	dtrat	work units	cpu seconds
single grid	9.5	566	289.06
multigrid	9.0	325	172.76

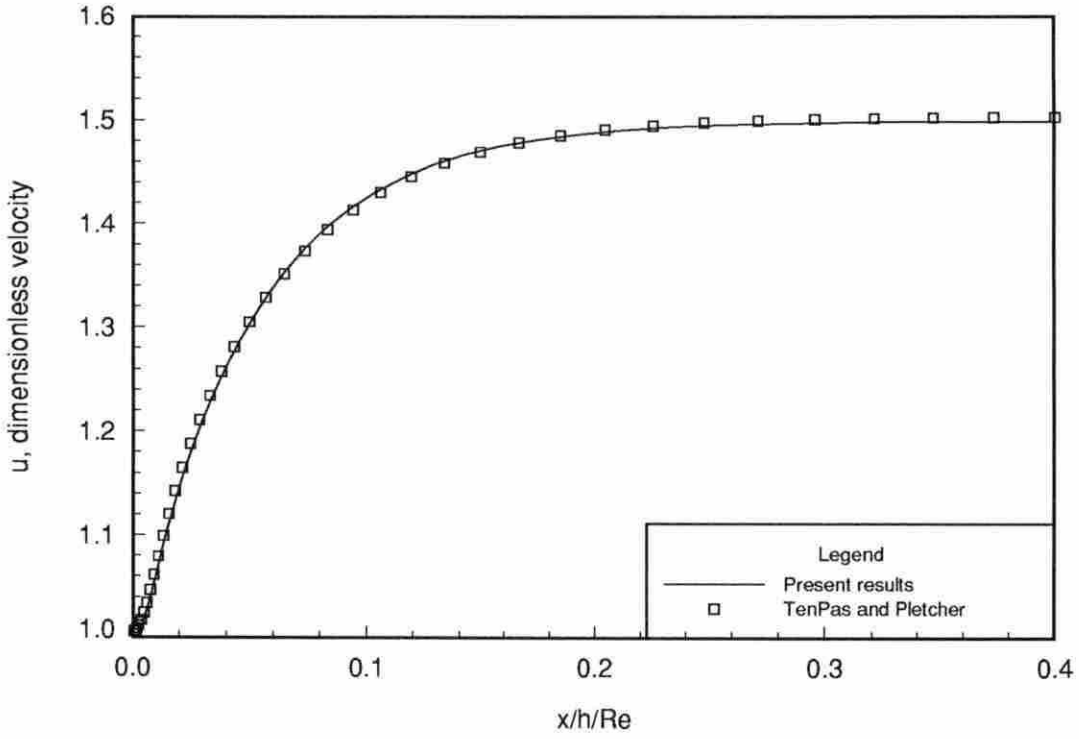

Figure 4.12: Developing channel centerline velocity comparison

5. CONCLUSIONS AND FUTURE WORK

An efficient two-level multigrid acceleration scheme has been developed and applied to an all-Mach-number, coupled, strongly implicit procedure for the Navier-Stokes equations. When applied to driven cavity flows from a Reynold's number of 100 to 3200, acceleration was found to be on the order of 1.4 to 4.7 which is similar to that found when Laplace's equation was solved with two levels.

From Laplace's equation, it was found that multigrid greatly reduced the advantage of the strongly implicit procedure over Gauss-Seidel iteration in terms of cpu time as the number of levels increased. This suggests that a solver not as efficient as the strongly implicit procedure could perform with an acceptable trade-off in computational effort when using multigrid to solve the Navier-Stokes equations. This is good news in that SIP, despite it's superb performance, requires more memory than many other solvers making it unattractive for solving larger problems.

Of the possibilities for future work, increasing the number of grids has the greatest potential and priority. The experiences solving Laplace's equation with multigrid indicate that an increase in the number of levels greatly reduces the required computational effort. As the number of levels increases, however, the level-switching strategy may have to be modified. Even though the strategy employed in this study is rather effective, it is rather simplistic and may not perform as well with more

levels. For example, the same value for the pseudo-time scaling parameter *dtrat* was employed on both levels in this study. It may be desirable to vary this parameter for each grid in some sort of way which is not problem dependent in order to increase the improvements in computational effort without having to "tweak another knob".

The eventual goal of this ongoing project is to develop a multigrid solver application for reacting flows. To investigate this topic, the current solver needs to be expanded to handle chemically reacting flows or abandoned in favor of adding multigrid to another ALLSPD code which already contains chemistry. Recently, Liu, Liu, and McCormick [15] reported an improvement of over two orders of magnitude using multigrid in the computation of two-dimensional laminar diffusion flames involving five species using a 130x130 grid. Solving the system of equations with a line-distributive relaxation, five decimal-place accuracy was achieved in about 200 work units. The author of this thesis believes that it is possible for ALLSPD to solve a similar problem with several levels in the same number or fewer work units.

As noted in the previous chapter, the multigrid algorithm developed in this study did not vectorize as well as the original code. This is a problem which may be fairly easy and desirable to overcome.

The possibilities just mentioned for future work have the highest priority. The ALLSPD code, however, is continuously changing with further developments and with these developments the multigrid method will have to evolve with them. These developments might include grid adaptation, use with solvers other than the modified strongly implicit procedure, three dimensional problems, the use of parallel computers, and unstructured grids.

BIBLIOGRAPHY

[1] Brandt, A., "Multilevel Computations: Review and Recent Developments" *Multigrid Methods: Theory, Applications, and Supercomputing*, (Edited by McCormick, S. F.), Marcel Dekker, New York, 35-62 (1988).

[2] Southwell, R. V., "Stress Calculation in Frameworks by the Method of Systematic Relaxation of Constraints." *Proc. Roy. Soc. London, Ser A*, **151**, 56-95 (1935).

[3] Fedorenko, R. P., "On the Speed of Convergence of an Iterative Process." *Ž. Vyčisl Mat. i Mat. Fiz.*, **4**, 559-564 (1964).

[4] Brandt, A., *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, Sankt Augustin, Darmstadt (1984).

[5] Jameson, A. and Yoon, S., "Multigrid Solution of the Euler Equations Using Implicit Schemes," *AIAA Journal*, **24**, 1737-1743 (1986).

[6] Smith, W. A., "Multigrid Solution of Transonic Flow on Unstructured Grids," paper presented at ASME WAM, November 1990, Dallas, Texas.

[7] Stone, H. L., "Iterative Solution of Implicit Approximations of Multi-dimensional Partial Differential Equations," *SIAM Journal of Numerical Analysis*, **5**, 530-558 (1968).

[8] Schneider, G. E. and Zedan, M., "A Modified Strongly Implicit Procedure for the Numerical Solution of Field Problems," *Numerical Heat Transfer*, **4**, 1-19 (1981).

[9] Anderson, D. A., Tannehill, J. C. and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill Book Co., New York (1984).

[10] Miller, T. F. and Schmidt, F. W., "Evaluation of a Multi-level Technique Applied to the Poisson and Navier-Stokes Equations," *Numerical Heat Transfer*, **13**, 1-26 (1988).

[11] Shuen, J. S., Chen, K. H. and Choi, Y., "A Time-Accurate Algorithm for Chemical Non-Equilibrium Viscous Flows at All-Speeds," AIAA 92-3639 (1992).

[12] Chen, K. H. and Pletcher, R. H., "Primitive Variable, Strongly Implicit Calculation Procedure for Viscous Flows at All Speeds," *AIAA Journal*, **29**, 1241-1249 (1991).

[13] Ghia, U., Ghia, K. N. and Shin, C. T., "High-Re Solutions for Incompressible Flow Using The Navier-Stokes Equations and a Multigrid Method," *Journal of Computational Physics*, **48**, 387-411 (1982).

[14] TenPas, P. W. and Pletcher, R. H., "Solution of the Navier-Stokes Equations for Subsonic Flows using a Coupled Space-Marching Method," AIAA 87-1173-cp (1987).

[15] Liu, C., Liu, Z. and McCormick, S., "Multigrid Methods for Numerical Simulation of Laminar Diffusion Flames," AIAA 93-0236 (1993).

APPENDIX A. THE COEFFICIENT TERMS A_1 THROUGH A_9 FOR LAPLACE'S EQUATION

The coefficient terms for the vector of unknowns, A_1 through A_9, which were developed for solving Laplace's equation are presented here. These terms were generated from Equation 3.4 and the discretization method introduced in Chapter 3.

$$
\begin{aligned}
A_1 \;=\; & \frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{i+1/2}}{J_{i+1/2}\left(\eta_{i+1/2,j+1}-\eta_{i+1/2,j-1}\right)\left(\xi_{i+1/2}-\xi_{i-1/2}\right)}\right. \\
& \left. -\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{i-1/2}}{J_{i-1/2}\left(\eta_{i-1/2,j+1}-\eta_{i-1/2,j-1}\right)\left(\xi_{i+1/2}-\xi_{i-1/2}\right)}\right] \\
& +\frac{\left(x_\xi^2+y_\xi^2\right)\Big|_{j+1/2}}{J_{j+1/2}\left(\eta_{j+1}-\eta_j\right)\left(\eta_{j+1/2}-\eta_{j-1/2}\right)}
\end{aligned}
$$

$$
\begin{aligned}
A_2 \;=\; & \frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{i+1/2}}{J_{i+1/2}\left(\eta_{i+1/2,j+1}-\eta_{i+1/2,j-1}\right)\left(\xi_{i+1/2}-\xi_{i-1/2}\right)}\right. \\
& \left. +\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{j+1/2}}{J_{j+1/2}\left(\xi_{i+1,j+1/2}-\xi_{i-1,j+1/2}\right)\left(\eta_{j+1/2}-\eta_{j-1/2}\right)}\right]
\end{aligned}
$$

$$
\begin{aligned}
A_3 \;=\; & \frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\big|_{j+1/2}}{J_{j+1/2}\left(\xi_{i+1,j+1/2}-\xi_{i-1,j+1/2}\right)\left(\eta_{j+1/2}-\eta_{j-1/2}\right)}\right. \\
& \left. -\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\big|_{j-1/2}}{J_{j-1/2}\left(\xi_{i+1,j-1/2}-\xi_{i-1,j-1/2}\right)\left(\eta_{j+1/2}-\eta_{j-1/2}\right)}\right] \\
& +\frac{\left(x_\eta^2+y_\eta^2\right)_{i+1/2}}{J_{i+1/2}\left(\xi_{i+1}-\xi_i\right)\left(\xi_{i+1/2}-\xi_{i-1/2}\right)}
\end{aligned}
$$

$$
\begin{aligned}
A_4 \;=\; & -\frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\big|_{i+1/2}}{J_{i+1/2}\left(\eta_{i+1/2,j+1}-\eta_{i+1/2,j-1}\right)\left(\xi_{i+1/2}-\xi_{i-1/2}\right)}\right. \\
& \left. +\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\big|_{j-1/2}}{J_{j-1/2}\left(\xi_{i+1,j-1/2}-\xi_{i-1,j-1/2}\right)\left(\eta_{j+1/2}-\eta_{j-1/2}\right)}\right]
\end{aligned}
$$

$$
\begin{aligned}
A_5 \;=\; & -\frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\big|_{i+1/2}}{J_{i+1/2}\left(\eta_{i+1/2,j+1}-\eta_{i+1/2,j-1}\right)\left(\xi_{i+1/2}-\xi_{i-1/2}\right)}\right. \\
& \left. -\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\big|_{i-1/2}}{J_{i-1/2}\left(\eta_{i-1/2,j+1}-\eta_{i-1/2,j-1}\right)\left(\xi_{i+1/2}-\xi_{i-1/2}\right)}\right] \\
& +\frac{\left(x_\xi^2+y_\xi^2\right)_{j-1/2}}{J_{j-1/2}\left(\eta_j-\eta_{j-1}\right)\left(\eta_{j+1/2}-\eta_{j-1/2}\right)}
\end{aligned}
$$

$$A_6 = \frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{i-1/2}}{J_{i-1/2}\left(\eta_{i-1/2,j+1} - \eta_{i-1/2,j-1}\right)\left(\xi_{i+1/2} - \xi_{i-1/2}\right)} + \frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{j-1/2}}{J_{j-1/2}\left(\xi_{i+1,j-1/2} - \xi_{i-1,j-1/2}\right)\left(\eta_{j+1/2} - \eta_{j-1/2}\right)}\right]$$

$$A_7 = -\frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{j+1/2}}{J_{j+1/2}\left(\xi_{i+1,j+1/2} - \xi_{i-1,j+1/2}\right)\left(\eta_{j+1/2} - \eta_{j-1/2}\right)} - \frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{j-1/2}}{J_{j-1/2}\left(\xi_{i+1,j-1/2} - \xi_{i-1,j-1/2}\right)\left(\eta_{j+1/2} - \eta_{j-1/2}\right)}\right] + \frac{\left(x_\eta^2 + y_\eta^2\right)_{i-1/2}}{J_{i-1/2}\left(\xi_i - \xi_{i-1}\right)\left(\xi_{i+1/2} - \xi_{i-1/2}\right)}$$

$$A_8 = -\frac{1}{2}\left[\frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{i-1/2}}{J_{i-1/2}\left(\eta_{i-1/2,j+1} - \eta_{i-1/2,j-1}\right)\left(\xi_{i+1/2} - \xi_{i-1/2}\right)} + \frac{\left(-x_\xi x_\eta - y_\xi y_\eta\right)\Big|_{j+1/2}}{J_{j+1/2}\left(\xi_{i+1,j+1/2} - \xi_{i-1,j+1/2}\right)\left(\eta_{j+1/2} - \eta_{j-1/2}\right)}\right]$$

$$
\begin{aligned}
A_9 \;=\; -\Bigg[& \frac{\left(x_\eta^2 + y_\eta^2\right)\Big|_{i+1/2}}{J_{i+1/2}\left(\xi_{i+1} - \xi_i\right)\left(\xi_{i+1/2} - \xi_{i-1/2}\right)} \\
& + \frac{\left(x_\eta^2 + y_\eta^2\right)\Big|_{i-1/2}}{J_{i-1/2}\left(\xi_i - \xi_{i-1}\right)\left(\xi_{i+1/2} - \xi_{i-1/2}\right)} \\
& + \frac{\left(x_\xi^2 + y_\xi^2\right)\Big|_{j+1/2}}{J_{j+1/2}\left(\eta_{j+1} - \eta_j\right)\left(\eta_{j+1/2} - \eta_{j-1/2}\right)} \\
& + \frac{\left(x_\xi^2 + y_\xi^2\right)\Big|_{j-1/2}}{J_{j-1/2}\left(\eta_j - \eta_{j-1}\right)\left(\eta_{j+1/2} - \eta_{j-1/2}\right)} \Bigg]
\end{aligned}
$$

APPENDIX B. THE COLLECTION OF STRETCHED MESHES GENERATED FOR SOLVING LAPLACE'S EQUATION

As described in Chapter 3, a series of stretched meshes were generated for examining the performance of different restriction and prolongation operators when solving Laplace's equation. This appendix contains figures of several of the meshes used in the comparison from a cell area ratio of 0.80 through 1.25.

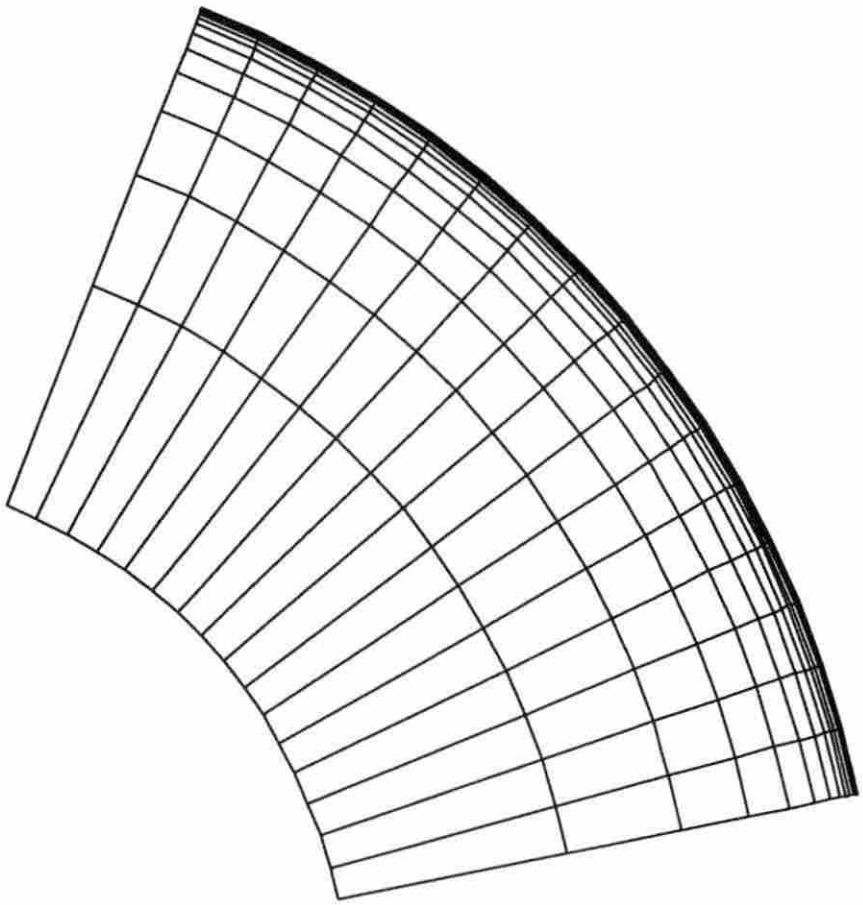

Figure B.1: A 17x17 grid with a cell area ratio of 0.80

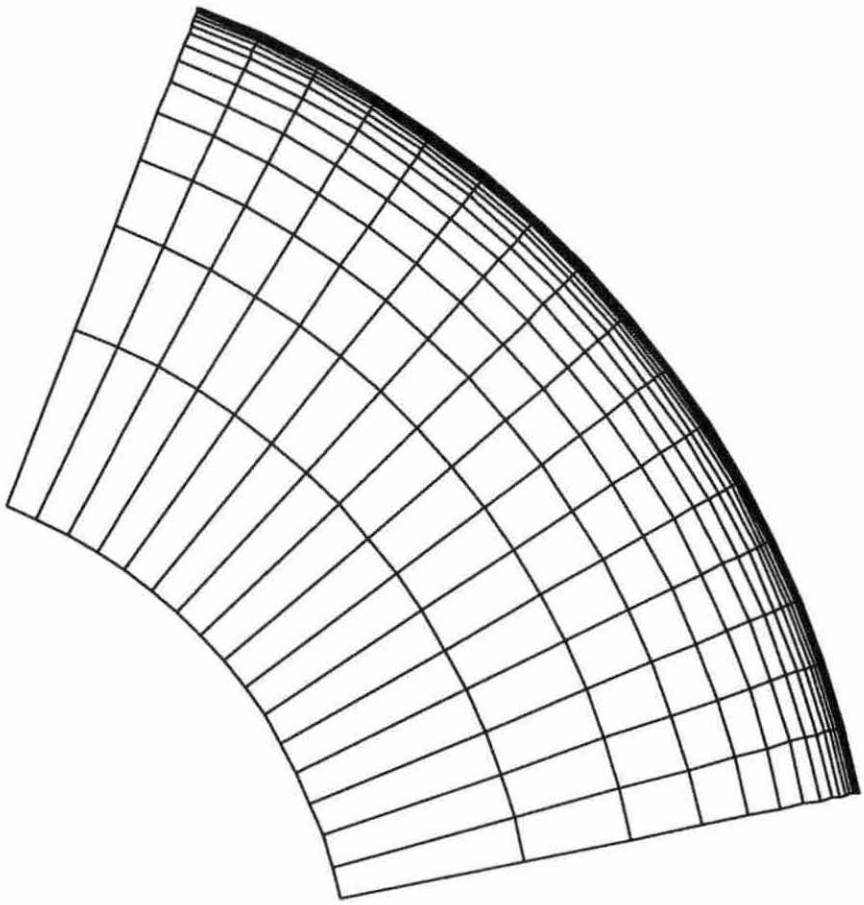

Figure B.2: A 17x17 grid with a cell area ratio of 0.85

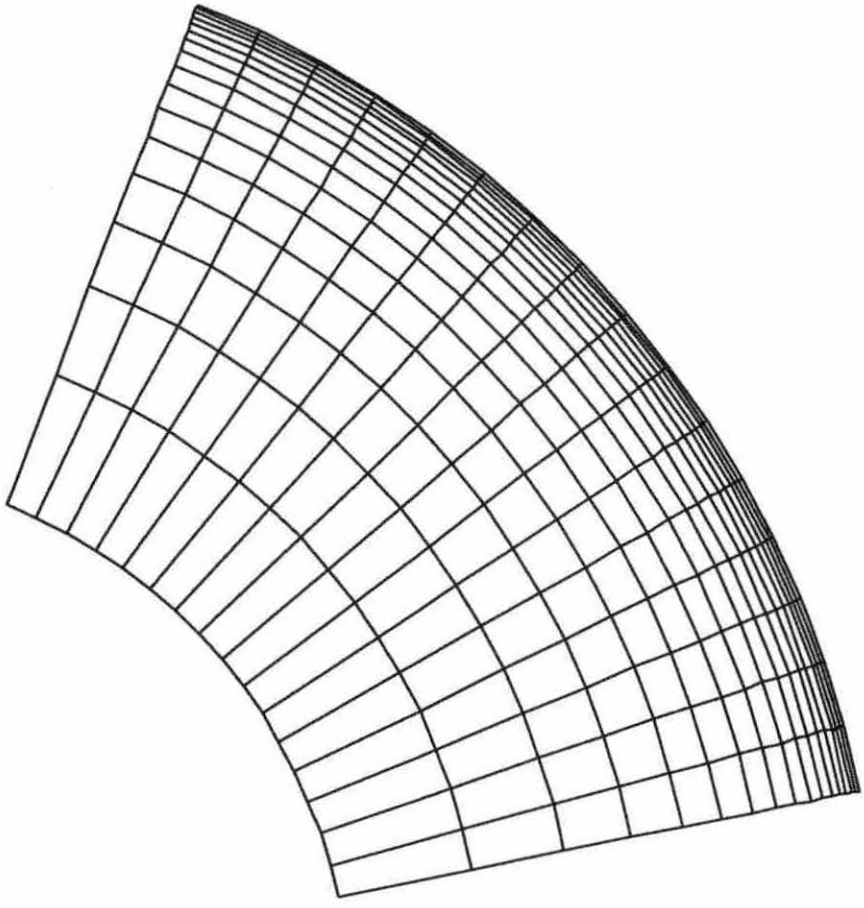

Figure B.3: A 17x17 grid with a cell area ratio of 0.90

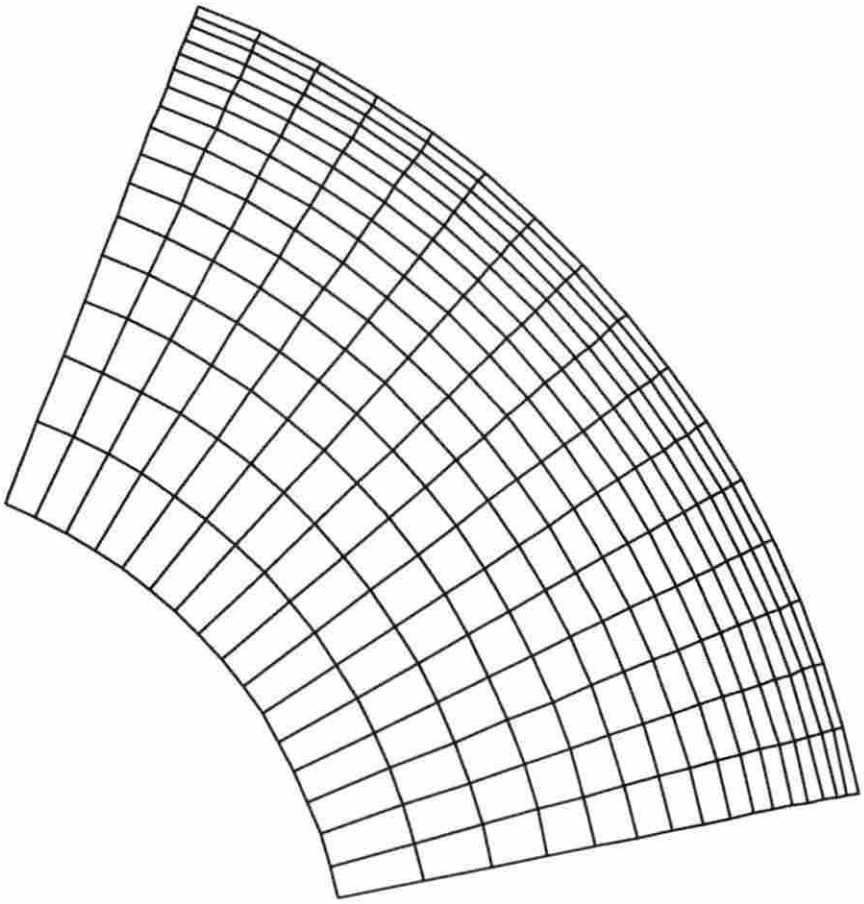

Figure B.4: A 17x17 grid with a cell area ratio of 0.95

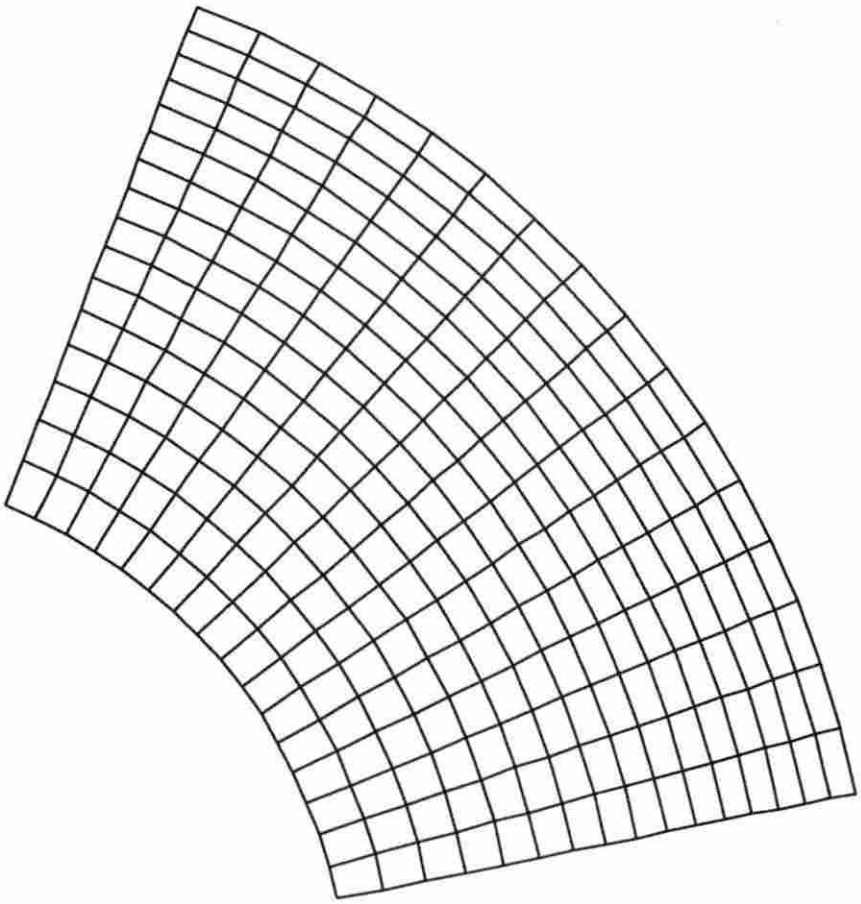

Figure B.5: A 17x17 grid with a cell area ratio of 1.00

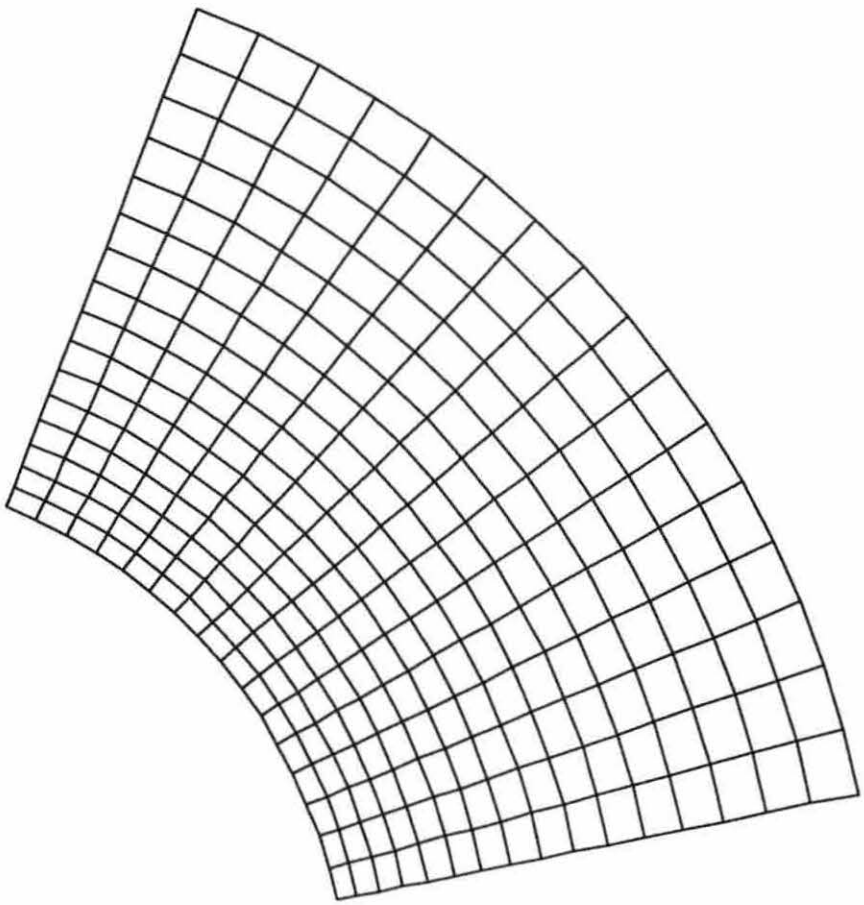

Figure B.6: A 17x17 grid with a cell area ratio of 1.05

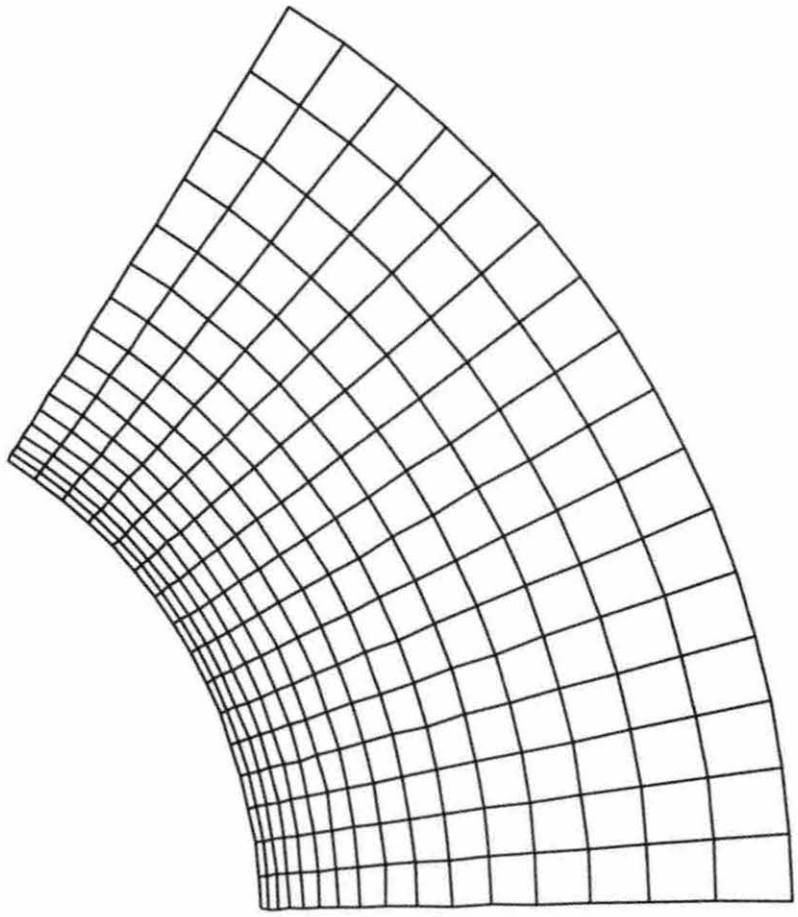

Figure B.7: A 17x17 grid with a cell area ratio of 1.10

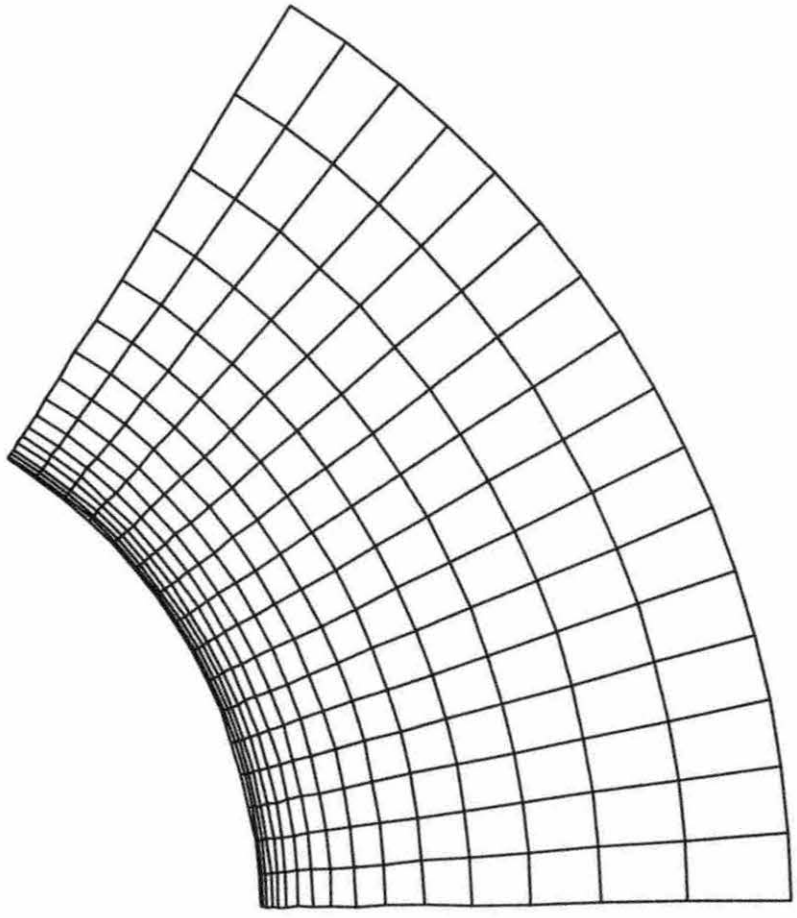

Figure B.8: A 17x17 grid with a cell area ratio of 1.15

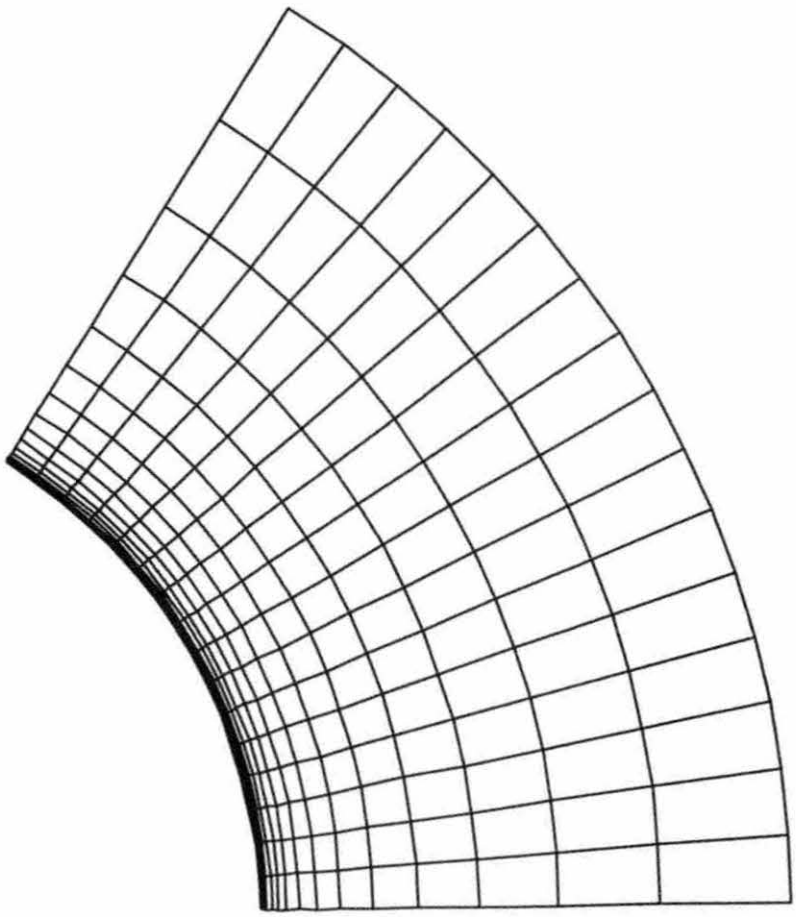

Figure B.9: A 17x17 grid with a cell area ratio of 1.20

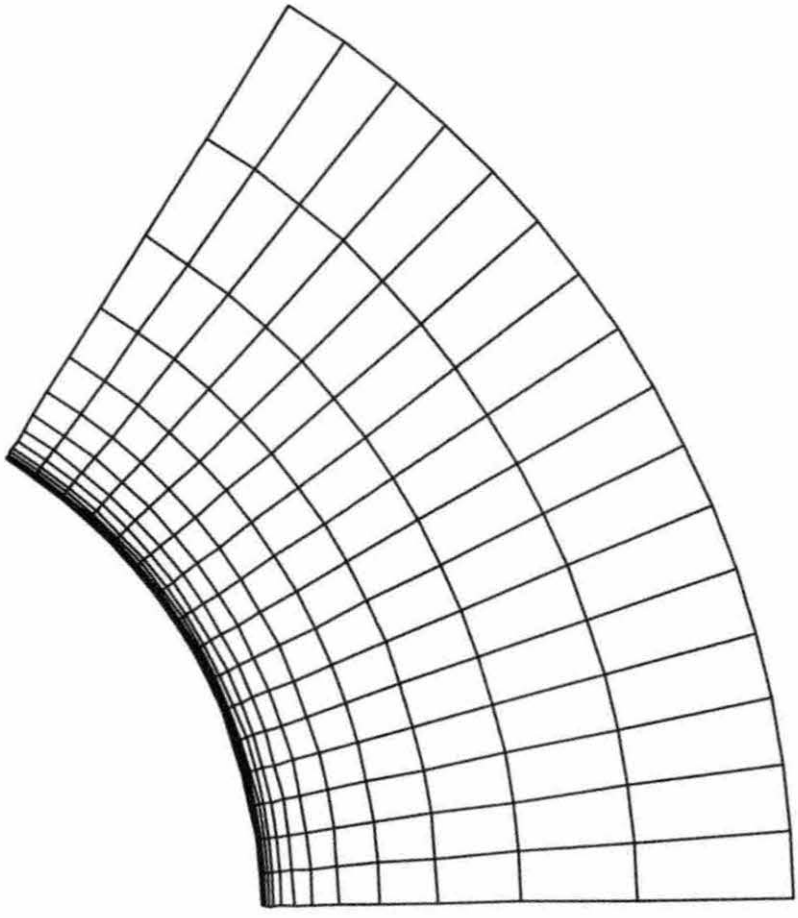

Figure B.10: A 17x17 grid with a cell area ratio of 1.25