ADAPTIVE FINITE ELEMENT SIMULATION OF INCOMPRESSIBLE VISCOUS FLOW

by

ROBERT MILLER FITHEN

Dissertation submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirments
for the degree of

DOCTOR OF PHILOSOPHY

in

Engineering Mechanics

Approved by:

J.N. Reddy

J. N. Reddy
(Chairman of Committee)

Saad Ragab

S. A. Ragab
(Member)

[illegible]

M. S. Cramer
(Member)

[illegible]

B. Grossman
(Member)

[illegible]

E. G. Henneke II
(Member)

August 1993

LD
5655
V856
1993
F573
C.2

C.2

ADAPTIVE FINITE ELEMENT SIMULATION OF INCOMPRESSIBLE VISCOUS FLOW

by

Robert Miller Fithen

Chairman of Advisory Committee: J. N. Reddy

Engineering Science and Mechanics

ABSTRACT

A finite element method is employed for solving two– and three–dimensional incompressible flows. The formulation is based on a segregated solution method. In this segregated formulation, the velocities and pressures are uncoupled and the equations for each are solved one after the other. This segregated solution method is numerically compared to the penalty method and to previous reported data to determine its validity. Next an iterative solution method which employs an element–by–element data structure of the finite element method is developed. Two types of iterative methods are used. For a symmetric stiffness matrix, the conjugate gradient method is used. For an unsymmetric stiffness matrix, the bi–conjugate gradient method is used. Both iterative solution methods make use of a diagonal preconditioning method (Jacobi preconditioning). Several problems are solved using this segregated method. In two–dimensions, flow over a backward facing step and flow in a cavity are investigated. In three–dimensions, the problems include flow in a cavity at Reynolds number 100 and 1000, and flow in a curved duct. The simulation compares very well with previously reported data, where available.

This segregated solution method is combined with hierarchical basis functions in order to develop a p–adaptive solution method. These basis functions are based on Chebyshev polynomials and extend up to fifth order. An error estimate is developed based on the difference between two different solutions at varying polynomial orders. This error estimate is used to spatially converge a solution for two–dimensional flow over a backward facing step. As the error indicator decreases to zero, the numerical data obtained from the finite element simulation compares favorably with experimental data. This p–adaptive solution method is then extended to three–dimensional problems.

Flow over a three–dimensional backward facing step is then simulated. As hoped, the error indicator correctly shows which elements in the grid need further enrichment. Several weakness of the adaptive methodology are shown through the use of this example. From the weakness of the adaptive methodology, new areas of possible research present themselves. Several suggestions as to possible solutions to these weakness are given in the conclusions of this work.

Contents

List of Figures

List of Tables

Nomenclature

A_{ij}	Advection matrix
B^{ij}	Submatrix for the penalty formulation
$\{b\}$	Right hand side vector
d_k	Direction vector
f	Body forces
H_{ij}	Hessian matrix
$I\,()$	General functional
$[K]$	System matrix
K_{ij}	Stiffness matrix
$[M]$	One sided preconditioner matrix (also mass matrix)
P	Pressure
$[Q]$	Two sided preconditioner matrix
S^{ij}	General submatrix
t	time
u_i	Velocity components
$\{x\}$	Solution vector

Greek Symbols

ϵ_{ij}	Strain tensor
Γ	Boundary of the fluid domain
γ	Penalty parameter

λ_k	k^{th} Eigenvalue
μ	Viscosity
Ω	Fluid domain
Ψ_i	i^{th} basis function
ρ	Density
σ_{ij}	Stress tensor

Chapter 1

Introduction

1.1 Motivation

Computational Fluid Dynamics (CFD) is now accepted as a useful quantitative and qualitative design tool. Within the aerospace industry CFD has become very prevalent, replacing water or wind tunnel experiments with a sort of numerical wind tunnel. This transition is due to several things; relative cost, setup time, and a wealth of information provided by the CFD solutions. However, engineers who do not possess extensive numerical expertise tend to assume that an aesthetically pleasing solution is accurate and even exact. Such assumptions can lead to catastrophic results in the final design. Researchers recognize this dilemma and have attempted to address the problem by developing many different forms of adaptive algorithms.

The goal of adaptive algorithms is very simple, alter the mesh in such a way as to provide a solution whose error is below a predetermined level. This error can be approximated in several ways. One of the most common ways of determining the error is based on the assumption that an exact solution will be obtained on a refined mesh. The difference between the fine grid solution and the coarse grid solution is assumed to be the error. As long as the numerical scheme is consistent, this assumption of error analysis is valid.

Adaptive methods appear in the literature in basically three forms; r-methods, h-methods, and p-methods. R-methods move existing grid points around in the domain to obtain a more accurate solution. The h-methods simply add new nodes to the mesh. The p-methods maintain the existing mesh and increase the polynomial order of the approximation. These adaptive algorithms are based on increasing the subspace of admissible functions used to represent the numerical solution. This increase in the subspace of admissible functions is closely related to a Fourier series solution of a partial differential equation. As with the Fourier series solution, extending the solution subspace to include more admissible functions insures the solution will approach the exact solution in the limit. Use of these r, p, and h adaptive schemes have shown improvements in solution accuracy.

While each of the three adaptive schemes increase the accuracy of the solution, there are several differences to each method. These differences are best exposed by trying to solve developed flow in a pipe where the exact solution is quadratic in r, and constant in the direction of the flow. The r-method is the most computationally efficient, in that the data structure remains intact. However, this method will only provide a small increase in accuracy. If linear elements are used in the h-method, the number of elements must go to infinity to obtain the exact solution. In this developed flow example, one element of second order could obtain the exact solution. Consequently, the p-adaptive approach will obtain the exact solution where the r- and h-adaptive methods have failed.

While the p-method looks like the answer to any adaptive approach, it is not without problems. The element matrices developed in the p-method become very large; this creates a problem as the bandwidth of the global matrices become large and solution efficiency deteriorates. Also, as the polynomial order increases, the numerical integration time required for each element becomes very costly. Another problem with p-methods can be seen when a discontinuity exists within the domain, the p-method then shows a Gibbs like phenomenon around the discontinuity. All of these problems with the p-method have spawned off new research in the areas of shape function selection, iterative solvers, and combining p-methods with r- or h- methods.

1.2 Present Research

The focus of the present research is the development of a three-dimensional p-adaptive capability, in incompressible flows. A computational tool with this capability could be used in many situations to develop an accurate solution for internal and external flows.

In Chapter 2, a literature review is performed. The purpose of the literature review is to obtain an understanding of the current state of the adaptive finite element method that is applied to fluid mechanics problems. The literature review will also place the current work in a proper perspective by elaborating on gaps and/or future trends in the current state of the art. This literature review takes place in several phases. The first phase examines the variational formulation for incompressible flow. Secondly, the solution methods are covered, including both direct and iterative methods. Next, the hierarchical element formulation is covered including the p-adaptive methods. And finally, error analysis within the context of hierarchical element formulations is covered.

In Chapter 3, the actual problem is formulated with both segregated and penalty formulation. A computational comparison is drawn between the two methods, by using a banded solution method. These formulations are for moderate Reynolds number using time dependent analysis to obtain the steady state response. This analysis takes place for two-dimensional structured grids. A parametric study is done to show how the bandwidth effects the solution time of both formulations.

In Chapter 4, a comparison between the frontal and iterative solution methods is made. Several three-dimensional examples are solved using this iterative method. These examples include three-dimensional cavity flow, and three-dimensional flow through a curved duct.

In Chapter 5, the segregated formulation is extended to include p-adaptive elements. The p-adaptive elements are based on a hierarchical element formulation. Using a simple error estimate, the solution is carried out until a specified tolerance is reached within each element. Several examples are solved using this p-adaptive element formulation. The examples include two and three-dimensional flow over a backward facing step.

Chapter 6 presents a discussion and conclusions. This chapter elaborates on the strengths and weakness of the p-adaptive methodology. This chapter also suggests directions for future work in this area.

Chapter 2

Literature Review

2.1 Remarks

In recent years an enormous amount of work has been done in the area of computational fluid dynamics (CFD). The scope of this work is varied in the assumptions used to obtain the governing differential equations. Depending on the preconceived knowledge of the flow, one can narrow the applicability of the solution to include only the desired effects. For example, if one were interested in very slow flow, a Stokes flow simulation would be in order. For this reason, the scope of the present work must be well defined in order to properly assess the current state of the art within that sub-area of CFD. The extent of the present work involves the steady state solution of the incompressible Navier-Stokes equations in two and three-dimensions. The numerical approach contains a hierarchical p-adaptive finite element formulation. Only work directly related to the topic at hand is included for review in this chapter.

2.2 Finite Element Formulation

Simulating incompressible flows with the finite element method has proven an enormous challenge over the years. The major difficulty to such an endeavor is the actual form of the continuity equation. The reason for such difficulty is the absence of a pressure term in the continuity equation. Hence, the continuity equation is viewed as a constraint applied to the momentum equations.

One of the most common methods used in finite elements for simulation of incompressible fluid flow is the penalty method. Some of the earliest work in the penalty method was carried out by Babuski [1, 2], Reddy [3, 4], Hughes [5], and Marshall [6].

Another common method used to solve incompressible flows involves the replacement of the continuity equations with the Poisson equation for pressure. This type of approach was first used by Harlow and Welch [7] in conjunction with the marker in cell (MAC) method. Chorin [8] developed a similar approach to solve the incompressible Navier-Stokes equations. Chorin's [8] method, called pressure projection, seeks a velocity field which satisfies the continuity constraint at the end of each step. The work of Harlow and Welch [7] and Chorin [8] spurred a large quantity of work within the finite difference, finite element, and control volume communities. The complete description of these methods, which are still in prominent use today, are given by Patankar [9]. An excellent display of the capability of pressure projection methods is shown by Patankar *et. al* [10]. The pressure projection and Poisson equation method are still used today by the finite difference, element, and control volume communities. Comini and Del Giudice [11, 12] used the ideas of Chorin and Patanker to develop a finite element method using a pressure correction scheme. The scheme is based on a pseudo-transient method for obtaining the steady state solution to the incompressible Navier-Stokes equations. Kim and Chung [13] used this method to predict the flow of turbulent diffusion flames in two-dimensions. Sohn, Kim, and Chung [14] performed a comparison between the segregated approach of Comini and Del Giudice, and the standard mixed approach [15] within the finite element context. Their analysis showed that the segregated approach required more iteration to convergence. However, there is no mention of the respective times required for obtaining each solution.

To date, there have been no computational comparisons of the segregated and penalty methods for the solution of incompressible flow. Although the comparison of Sohn, Kim, and Chung [14] has shown an advantage of segregated over mixed methods, the penalty formulation requires less computational effort than the mixed methods due to the problem size reduction [15]. A comparison would seek to answer the following questions: Does the segregated approach hold any advantage over the penalty method for the solution of the incompressible Navier-Stokes equations? And if so, under what conditions? Finally, there are no published works using the segregated methods within a three-dimensional context. There are no formulation problems either in using this method in three-dimensions. However, it may be that numerical limitations exist.

2.3 Solution Methods

Through the use of finite element methods, a continuous system is broken into a discrete system by using a variational statement. This process yields a system of equations of the form:

$$[K]\{x\} = \{b\}\,. \tag{2.1}$$

In the early years of the finite element method, it was known that the form of the system matrix $[K]$ played a significant role in the solution time needed for a particular problem. Approximately 70 to 90 % of the required computer time to obtain a solution is taken by an equation solver. This spawned a large amount of research into the solution of systems of equations. Some of the research was devoted to examining the structure of the system matrix $[K]$ in order to increase computational speed. This was accomplished by using banded solution schemes and nodal renumbering algorithms. Other work focused on the task of finding a solution method which would drop the operation count. The earliest success was obtained using a combination of these methods.

2.3.1 Direct Solution Methods

Direct solution methods have developed into three distinct categories. These three categories commonly used in the finite element method are banded methods, profile or envelope methods, and frontal methods. The difference between these methods is primarily that of data structure and implementation. Each of the methods are reviewed briefly with explanations.

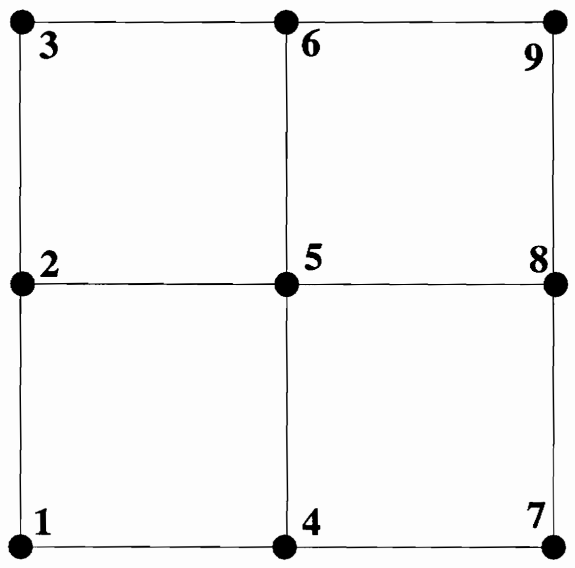

Figure 2.1: Example grid

Banded Solution Methods

Banded solution methods take advantage of the sparseness of the system matrices. For any row in the matrix, only zeros will exist a certain distance from the main diagonal. For example, if a grid as shown in Figure 2.1 were generated, the corresponding global stiffness matrix would become:

$$\begin{bmatrix} k_{11} & k_{12} & 0 & k_{14} & k_{15} & 0 & 0 & 0 & 0 \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & 0 & 0 & 0 \\ 0 & k_{32} & k_{33} & 0 & k_{35} & k_{36} & 0 & 0 & 0 \\ k_{41} & k_{42} & 0 & k_{44} & k_{45} & 0 & k_{47} & k_{48} & 0 \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} & k_{57} & k_{58} & k_{59} \\ 0 & k_{62} & k_{63} & 0 & k_{65} & k_{66} & 0 & k_{68} & k_{69} \\ 0 & 0 & 0 & k_{74} & k_{75} & 0 & k_{77} & k_{78} & 0 \\ 0 & 0 & 0 & k_{84} & k_{85} & k_{86} & k_{87} & k_{88} & k_{89} \\ 0 & 0 & 0 & 0 & k_{95} & k_{96} & 0 & k_{98} & k_{99} \end{bmatrix} . \tag{2.2}$$

The half bandwidth of such a system is four. For the routines **SGBCO** and **SGBFA** within LINPACK [16], the storage of this matrix would be:

$$\begin{bmatrix} * & * & * & * & k_{15} & k_{26} & 0 & k_{48} & k_{59} \\ * & * & * & k_{14} & k_{25} & k_{36} & k_{47} & k_{58} & k_{69} \\ * & * & 0 & k_{24} & k_{35} & 0 & k_{57} & k_{68} & 0 \\ * & k_{12} & k_{23} & 0 & k_{45} & k_{56} & 0 & k_{78} & k_{89} \\ k_{11} & k_{22} & k_{33} & k_{44} & k_{55} & k_{66} & k_{77} & k_{88} & k_{99} \\ k_{21} & k_{32} & 0 & k_{54} & k_{65} & 0 & k_{87} & k_{98} & * \\ 0 & k_{42} & k_{53} & 0 & k_{75} & k_{86} & 0 & * & * \\ k_{41} & k_{52} & k_{63} & k_{74} & k_{85} & k_{96} & * & * & * \\ k_{51} & k_{62} & 0 & k_{84} & k_{95} & * & * & * & * \end{bmatrix} . \tag{2.3}$$

This storage method usually requires less memory than the complete matrix. The only exception to this rule is for matrices with bandwidths of comparable size to the overall matrix dimensions.

LINPACK has developed into a standard by which other solution packages are judged [17, 18]. LINPACK has also become a standard in benchmark-

Table 2.1: LINPACK performance on a 100 x 100 matrix

Computer	**MFLOPS**
CRAY XMP	24
IBM 3090 200	6.8
CONVEX C-1	2.9
Alliant FX/8	2.5
IBM 370/195	2.5
IBM 3033	1.7
NeXTstation	1.0
DEC 8650	0.96
VAX 11/780 FPA	0.13

ing computer systems [19]. In fact, the LINPACK computer benchmarking method has become so prevalent, that at the present time (1993) most computer venders use this benchmark in their advertising. A partial listing of Dongarra's [19] benchmarking report is included in Table 2.1. The NeXT computer speed rating is included due to its use throughout this dissertation [20]. Several words of caution must be stated for this comparison. The LINPACK routines used in the comparison are **DGEFA** and **DGESL** and the matrix size is 100 by 100. However, in the 300 by 300 matrix in Table 2.2 there is a significant increase in performance in vector machines and very small changes in the non-vector machines. From this, one may conclude that even though the NeXT computer is close to the performance of some

Table 2.2: LINPACK performance on a 300 x 300 matrix

Computer	**MFLOPS**
CRAY XMP	257
IBM 3090 200	18
CONVEX C-1	8.7
Alliant FX/8	6.9
IBM 370/195	4.4
IBM 3033	2.5
VAX 11/780 FPA	0.11

high performance computers for low matrix size, it will be severely lagging in performance for large matrix operations.

The LINPACK routines use a set of low level routines called BLAS [21], an acronym for Basic Linear Algebra Subroutines. These subroutines are very efficient due to their ability to perform operations on multi dimensional arrays by using single dimension arrays. These BLAS routines may be replaced with machine code versions on any specific machine in order to increase computational speed.

Envelope Solution Methods

Although all the same, envelope solution methods go by several names: profile, skyline, pipe, and variable bandwidth. This method makes use of one fact; the bandwidth for any row may be different than the bandwidth of the system. In the example shown in Figure 2.1 the halfbandwidth is 4 for all rows except for rows 7 and 3 where the halfbandwidth is 3. This can be seen from equation 2.3 where a zero appears in column 7 of row 1, and column 3 of row 3. The skyline method uses this variable bandwidth in its storage and solution method. A full explanation can be found in [22]. This method will not be used in the present work.

Frontal Solution Methods

The development of the frontal solution method was spurred by the need to solve large scale problems on small memory computer systems. Irons [23] introduced the method in 1970. Hood [24] developed a frontal program for solving unsymmetric matrices in 1976. Since 1976, several refined versions of the program have appeared [25, 26]. The frontal method is very simple in concept, its goal is the removal of any assembled degree of freedom as soon as possible. The removed degrees of freedom are reduced and written to disk storage for later back substitution. The equations held in memory during this process are called the frontal equations. The basic algorithm loops through the elements, from first to last, assemblying only the portion of the global matrix associated with the present element. Next, the algorithm looks for any completely assembled node. If an assembled node is found, that node will be reduced and the equation related to the node will be written to disk making room for another equation in the front. At the end of the element loop, the algorithm goes through the equations previously written to disk and performs a back substitution (the final phase of Gauss elimination). A

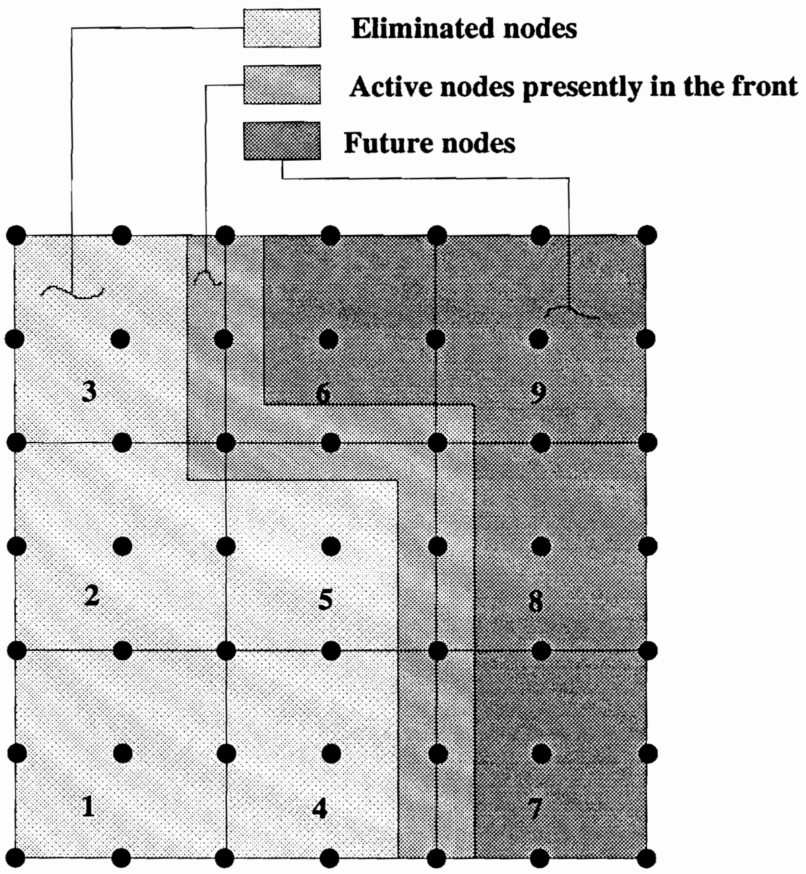

Figure 2.2: Front and element numbering

graphical representation is shown in Figure 2.2. Here the assemble process is currently in the element loop at element number 5. All nodes which have been completely assembled will be eliminated (eliminated nodes). All nodes which presently reside within the core of memory (active nodes presently in the front) have not yet been completely assembled. Future nodes have not yet been included, but will be as the element number in the loop increases.

2.3.2 Nodal Renumbering

In an attempt to reduce the operation count early researchers developed nodal renumbering schemes which produced a lower bandwidth and decreased the solution time. Work began in the sixties to approach this problem when Rosen [27] proposed a method for row column interchange to reduce the bandwidth. This row and column interchange originated from the simple fact that exchange of two nodes in any given mesh simply results in the exchange of relevant rows and columns in the global stiffness matrix. Later the Cuthill–McKee algorithm [28, 29] used graph theory to obtain a reduction in bandwidth. As solution algorithms began to change, from banded methods to profile methods, emphasis in renumbering methods changed also. It was found that a simple change to the Cuthill–McKee algorithm could improve a matrix profile without hurting its bandwidth [30]. This method was called Reverse–Cuthill–McKee algorithm because the node numbering scheme was simply reversed. Others [31, 32, 33] have developed an improved method of node and element renumbering directed at a profile frontal minimization. At the present, no method has been developed which can be shown to substantially reduce the bandwidth or profile over that produced by the Cuthill–McKee and Reverse–Cuthill–McKee algorithms.

2.3.3 Iterative Solution Methods

There is a large quantity of literature available on the iterative solution to the set of equations represented by equation 2.1. The methods of primary interest in the present study are the gradient methods. These methods can be understood through use of optimization methods [34, 35].

In finite element analysis, a continuous system is made discrete through use of a limited number of independent functional representations over the given domain. This yields a functional of the form:

$$I(x_i) = \frac{1}{2} x_i K_{ij} x_j - b_i x_i \tag{2.4}$$

After application of the boundary conditions this quadratic form becomes positive definite. This functional reaches a minimum when its first variation becomes zero.

$$\frac{\partial I(x_i)}{\partial x_k} = 0 \tag{2.5}$$

Combining these two equations yields a set of N equations and N unknowns.

$$K_{ij} x_j = b_i \tag{2.6}$$

Within the gradient method the solution to these N equations is sought by minimization of the original problem statement, $x_j = X_j$ where,

$$I(X_i + \lambda d_i) \geq I(X_i) \qquad \forall\ \lambda d_i \tag{2.7}$$

The generalized N dimensional minimization problem can be solved by the following general steps.

1. Guess an initial position vector x_i.

2. Use an appropriate methodology to find a new direction vector d_i.

3. Use an appropriate 1-dimensional minimization method to minimize $I(x_i + \lambda d_i)$ with respect to λ.

4. Replace position x_i with $x_i + \lambda d_i$

5. Check for convergence; if not converged, return to step 2.

One of the original methods proposed in 1847 by Cauchy [36] is the method of steepest descent. In the method of steepest descent the direction vector d_i is chosen as follows:

$$d_k = -\frac{\partial I}{\partial x_k}. \tag{2.8}$$

Which means d_k is in the direction of steepest descent of I at point x_k. Next λ is found such that $I(x_i + \lambda d_i)$ is minimized by:

$$\frac{\partial I(x_i + \lambda d_i)}{\partial \lambda} = 0. \tag{2.9}$$

An example of this method is given for a two by two matrix by

$$I(x,y) = \frac{1}{2}\{x \quad y\}\begin{bmatrix} 8 & -4 \\ -4 & 8 \end{bmatrix}\begin{Bmatrix} x \\ y \end{Bmatrix} - \{x \quad y\}\begin{Bmatrix} 16 \\ 4 \end{Bmatrix}. \tag{2.10}$$

If the starting point of (-1,0) is used, then $d_k = (24, 0)$ and a graphical representation of the process is shown in Figure 2.3. One particularly disappointing feature of this method can be observed in Figure 2.3 where zigzagging takes place. As the difference between the highest and lowest eigenvalue of the matrix K_{ij} increases the zigzagging becomes more pronounced. As the eigenvalue spectrum spreads the contour plot becomes more elliptic (the difference between the major axes and minor axes increases). However, if the initial direction vector is chosen as an eigenvector this zigzagging will not occur.

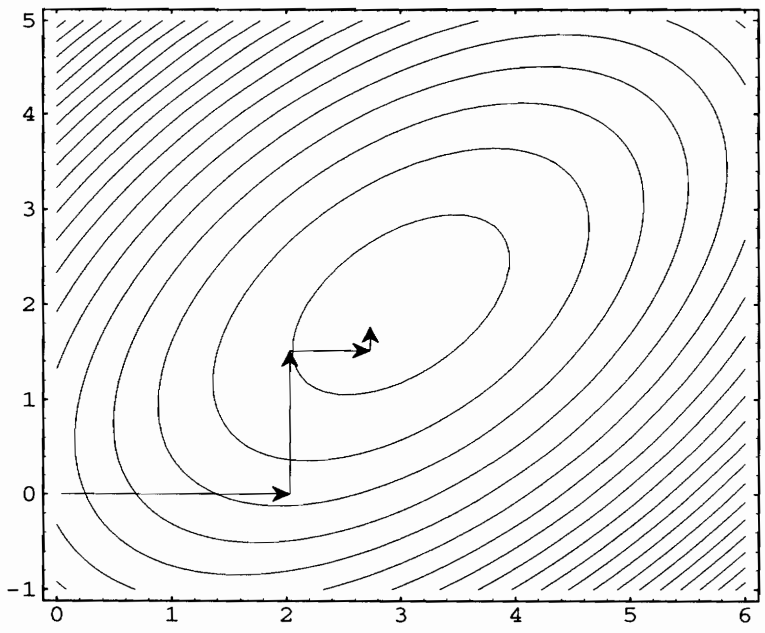

Figure 2.3: Illustration of steepest descent method

By requiring the d_k vectors to be H-conjugate we obtain the conjugate gradient method. The H-conjugate condition can be expressed as: $d_i H_{ij} d_j = 0$ for $i \neq j$. Here,

$$H_{ij} = \frac{\partial^2 I}{\partial x_i \partial x_j} \tag{2.11}$$

is called the Hessian matrix. Within the context of a quadratic functional, the Hessian matrix is simply the stiffness matrix ($H_{ij} = K_{ij}$). Using the same example as before with the same starting point and direction, we require the second direction vector to satisfy:

$$0 = \{ x \quad y \} \begin{bmatrix} 8 & -4 \\ -4 & 8 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}. \tag{2.12}$$

Hence, $d_k = (1,2)$, as shown in Figure 2.4.

The use of conjugate gradient method for an N dimensional quadratic form results in convergence to the exact solution in N iterations or less (only if infinite precision is used) [37]. A complete description of these methods can be found in the literature by Jennings [37], and Golub and Van Loan [18]. Most conjugate gradient methods are related to the methods of Hestenes and Stiefel [38] which were developed in 1952. Since the time of Hestenes and Stiefel, a large quantity of work in the area has been performed. Even today this research area is very active. Fletcher, Reeves, Powell, and Davidson [39, 40] played vital rolls in the early development of the conjugate gradient methods. As a result, there are several methods which bear their names. Some of these methods are in wide use today. Fried [41] first suggested use of these gradient methods for solution of a finite element system matrix.

Jennings [42] showed a theoretical relation between the error reduction ϵ, minimum and maximum eigenvalues λ_1 and λ_n and the required iterations k

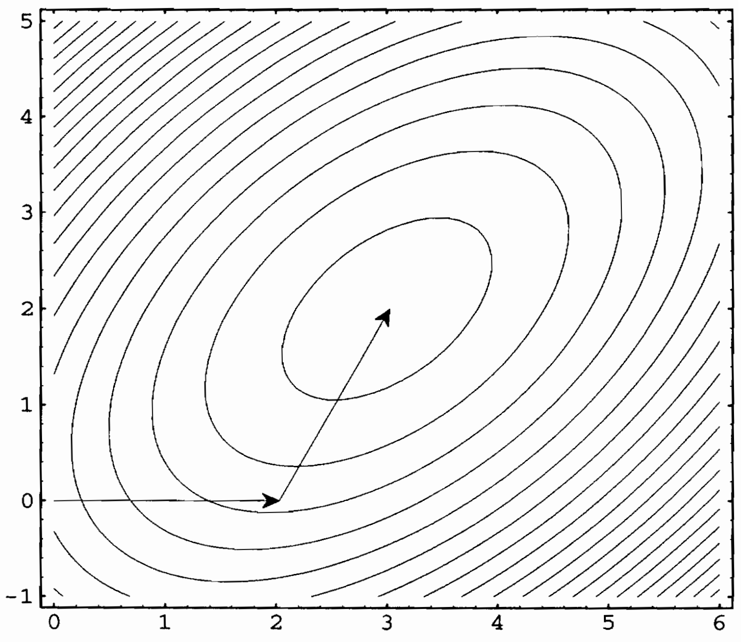

Figure 2.4: Illustration of conjugate gradient method

as

$$k \leq \sqrt{\left(\frac{\lambda_n}{\lambda_1}\right) \ln\left(\frac{2}{\epsilon}\right)} \tag{2.13}$$

Jennings described this equation as an upper bound for the convergence rate, since it is solely based on the extreme eigenvalues. Jennings also looked into the detail of preconditioning such that $\left(\frac{\lambda_n}{\lambda_1}\right)$ decreases, thereby increasing the convergence rate.

The subject of unsymmetric matrices is handled by premultiplying the system of equations by the transpose of the unsymmetric matrix. The ramifications of such a procedure are two fold; possible destruction of sparsity and decrease of the convergence rate. However, this method is very effective as shown by Prakhya [43]. Axelsson [44] addressed unsymmetric matrices by use of the Krylov sequence. Jea and Young [45] developed three forms of Lanczos algorithms by using the idealized generalized conjugate gradient methods. One of these Lanczos algorithms will be used in the present research work. At the present time, little work has been performed using any of these methods in the solution of the incompressible Navier-Stokes equations that use the segregated approach.

2.3.4 Preconditioning

As stated previously Jennings [42] showed a theoretical relation between the error reduction and condition number of the system matrix $\left(\frac{\lambda_n}{\lambda_1}\right)$. This gives rise to a simple question, can we change the system:

$$[K]\{x\} = \{b\} \tag{2.14}$$

in some way such that the condition number will decrease thereby increasing the convergence rate? The system is premultiplied by a preconditioner matrix $[M]$ as shown below:

$$[M]\,[K]\,\{x\} = [M]\,\{b\} \tag{2.15}$$

If $[M] = [K]^{-1}$ the condition number of the system matrix $[M]\,[K]$ is 1, and the conjugate gradient method will take only one iteration to obtain an exact solution (remember the equation given by Jennings is an upper bound on the convergence rate). One may conclude that the closer $[M]$ is to $[K]^{-1}$, the quicker the conjugate gradient method obtains a solution. Obtaining this matrix is the entire focus of the area of preconditioning. The basic idea is to perform some type of operations on the matrix $[K]$ in order to obtain the matrix $[M]$. This will usually be an incomplete or approximate inverse to $[K]$. One of the simplest methods is Jacobi preconditioning. In Jacobi preconditioning we alter the equation in the following way.

$$[B]\,\{y\} = \{d\} \tag{2.16}$$

Where,

$$[B] = [Q]^{-T}\,[K]\,[Q]^{-1} \tag{2.17}$$

$$\{y\} = [Q]\,\{x\} \tag{2.18}$$

$$\{d\} = [Q]^{-T}\,\{b\} \tag{2.19}$$

$$Q_{ij} = \sqrt{K_{(i)(i)}}\ \ \delta_{ij}. \tag{2.20}$$

This effectively places unity on the diagonal of $[B]$ and all non-diagonal terms in $[B]$ are less than one. Other forms of preconditioning include: incomplete

Cholesky, incomplete block, and domain decomposition methods [37, 46]. While these ideas are effective they will not be included in the present research. Only Jacobi preconditioning is used in the present work.

2.3.5 Element-by-Element Structure

One of the advantages of using the conjugate gradient methods comes from the construction of the global matrices themselves. In the finite element method, one constructs the global matrices by adding contributions from each individual element matrix into the global matrix.

$$[K] = \sum_{e=1}^{N} [K^e] \tag{2.21}$$

Matrix vector operation using this global matrix can be performed as shown below.

$$[K]\{u\} = \left(\sum_{e=1}^{N} [K^e]\right)\{u\} \tag{2.22}$$

Or keeping up with the indices,

$$\{v\} = [K]\{u\} = \left(\sum_{e=1}^{N} [K^e]\{u\}\right). \tag{2.23}$$

This equation implies a three level loop given by,

```
Do e = 1 to Nelements
  Do i = 1 to Nnodes per element
    Do j = 1 to Nnodes per element
      v(g(i)) = K(e,i,j) * u(g(j))
    EndDo j
```

```
        EndDo i
    EndDo e .
```

Here the g(i) vector is a steering vector used to direct the local node i to the global node g(i). By using the conjugate gradient method, the only required operation for equation solution is a matrix vector multiplication. Hence, the global matrices never need be assembled. With this in mind, an element by element solution scheme may be devised based on element matrix times global variable multiplication. A very important feature occurs within the conjugate gradient methods, these methods have no sensitivity to either nodal or element numbering methods. Remember, the frontal method is very sensitive to element numbering, and profile or banded solution methods are very sensitive to nodal numbering methods. Conjugate gradient methods need no element or nodal reordering for efficient solution of the system matrices. Such a method is perfect for an adaptive method where new degrees of freedom are added at will. The frontal or profile solver would need a complete reordering of the elements or nodes to obtain an efficient solution scheme. While this can be done, if several levels of adaptation take place, the time required to reorder the nodes and elements would be very restrictive and perhaps impossible.

2.4 Hierarchical P-Element Formulation

Much attention has been given, as of late, to the accuracy of general numerical solutions. In order to assure a numerical solution is in fact accurate, one must have some other semi-exact solution in which a comparison can be made. If the numerical formulation is consistent, a solution on a fine mesh will be closer to the exact solution than that of a coarse mesh solution. With this in mind, a practicing engineer has the simple task of generating successively finer grids and obtaining solutions on these grids until the difference between two successive solutions is minimal. At this point the engineer can effectively use the solution as a design tool. Unfortunately, this process is often impossible due to time and financial constraints, and the engineer must sacrifice his professional reputation in the design process.

A relatively new finite element procedure has been developed to address the problem of solution accuracy. Instead of remeshing the domain successively, we allow each element to increase its number of degrees of freedom in some manner. This method will obtain solutions which approach the exact solution in the limit. Upon adding the new degrees of freedom, the existing basis functions are kept, and the solution from the previous step can conveniently be used as an initial guess for the next solution. If the element basis functions are constructed in this way, the matrix contributions for any previous degree of freedom are embedded within the present element matrices, and convergence to the exact solution is guaranteed through the inclusion principle (Meirovitch [47]). The p-element formulation seeks to increase the number of degrees of freedom by increasing the polynomial order while retaining existing basis functions. This type of finite element formulation has been termed hierarchical formulation. Zienkiewicz *et. al* introduced this concept as early as 1971 [48], but the concepts were not put into a useful form until 1976 by Peano *et. al* [49, 50]. By 1981, a theoretical basis for the p-element method was established by Babuska *et. al* [51, 52]. In 1984, Patera *et. al* produced a *spectral element* method for fluids [53, 54, 55].

There are many similarities between p-element and spectral element approaches. The spectral element methods use Chebyshev polynomials and collocation to develop the equations. By contrast, the p-element method is more general, allowing for any acceptable weighted residual method, any higher order polynomial type, and Gauss integration techniques.

Several problems presently exist within the frame work of p-element formulations. Originally, published results seemed to indicate hierarchical p-element formulation reduced the condition number of the global stiffness matrix, as compared to Lagrange isoparametric elements of the same order [56, 57, 58]. This being true, one may justifiably assume that even though the two element formulations Lagrange isoparametric and hierarchical p-element, are based on the same set of monomials from Pascals triangle, how one actually chooses the basis function plays a significant role in determining the condition number of the global stiffness and mass matrices [59]. Knowing these facts, one may conclude that not all p-element formulations produce well conditioned matrices. In fact, many times, p-element formulations give worse conditioned matrices than an equivalent Lagrange element formulation. In the present work, a p-element formulation will be developed in both two and three-dimensions. At the present time, there have been very few usages of p-element formulations in incompressible flows, and no usage of the p-element method for incompressible flows in three-dimensions. It is the goal of this research to step into this area by developing a three-dimensional capability using the p-element formulations for the simulation of three-dimensional incompressible flows.

2.5 A Posteriori Error Analysis

The use of hierarchical basis functions delivers another advantage in the area of error analysis. A typical numerical analyst will go through a simple process to obtain an accurate numerical solution. First obtaining a solution on a course grid, then a fine grid, and finally comparing the two. The analyst would quit when two subsequent solutions did not differ by a specified amount. During this process, it is implied that the latest solution is very close to the exact solution, and if a consistent numerical scheme is devised this assumption is valid. If this process is carried out using hierarchical elements, then

$$T = \sum_{i=1}^{4} \Psi_i T_i \tag{2.24}$$

$$T^h = \sum_{i=1}^{4} \Psi_i T_i^h + \sum_{i=5}^{n} \Phi_i \zeta_i, \tag{2.25}$$

where T is the solution to the standard finite element problem, and T^h is the solution using hierarchical element enrichment. The difference between the two solutions is

$$\mathcal{E} = T^h - T = \sum_{i=1}^{4} \Psi_i \left(T_i^h - T_i \right) + \sum_{i=5}^{n} \Phi_i \zeta_i. \tag{2.26}$$

Hence the difference between the two solutions can be observed by simply plotting $\mathcal{E}$ over the domain. This plot can give guidance as to where in the domain further refinements need to take place.

2.6 A Priori Error Analysis

In a posteriori error analysis it is implied that the error between the two most recent solutions gives a measure as to future error trends. Although this may be true, we seek to know future trends in the error without knowledge of the past error trends (a priori). This can be done within the context of hierarchical finite elements, by obtaining an incomplete solution to the fine grid solution and comparing this solution to the coarse solution.

2.7 Contributions

In this section an enumeration of possible contributions is made. In the work of Comini and others [11, 12, 13, 14] only two-dimensional flow is considered. In fact, at this time no three-dimensional analysis using this particular formulation has appeared in the literature. This is not due to any formulation difficulties, so the transition to three-dimensions should be straight forward. No work has been performed using this formulation in conjunction with p-adaptive procedures (in two or three dimensions). In the literature for this particular formulation, no iterative solution methods have been used, only direct inversion methods. The present work seeks to fill these gaps in the literature.

Chapter 3

Problem Formulation

3.1 Governing Equations

In order to obtain a set of governing equations, attention must be directed to a set of dynamical force balance equations. Using an Eulerian description of motion these equations can be written as shown below.

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = \rho f_i + \frac{\partial \sigma_{ji}}{\partial x_j} \tag{3.1}$$

Where *rhos* is the fluid density and u_i is the fluid velocity. If the fluid in question is homogeneous, isotropic, and Newtonian then the stress σ_{ji} can be written as a function of the strain rate by

$$\sigma_{ij} = -\left(p - \lambda \epsilon_{kk}\right) \delta_{ij} + 2\mu \epsilon_{ij}. \tag{3.2}$$

The strain rate ϵ_{ij} can be expressed as:

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{3.3}$$

The Eulerian form of the mass continuity equation can be expressed as shown below.

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \tag{3.4}$$

If the flow is incompressible then the co-moving derivative of the density ($\frac{D\rho}{Dt}$) is zero and the mass continuity equation is reduced to

$$\frac{\partial u_i}{\partial x_i} = \epsilon_{ii} = 0. \tag{3.5}$$

This situation occurs often in the physical world. Examples include; low speed flow of air and other gases, and the flow of water or other liquids. Using this form of the continuity equation, the constitutive equations may be reduced to:

$$\sigma_{ij} = -p\delta_{ij} + 2\mu\epsilon_{ij}. \tag{3.6}$$

Substituting the constitutive relations into the momentum equations for a constant property fluid ($\mu = C$) yields

$$\rho\frac{\partial u_i}{\partial t} + \rho u_j\frac{\partial u_i}{\partial x_j} = \rho f_i - \frac{\partial p}{\partial x_i} + \mu\frac{\partial^2 u_i}{\partial x_j \partial x_j}. \tag{3.7}$$

These two sets of equations, continuity and momentum, can be solved in a variety of ways. One approach is to solve the momentum equations subject to a constraint applied to the velocity field such that the mass continuity is satisfied. This constraint is actually applied through the pressure. By forcing the pressure to obtain the correct distribution over the domain, the velocities will satisfy both the momentum and mass continuity simultaneously.

3.1.1 Penalty Formulation

The penalty method seeks satisfaction of the continuity equation through an actual constraint applied to the momentum equations. If the pressure and velocity gradients are related by

$$p = -\gamma \frac{\partial u_k}{\partial x_k}. \qquad (3.8)$$

Where γ is an specified large number, the continuity equation will be satisfied in an approximate sense. In order to implement this idea a weighted residual statement of the momentum equations is sought.

$$\int_\Omega \delta u_i \left(\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} - \rho f_i - \frac{\partial \sigma_{ji}}{\partial x_j} \right) d\Omega = 0 \qquad (3.9)$$

Carrying out an integration by parts yields:

$$\int_\Omega \left[\delta u_i \left(\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} - \rho f_i \right) + \sigma_{ji} \frac{\partial \delta u_i}{\partial x_j} \right] d\Omega - \int_\Gamma \delta u_i \sigma_{ji} n_j d\Gamma = 0 \quad (3.10)$$

Substituting in the constitutive equations gives:

$$\int_\Omega \left[\delta u_i \left(\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} - \rho f_i \right) - p \frac{\partial \delta u_i}{\partial x_i} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \frac{\partial \delta u_i}{\partial x_j} \right] d\Omega$$
$$- \int_\Gamma \delta u_i \sigma_{ji} n_j d\Gamma = 0 \qquad (3.11)$$

Inserting the penalty pressure terms gives:

$$\int_\Omega \left[\delta u_i \left(\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} - \rho f_i \right) + \gamma \frac{\partial u_k}{\partial x_k} \frac{\partial \delta u_i}{\partial x_i} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \frac{\partial \delta u_i}{\partial x_j} \right] d\Omega$$
$$- \int_\Gamma \delta u_i \sigma_{ji} n_j d\Gamma = 0 \qquad (3.12)$$

In order to obtain reasonable approximations for the velocities, the penalty terms must be made singular through reduced integration methods. The main advantage of using the penalty method is the reduction in problem size. In the mixed method the velocities and pressures become the dependent variables whereas in the penalty method only the velocities become the dependent variables.

3.1.2 Segregated Formulation

The segregated approach to solving the Navier-Stokes equations treats the pressure-velocity coupling in a different way. The method presented and used in this work is that of Comini and Del Guidice [11, 12]. This method uses a pseudo transient approach to obtain the steady state solution. The method adjusts the pressure field at the end of each time step such that mass continuity is satisfied. The dependent variables are written as:

$$u_i = u_i^{'} + u_i^* \tag{3.13}$$

$$p = p^{'} + p^*. \tag{3.14}$$

The superscript (*) terms are estimated values, and the superscript ($^{'}$) terms are corrected values. Substituting these relations into the momentum equation, where u_i^n is the velocity at time level n, the following linearized momentum equations result:

$$\rho\frac{\partial u_i^*}{\partial t} + \rho\frac{\partial u_i^{'}}{\partial t} = -\rho u_j^n\frac{\partial u_i^*}{\partial x_j} - \rho u_j^n\frac{\partial u_i^{'}}{\partial x_j} + \rho f_i - \frac{\partial p^*}{\partial x_i} - \frac{\partial p^{'}}{\partial x_i} + \mu\frac{\partial^2 u_i^*}{\partial x_j \partial x_j} + \mu\frac{\partial^2 u_i^{'}}{\partial x_j \partial x_j} \tag{3.15}$$

Assuming the term $\frac{\partial u_i^{'}}{\partial x_j}$ approaches zero as the solution approaches convergence, the following momentum equations are obtained.

$$\rho\frac{\partial u_i^*}{\partial t} + \rho\frac{\partial u_i^{'}}{\partial t} \cong -\rho u_j^n\frac{\partial u_i^*}{\partial x_j} + \rho f_i - \frac{\partial p^*}{\partial x_i} - \frac{\partial p^{'}}{\partial x_i} + \mu\frac{\partial^2 u_i^*}{\partial x_j \partial x_j} \tag{3.16}$$

The fact that $u_i^{'}$ does not appear in these equations does not affect the final steady state solution to these equations, since all correction terms disappear in a converged solution. These equations are split into two solution steps as follows:

$$\rho\frac{\partial u_i^*}{\partial t} = -\rho u_j^n\frac{\partial u_i^*}{\partial x_j} + \rho f_i - \frac{\partial p^*}{\partial x_i} + \mu\frac{\partial^2 u_i^*}{\partial x_j \partial x_j} \tag{3.17}$$

$$\rho\frac{\partial u_i^{'}}{\partial t} = -\frac{\partial p^{'}}{\partial x_i} \tag{3.18}$$

Taking the divergence of the last of these equations yields:

$$\rho\frac{\partial}{\partial x_i}\frac{\partial u_i^{'}}{\partial t} = \rho\frac{\partial}{\partial t}\frac{\partial u_i^{'}}{\partial x_i} = -\frac{\partial^2 p^{'}}{\partial x_i \partial x_i} \tag{3.19}$$

The continuity constraint is enforced at each time step by setting the divergence of the final velocity to zero.

$$\frac{\partial D}{\partial t} \cong \frac{D^{n+1} - D^n}{\Delta t} = \frac{D^* + D^{'} - D^n}{\Delta t} \tag{3.20}$$

Where,

$$D = \frac{\partial u_i}{\partial x_i}. \tag{3.21}$$

Using equation 3.19 and 3.20 the $p^{'}$ equation can be written as:

$$\frac{\partial^2 p^{'}}{\partial x_k \partial x_k} = \frac{\rho}{\Delta t}\frac{\partial u_i^*}{\partial x_i}. \tag{3.22}$$

The solution takes place first by satisfying the momentum equations, then the velocities are adjusted such that the continuity is satisfied. The final velocity field does not satisfy the momentum equations unless $u_i^{'} = 0$.

The solution proceeds in the following way:

1. Guess a pressure p^*.

2. Solve the momentum equations for u_i^*.

$$\rho\frac{\partial u_i^*}{\partial t} = -\rho u_j^n \frac{\partial u_i^*}{\partial x_j} + \rho f_i - \frac{\partial p^*}{\partial x_i} + \mu \frac{\partial^2 u_i^*}{\partial x_j \partial x_j} \tag{3.23}$$

3. Solve for the pressure correction p'.

$$\frac{\partial^2 p'}{\partial x_k \partial x_k} = \frac{\rho}{\Delta t}\frac{\partial u_i^*}{\partial x_i} \tag{3.24}$$

4. Solve for the velocity corrections u_i'.

$$\rho\frac{\partial u_i'}{\partial t} = -\frac{\partial p'}{\partial x_i} \tag{3.25}$$

5. Update the variables.

$$u_i = u_i' + u_i^* \tag{3.26}$$

$$p = p' + p^* \tag{3.27}$$

6. If the solution has not reached steady state go back to step (2).

At this point no reference has been given to a solution scheme, therefore either a finite difference, finite element, or control volume method may be applied to these series of solution steps equally well. For the present research only a finite element method will be used to solve this set of equations.

3.1.3 Segregated vs. Penalty Formulation

One may justifiably ask, "What is the advantage, if any, to using the segregated method over the penalty method?". To answer this question, several examples follow which may shed some light on the differences between the two methods. The conditions for comparison are as follows:

- The connectivity is run through the Cuthill-McKee renumbering sequence in order to minimize the bandwidth of the system of equations.
- All problems are run on a NeXTstation 68040 computer at 25 Mhz.
- Only problems requiring less than 14 megabytes are run. This prevents the computer from relying on swap space.
- All solutions in this section use the LINPACK routines. The routines used depend on the type of equations being solved.
- All solutions are stepped through time until the convergence criterion specified by equation 3.28 has been satisfied.

$$\sqrt{\frac{\sum_{i=1}^{N}\left[\left(u_i^n - u_i^{n+1}\right)^2 + \left(v_i^n - v_i^{n+1}\right)^2\right]}{\sum_{i=1}^{N}\left[\left(u_i^{n+1}\right)^2 + \left(v_i^{n+1}\right)^2\right]}} \leq 0.1\Delta t \tag{3.28}$$

- Only two dimensional flow will be considered for this comparison.

The penalty formulation for fully implicit time integration of the element equation 3.12 becomes:

$$\begin{bmatrix} [B^{11}] & [B^{12}] \\ [B^{12}]^T & [B^{22}] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{v\} \end{Bmatrix}^{n+1} = \begin{bmatrix} [M] & [0] \\ [0] & [M] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{v\} \end{Bmatrix}^{n} + \begin{Bmatrix} \{F^x\} \\ \{F^y\} \end{Bmatrix} . \tag{3.29}$$

Where,

$$\begin{aligned} [B^{11}] &= [M] + \Delta t \left\{2\mu\,[S^{11}] + \mu\,[S^{22}] + \gamma\,[S^{11}]_{RI} + \rho\,[A]\right\} \\ [B^{12}] &= \Delta t \left\{\mu\,[S^{12}]^T + \gamma\,[S^{12}]_{RI}\right\} \\ [B^{22}] &= [M] + \Delta t \left\{\mu\,[S^{11}] + 2\mu\,[S^{22}] + \gamma\,[S^{22}]_{RI} + \rho\,[A]\right\} \\ [M] &= \rho\,[S^{00}] . \end{aligned} \tag{3.30}$$

The subscript RI in these equations refers to the reduced integration terms. The elemental form of these matrix equations are shown below.

$$A_{ij} = \int_{\Omega^e} \Psi_i \left(\bar{u}\frac{\partial \Psi_j}{\partial x} + \bar{v}\frac{\partial \Psi_j}{\partial y} \right) d\Omega^e \tag{3.31}$$

$$S_{ij}^{rs} = \int_{\Omega^e} \frac{\partial \Psi_i}{\partial x_r}\frac{\partial \Psi_j}{\partial x_s} d\Omega^e \quad \text{where } x_1 = x \text{ and } x_2 = y \tag{3.32}$$

$$F_i^x = \int_{\Omega^e} \rho f_x \Psi_i d\Omega^e + \oint_{\Gamma^e} t_x \Psi_i d\Gamma^e \tag{3.33}$$

$$F_i^y = \int_{\Omega^e} \rho f_y \Psi_i d\Omega^e + \oint_{\Gamma^e} t_y \Psi_i d\Gamma^e \tag{3.34}$$

Equation 3.29 represents one matrix equation to be solved at each time step. Once this equation is assembled, the bandwidth becomes twice plus one of the corresponding bandwidth of a single variable problem. This fact will become important in the direct comparison of penalty vs. segregated methods. The solution process for the segregated method is similar to that of the SIMPLE method [9]. Within the context of fully implicit finite element methods the solution proceeds as follows:

1. Guess an initial pressure distribution P^*.

2. Solve each momentum equation.

$$\begin{aligned}&\left[\rho\,[M] + \rho\Delta t\,[A] + \mu\Delta t\left[K^l\right]\right]\{u^*\} = \\ &\quad -\Delta t\,[C^x]\{P^*\} + \Delta t\,\{R^x\} + \rho\,[M]\{u\}^n\end{aligned} \tag{3.35}$$

$$\begin{aligned}&\left[\rho\,[M] + \rho\Delta t\,[A] + \mu\Delta t\left[K^l\right]\right]\{v^*\} = \\ &\quad -\Delta t\,[C^y]\{P^*\} + \Delta t\,\{R^y\} + \rho\,[M]\{v\}^n\end{aligned} \tag{3.36}$$

3. Solve the pressure correction to enforce continuity.

$$\left[K^l\right]\left\{P'\right\} = -\frac{\rho}{\Delta t}\left[\left[C^x\right]\left\{u^*\right\} + \left[C^y\right]\left\{v^*\right\}\right] \tag{3.37}$$

4. Solve the velocity corrections.

$$\left[M\right]\left\{\dot{u}'\right\} = -\left[C^x\right]\left\{P'\right\} \tag{3.38}$$

$$\left[M\right]\left\{\dot{v}'\right\} = -\left[C^y\right]\left\{P'\right\} \tag{3.39}$$

5. Update velocities and pressures.

$$\left\{u\right\}^{n+1} = \left\{u^*\right\} + \Delta t\left\{\dot{u}'\right\} \tag{3.40}$$

$$\left\{v\right\}^{n+1} = \left\{v^*\right\} + \Delta t\left\{\dot{v}'\right\} \tag{3.41}$$

$$\left\{P\right\}^{n+1} = \left\{P^*\right\} + \left\{P'\right\} \tag{3.42}$$

6. Test for convergence, if not converged, then:

$$\left\{u^*\right\} = \left\{u\right\}^{n+1} \tag{3.43}$$

$$\left\{v^*\right\} = \left\{v\right\}^{n+1} \tag{3.44}$$

$$\left\{P^*\right\} = \left\{P\right\}^{n+1} \tag{3.45}$$

`Return to Step 2.`

The elemental level components of the matrices are defined as follows:

$$\left[K^l\right] = \left[S^{11}\right] + \left[S^{22}\right] \tag{3.46}$$

$$C^x_{ij} = \int_{\Omega^e} \Psi_i \frac{\partial \Psi_j}{\partial x}\, d\Omega^e \tag{3.47}$$

$$C^y_{ij} = \int_{\Omega^e} \Psi_i \frac{\partial \Psi_j}{\partial y}\, d\Omega^e \tag{3.48}$$

$$R^x_i = \int_{\Gamma^e} \Psi_i \frac{\partial u}{\partial n}\, d\Gamma^e + \rho \int_{\Omega^e} \Psi_i f_x\, d\Omega^e \tag{3.49}$$

$$R^y_i = \int_{\Gamma^e} \Psi_i \frac{\partial v}{\partial n}\, d\Gamma^e + \rho \int_{\Omega^e} \Psi_i f_y\, d\Omega^e \tag{3.50}$$

Programming

Solving these sets of matrix equations for both the penalty and segregated methods can be very challenging. In this section a brief description of the matrix solution methods is given. The routines used for both of the finite element methods are the LINPACK routines [16]. In the penalty method the $[A]$ matrix changes each iteration. This gives rise to a change in the complete system matrix each iteration. Since the system matrix changes each iteration, the matrix must be factored and solved during each iterative step. The system matrix is also unsymmetric, requiring the LINPACK routine DGBFA to factor the system matrix and DGBSL to obtain a solution from the factored set of equations. In the segregated solution scheme there are five solution steps within each iteration, two momentum, one pressure, and two velocity corrections. However, on closer examination, the one pressure and two velocity correction equations are linear and symmetric. Therefore, the pressure and velocity correction equations do not change each iteration. For this reason the $\left[K^l\right]$ and $[M]$ need only be factored once, before the time loop. The LINPACK routine DPBSL is used to accomplish this task. Once in the time loop, the system matrices associated with the u^* and v^* momentum equations, are factored using the routine DGBFA and subsequently solved using DGBSL. Since the $\left[K^l\right]$ and $[M]$ equations are previously factored, only a short amount of time is used to obtain solutions to P', u', and v'.

The size of the segregated system equations are one-half the size of the corresponding penalty formulation. Therefore, one would expect quicker solution time per system of equations. Yet, there are five systems of equations in the segregated formulation, verses one in the penalty. In the segregated method only two factorizations occur within each time step, along with five back substitutions. All of the LINPACK routines used are banded equation solvers. Hence, it is instructive to notice the bandwidth characteristics of both formulations. The bandwidth of the penalty method system equations are related to the segregated system of equations by:

$$hbw_p = ndim * hbw_s + 1. \tag{3.51}$$

$$\begin{aligned} hbw_p &= \text{halfbandwidth for penalty} \\ hbw_s &= \text{halfbandwidth for segregated} \\ ndim &= \text{dimension of the problem, two in this case} \end{aligned} \tag{3.52}$$

The halfbandwidth quoted throughout this chapter is for one degree of freedom per node.

Cavity Problem

The cavity problem has been used for years as a validation problem for incompressible flows. Work began as early as 1966 [60] on numerically predicting flow in a cavity. Since that time, numerous investigations have been performed on this problem. A particularly complete study was performed by Ghia and Ghia [61]. In their work, grids with as many as 66,000 nodes and

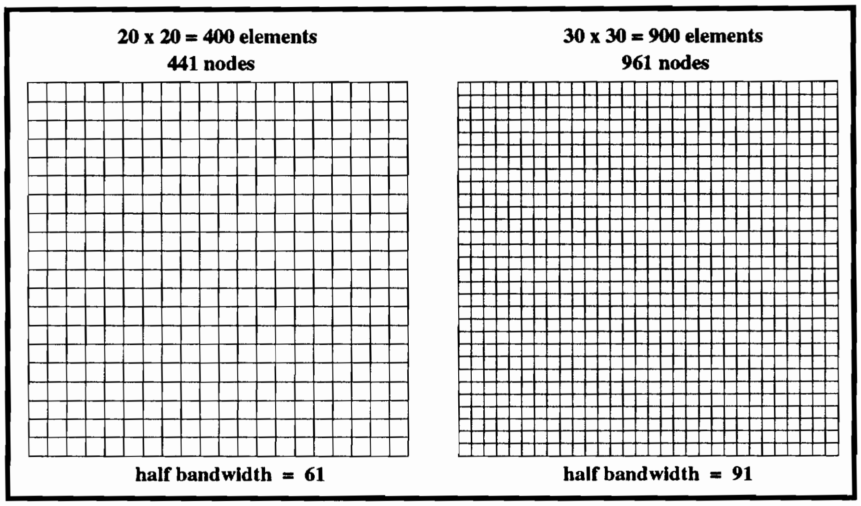

Figure 3.1: Grids for cavity examples

Reynolds numbers as high as 10,000 were calculated. They included a table of centerline velocity values for each set of flow conditions. In the present example, grids with 961 and 441 nodes are calculated. This is only a computer hardware limitation and not a formulation limit. The grids are shown in Figure 3.1. The boundary conditions for the cavity problem are shown in Figure 3.2. The boundary conditions at the singular points on the upper corners of the domain are shown as: $u = 1$, and $v = 0$. The initial conditions within the cavity are zero velocities. The half-bandwidth specified in Figure 3.1 is for a single degree of freedom per node only.

The computer timing results are shown in Tables 3.1, 3.2, 3.3, and 3.4. As shown in these tables, the segregated solution method is very competitive with the penalty method as far as computer time is concerned. At low Reynolds numbers the penalty method performs better than the segregated method. This better performance is due to a simple fact; in equation 3.37 as

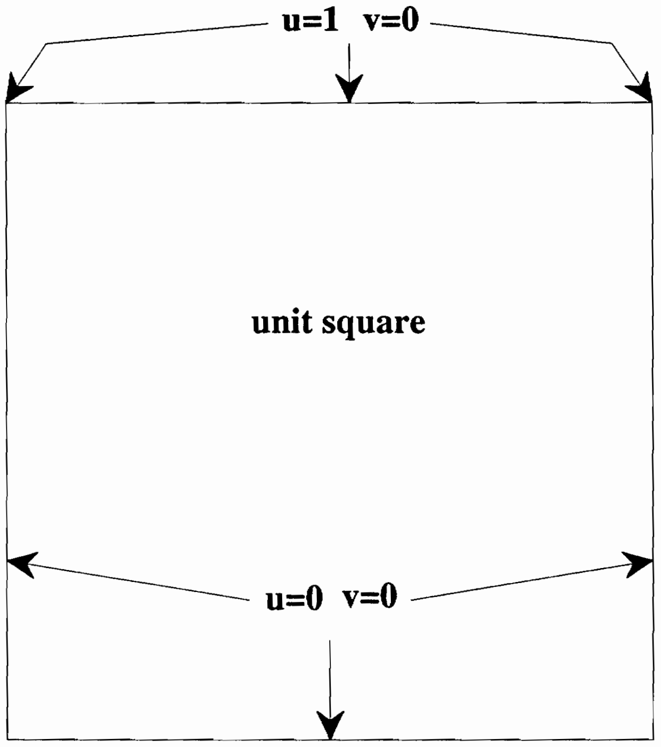

Figure 3.2: Cavity boundary conditions

Table 3.1: Segregated runs on a 30x30 cavity

Re	Computer time/step Linpack (sec)	# of time steps	Δt	Computer time Overhead (sec)	Computer time Linpack (sec)	Computer time Overall (sec)
100	12.54	28	1	179.84	351.18	531.02
100	12.55	27	2	175.38	338.95	514.33
1000	12.30	65	1	398.42	799.20	1197.62
1000	12.30	52	2	322.38	639.84	962.22

Table 3.2: Penalty runs on a 30x30 cavity

Re	Computer time/step Linpack (sec)	# of time steps	Δt	Computer time Overhead (sec)	Computer time Linpack (sec)	Computer time Overall (sec)
100	41.29	14	1	110.83	578.11	688.94
100	40.80	8	2	74.51	326.38	400.89
1000	40.91	59	1	376.30	2413.84	2790.14
1000	40.71	34	2	228.28	1384.11	1612.39

Table 3.3: Segregated runs on a 20x20 cavity

Re	Computer time/step Linpack (sec)	# of time steps	Δt	Computer time Overhead (sec)	Computer time Linpack (sec)	Computer time Overall (sec)
100	2.88	26	1	68.92	74.81	143.73
100	2.89	25	2	67.25	72.16	139.41
1000	2.83	65	1	163.69	183.64	347.33
1000	2.74	52	2	129.39	142.31	271.70

Table 3.4: Penalty runs on a 20x20 cavity

Re	Computer time/step Linpack (sec)	# of time steps	Δt	Computer time Overhead (sec)	Computer time Linpack (sec)	Computer time Overall (sec)
100	7.98	13	1	38.47	103.70	142.17
100	7.97	8	2	28.17	63.78	91.95
1000	8.04	72	1	161.77	578.73	740.50
1000	8.08	40	2	94.38	323.28	417.66

ρ becomes small, so too does the right hand side of the P' equation, and, as a result, the P' distribution is very small. This results in a slow convergence of the pressure which in turn slows the convergence of the entire system of equations. In fact, if one attempts to solve for Reynolds number equal to zero by setting $\rho = 0$, equation 3.37 will give $P' = 0$ throughout the domain, and the segregated solution method will never converge. This problem does not exist if the equations are written in non-dimensional form [11]. However, the penalty method can converge in one iteration for Reynolds number equal to zero. At higher Reynolds numbers, the segregated method tends to out perform the penalty method. At all Reynolds numbers, the amount of time spent within the LINPACK subroutines is much less for the segregated method than for the penalty method.

A comparison to the previously reported data of Ghia [61] is shown in Figures 3.3, 3.4, 3.5, and 3.6. At a Reynolds number of 100 the comparison to Ghia's work is reasonable. However, for a Reynolds number of 1000, the

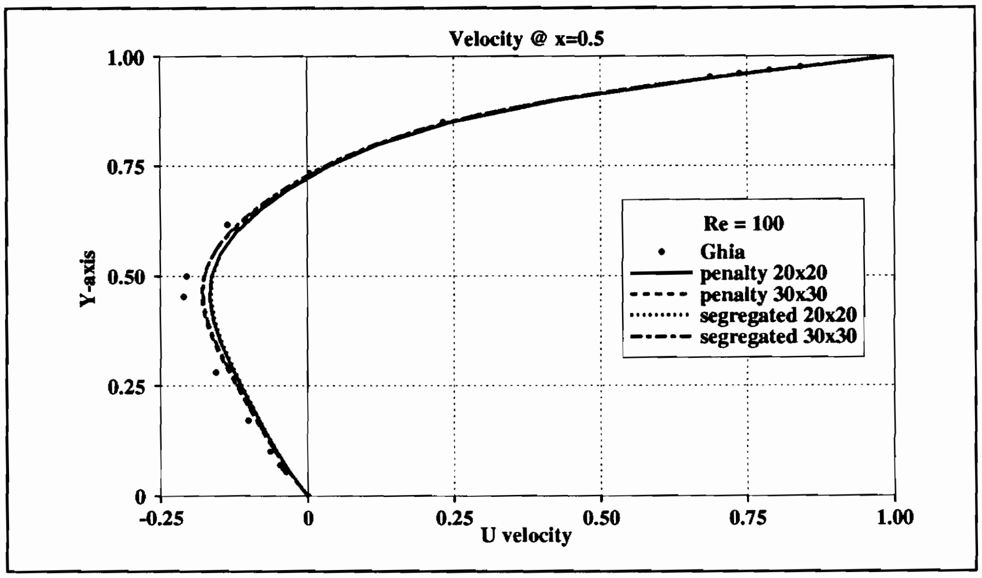

Figure 3.3: Cavity U velocity profiles at Re = 100

grids used in the present work were not fine enough to capture an accurate solution. This inaccurate solution exists for both the penalty and segregated solution methods.

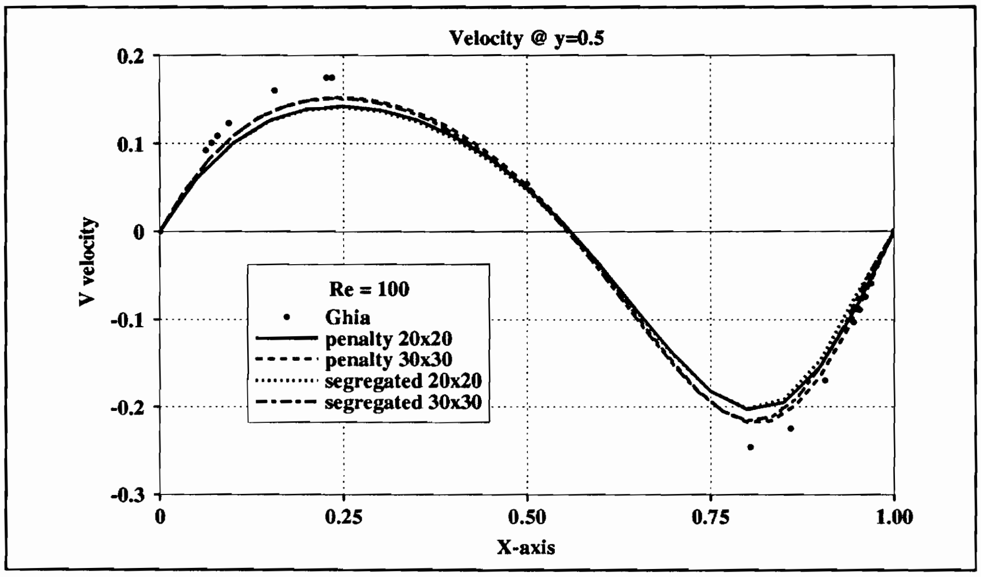

Figure 3.4: Cavity V velocity profiles at Re = 100

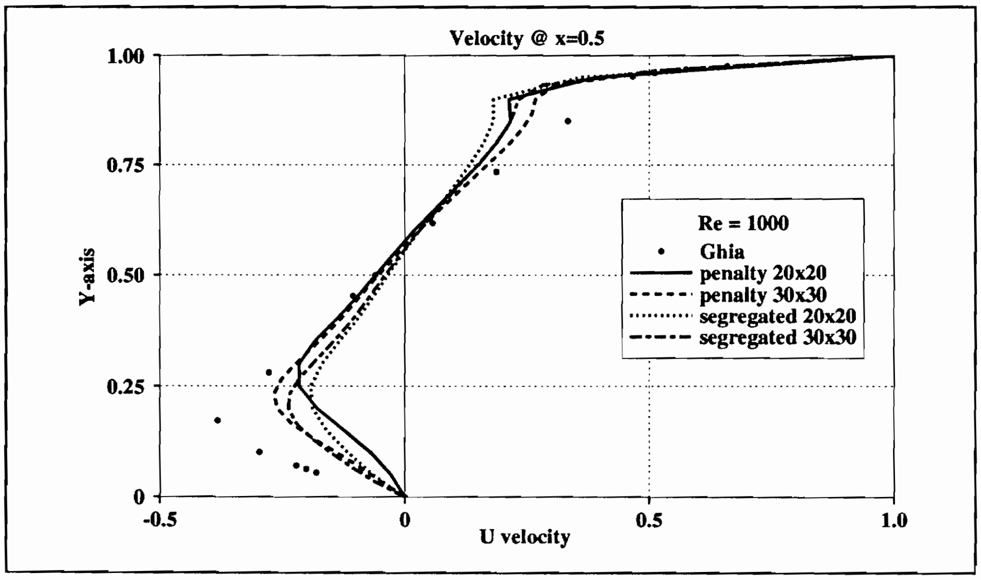

Figure 3.5: Cavity U velocity profiles at Re = 1000

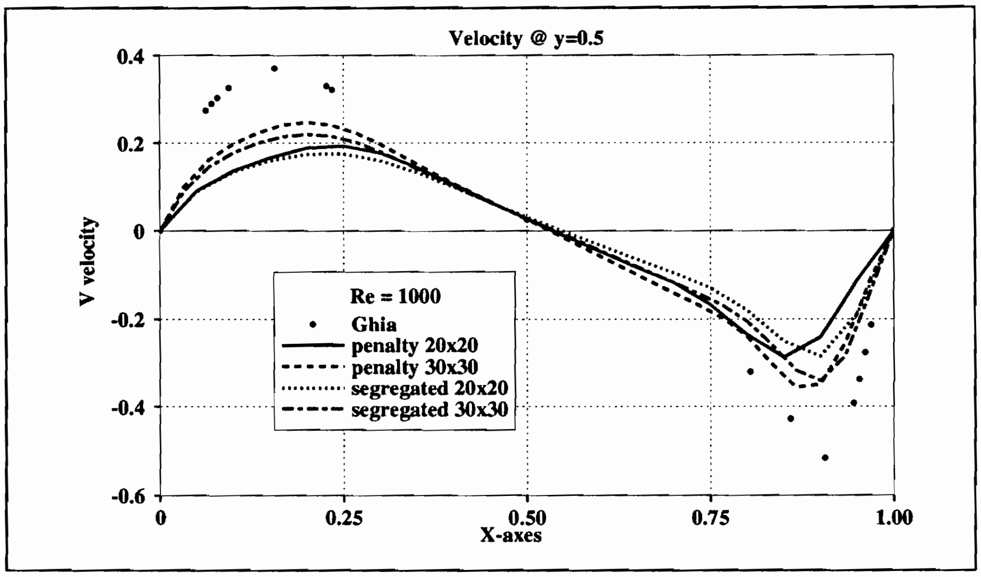

Figure 3.6: Cavity V velocity profiles at Re = 1000

Backstep Problem

Flow over a backward facing step has become another major validation case often chosen over the cavity case. This preference is due to the existence of a singularity in the cavity flow, and the presence of a shear layer behind the backstep.

Armaly *et. al* [62] studied this problem in extreme detail. He reported laser-Doppler measurements for Reynolds numbers between 70 and 8000 thereby covering the laminar, transition, and turbulent regimes. Several authors held GAMM-Workshop to produce a volume of work consisting of experimental and numerical prediction of flow over a backward facing step [63].

Case (iii) from this GAMM-Workshop is used here for comparison purposes. The geometry, boundary conditions, and Reynolds number are shown in Figure 3.7 The grid and data chosen for the simulation contains 1404 elements and 1512 nodes as shown in Figure 3.8. After use of the Cuthill McKee algorithm, the halfbandwidth for single degree of freedom per node is 49.

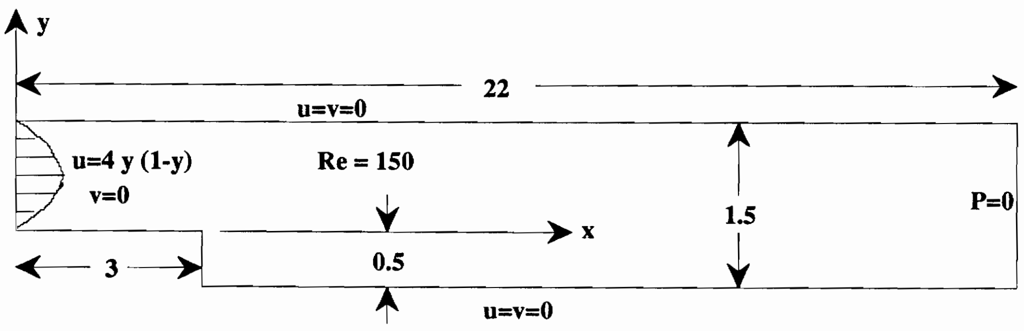

Figure 3.7: Backstep geometry

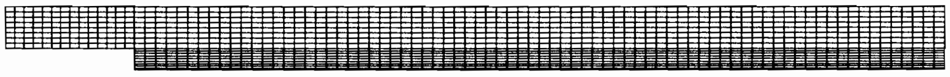

Figure 3.8: Backstep grid

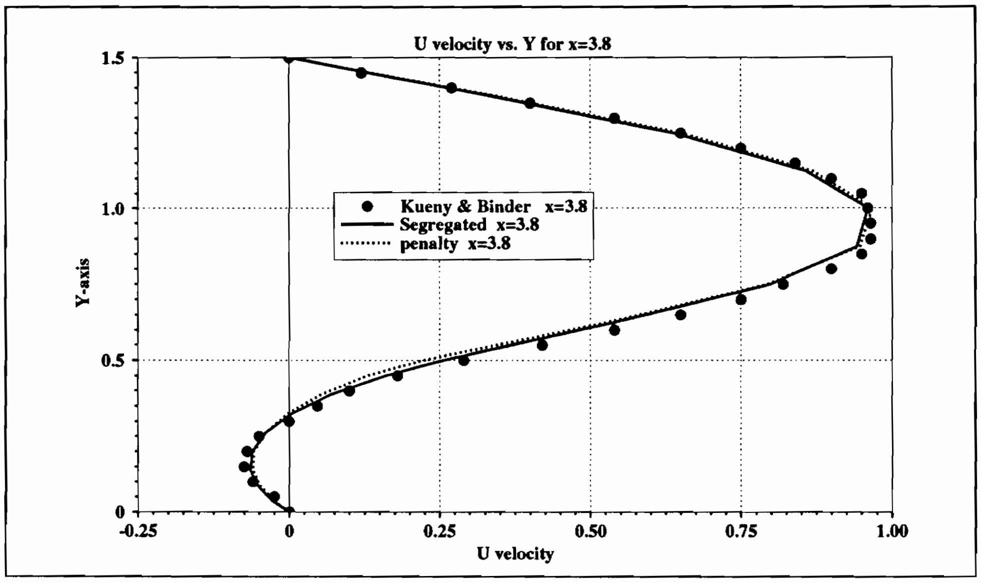

Figure 3.9: U velocitys at x=3.8

Comparisons to the experimental data of Kueny and Binder [64] are shown in Figures 3.9, 3.10, and 3.11. Both the penalty and segregated solutions compare favorably with the experimantal data. The penalty solution showed some problems downstream of the backstep. This problem could be attributed to the difficulty in obtaining a constant downstream pressure of zero. The work of Yagawa and Eguchi [65] may be consulted for further information on this subject.

As seen in Table 3.5, the segregated method performed equally as well as the penalty method. In fact the computer time required for a solution is

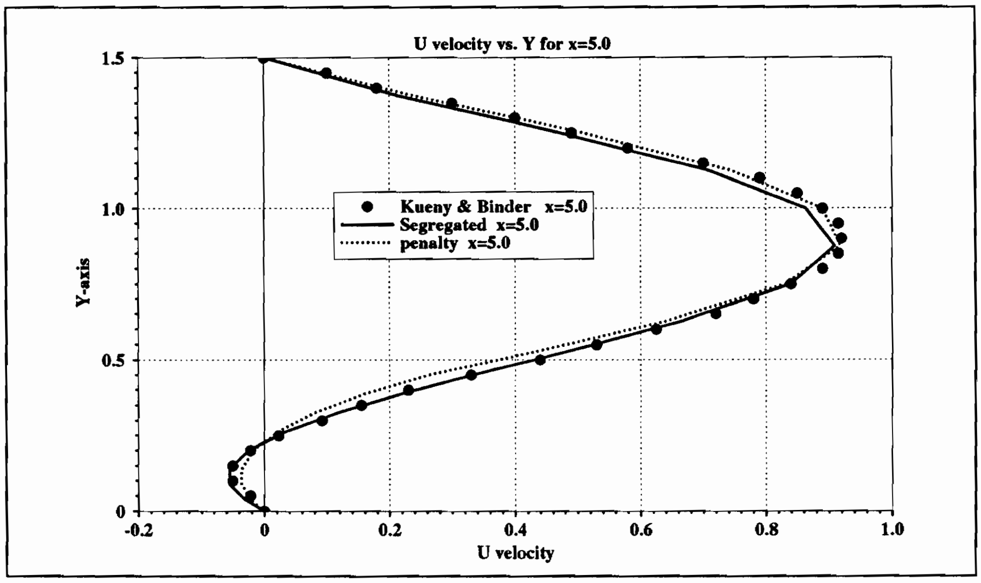

Figure 3.10: U velocitys at x=5.0

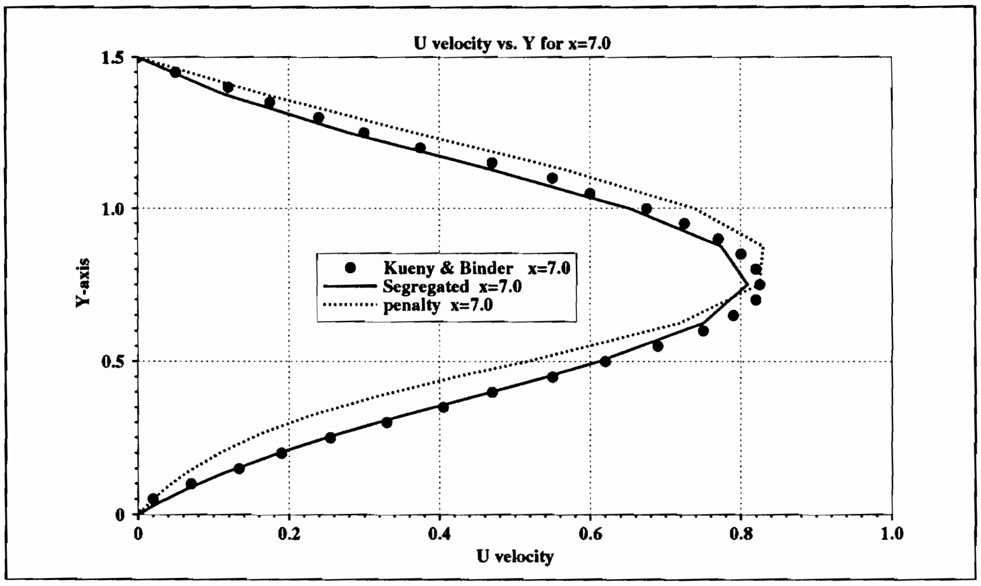

Figure 3.11: U velocitys at x=7.0

Table 3.5: Computer runs for two dimensional backstep

Method	Computer time/step Linpack (sec)	# of time steps	Δt	Computer time Overhead (sec)	Computer time Linpack (sec)	Computer time Overall (sec)
Pen.	23.62	42	1	314.96	992.31	1307.27
Seg.	6.81	56	1	480.02	381.20	861.22
Pen.	24.43	22	2	185.37	537.52	722.89
Seg.	6.88	33	2	290.22	227.03	517.25

lower for the segregated method than for the penalty method. Both methods performed very well when compared to the experimental data of Kueny and Binder [64].

3.2 Numerical Comparisons

In this section, a numerical simulation of the cavity flow defined in Figure 3.2 is performed. This simulation is based on the segregated solution method presented in previous section with an iterative solver, as described in the next chapter. The goal of this section is quite different from previous sections within this chapter. The goal is to obtain a solution which will correlate very well with previously reported data. For this comparison the data of Ghia [61] will be used. Five different grids were used in order to obtain a converged solution. These grids are shown in Figures 3.12, 3.13, 3.14, 3.15, and 3.16. Using these five grids the solution was obtained for a Reynolds number of

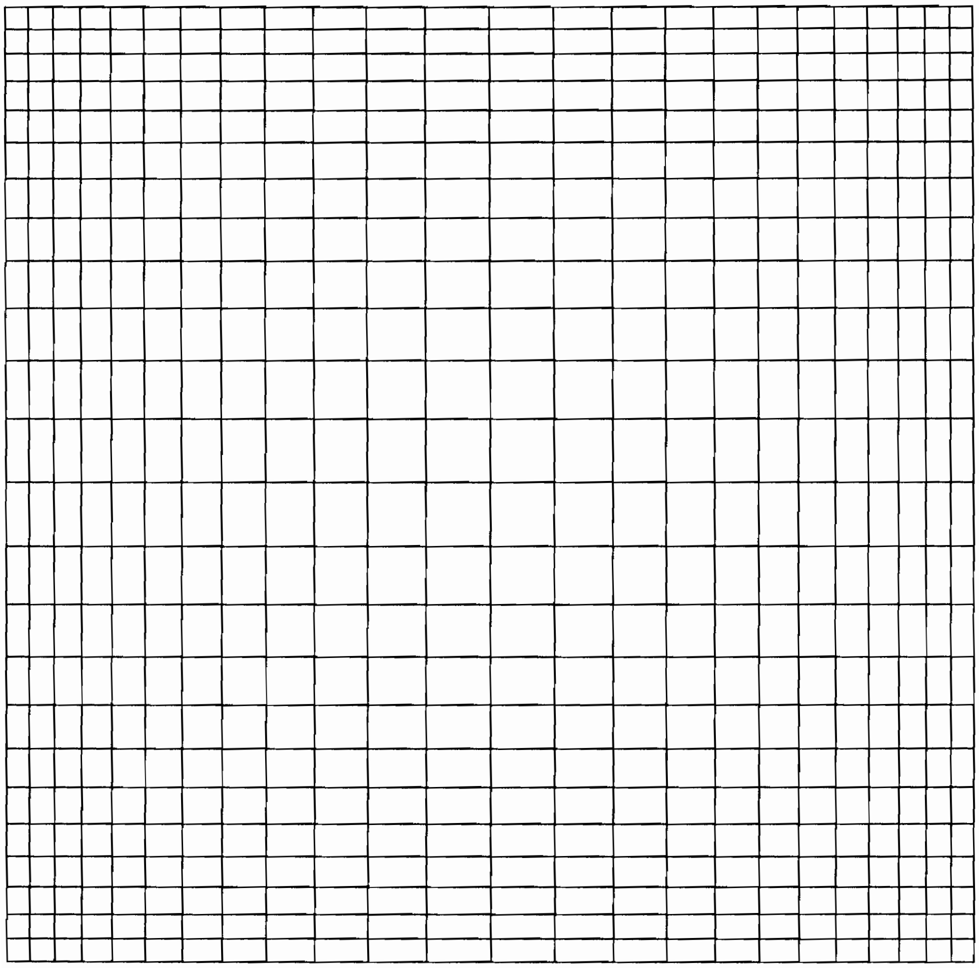
Figure 3.12: 25x25 grid for cavity simulation

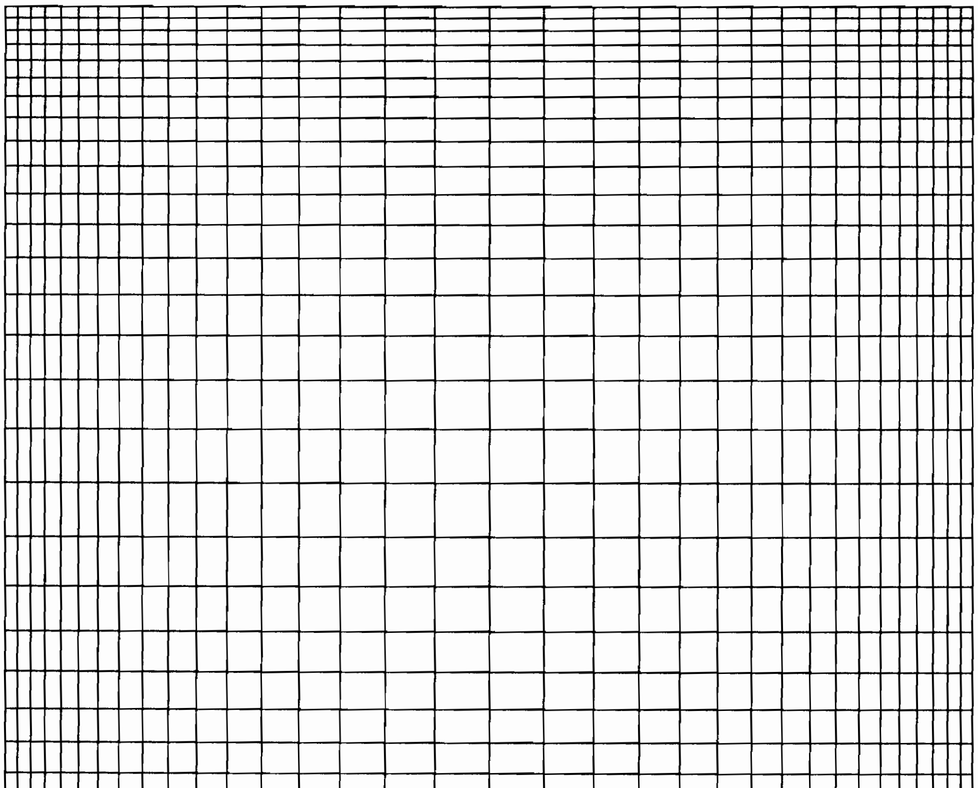

Figure 3.13: 35x35 grid for cavity simulation

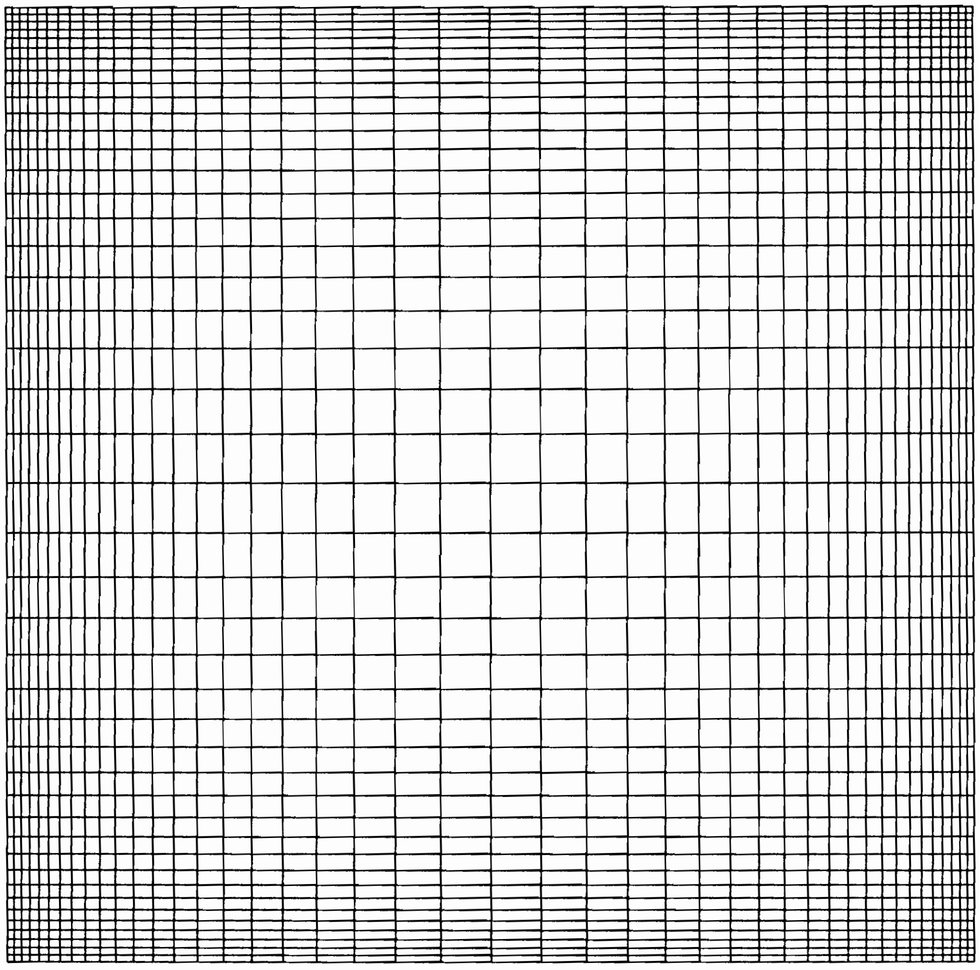

Figure 3.14: 45x45 grid for cavity simulation

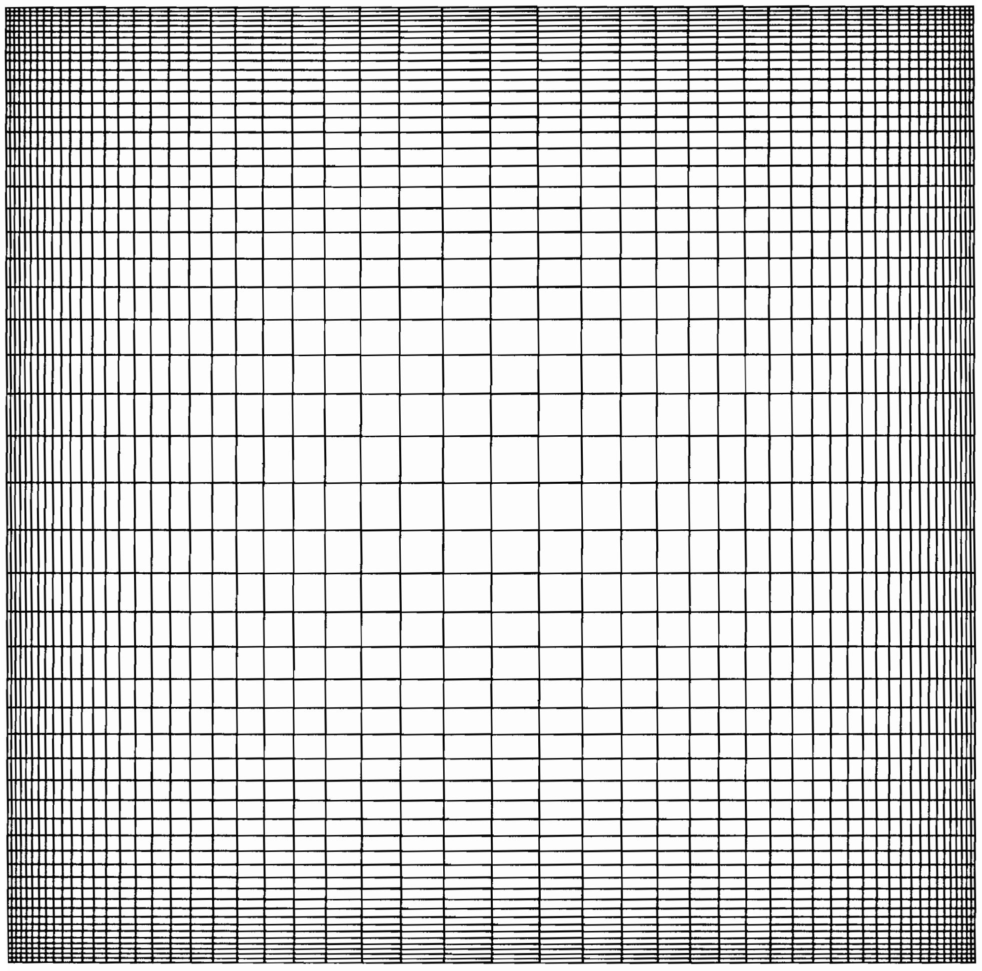

Figure 3.15: 55x55 grid for cavity simulation

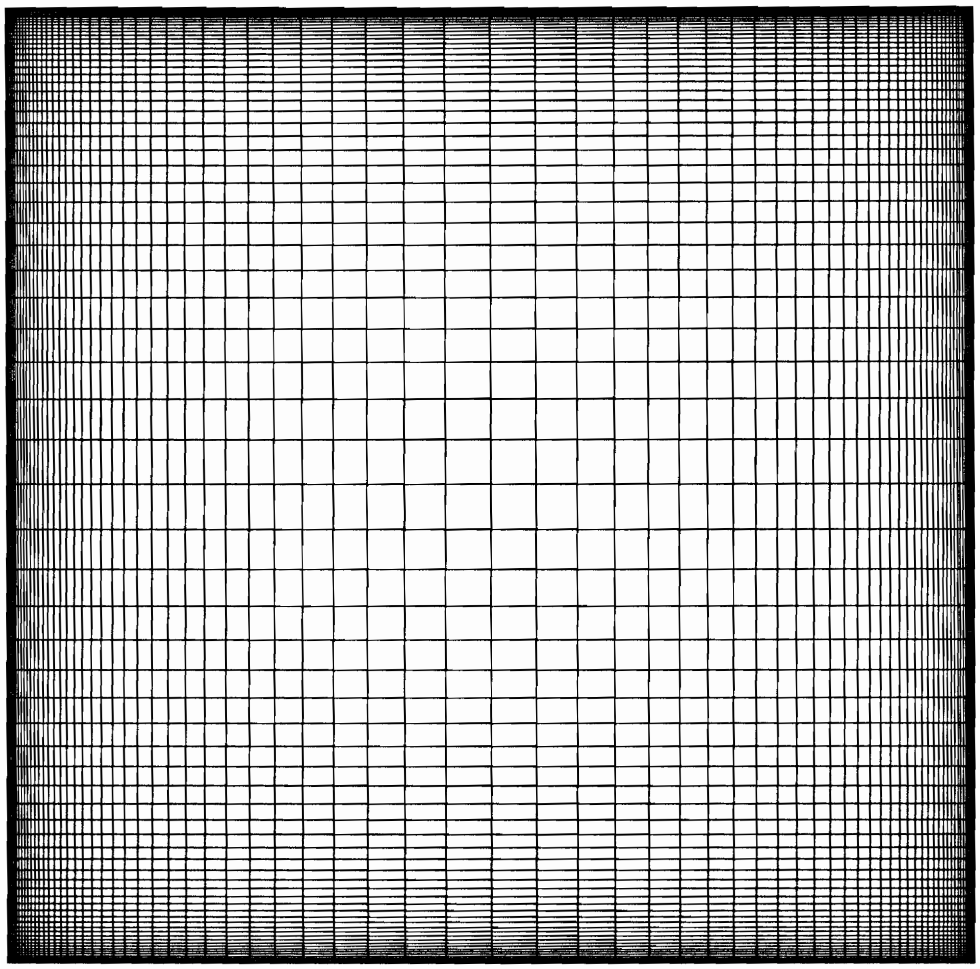

Figure 3.16: 75x75 grid for cavity simulation

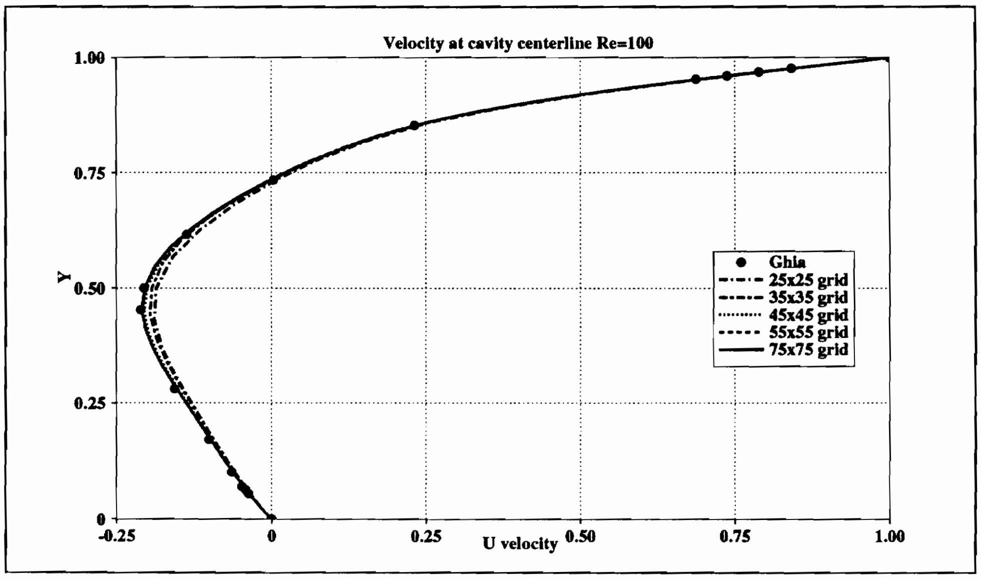

Figure 3.17: U velocity comparisons at Re=100

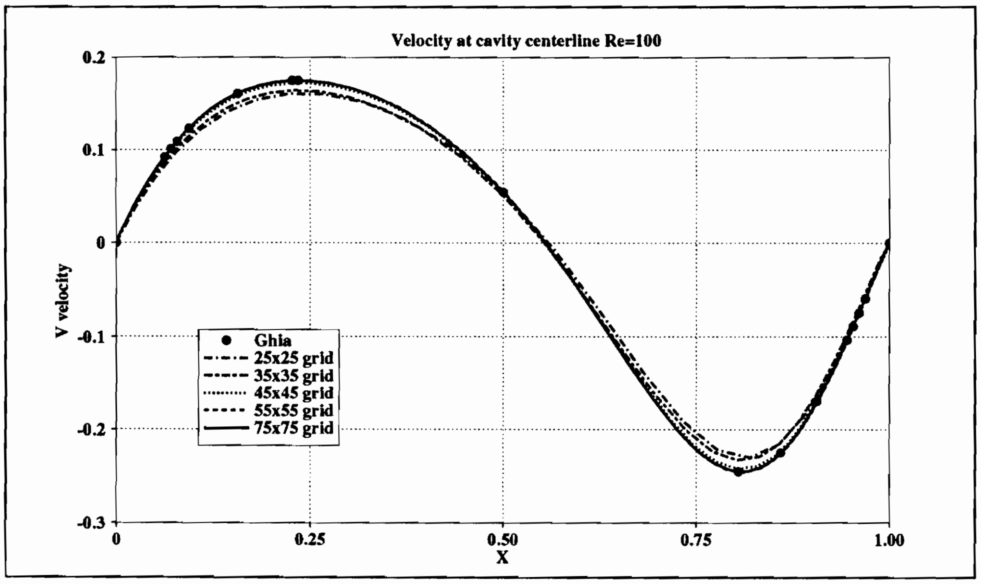

Figure 3.18: V velocity comparisons at Re=100

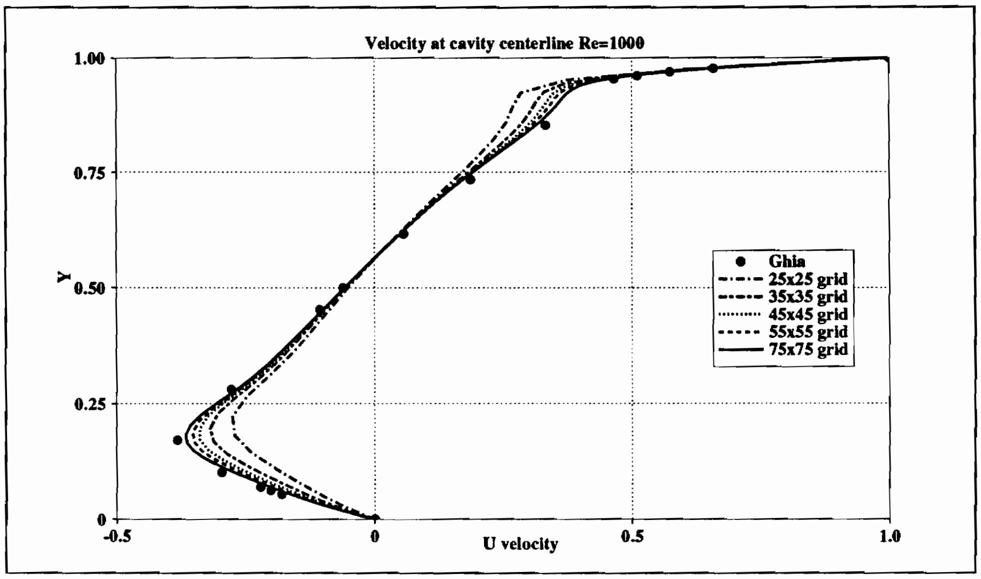

Figure 3.19: U velocity comparisons at Re=1000

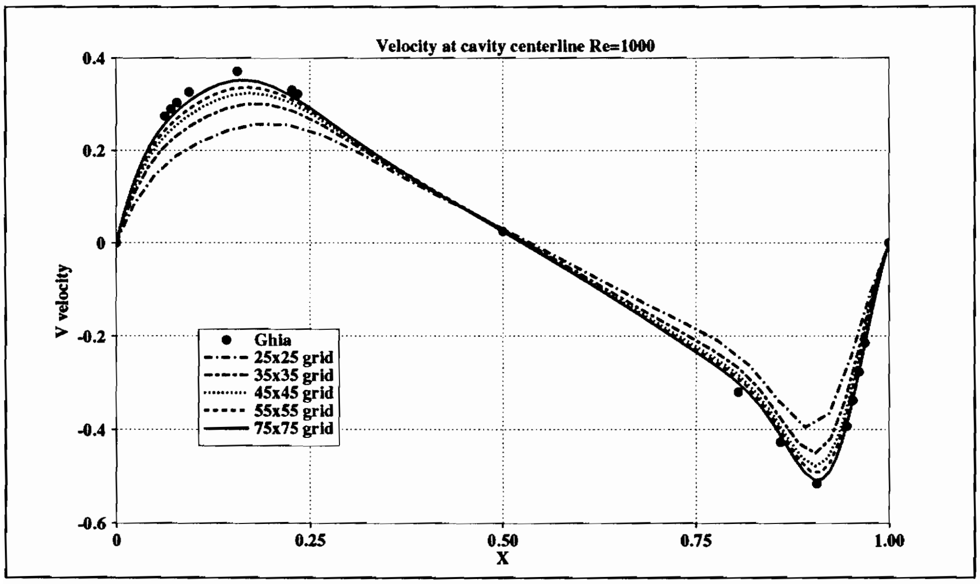

Figure 3.20: V velocity comparisons at Re=1000

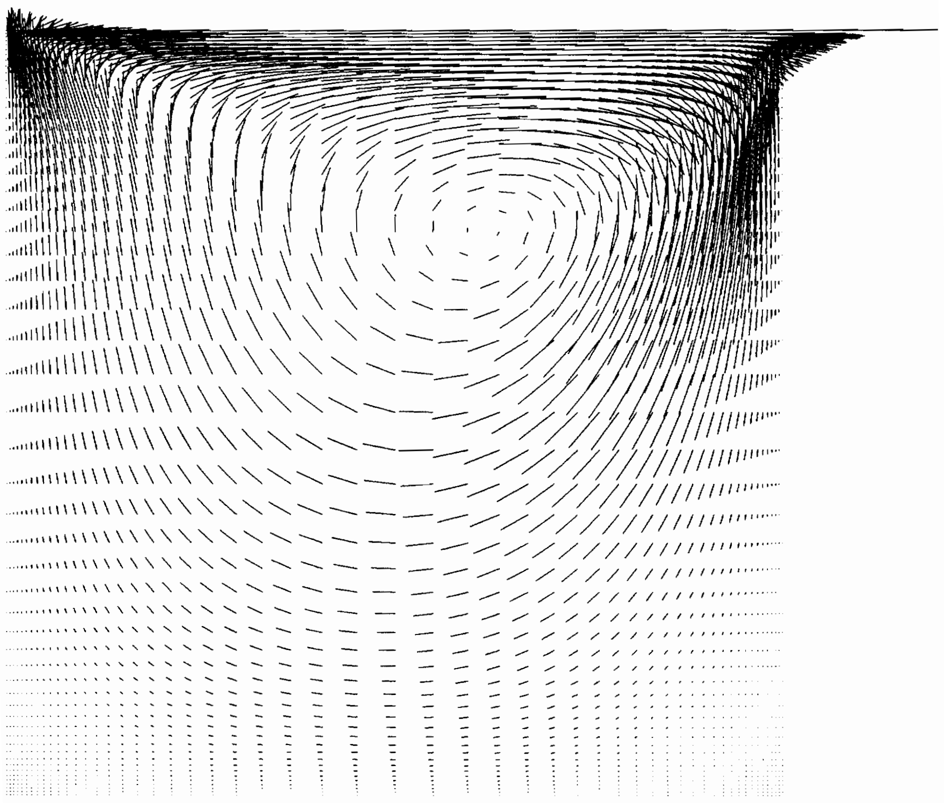
Figure 3.21: Velocity vectors for Re=100

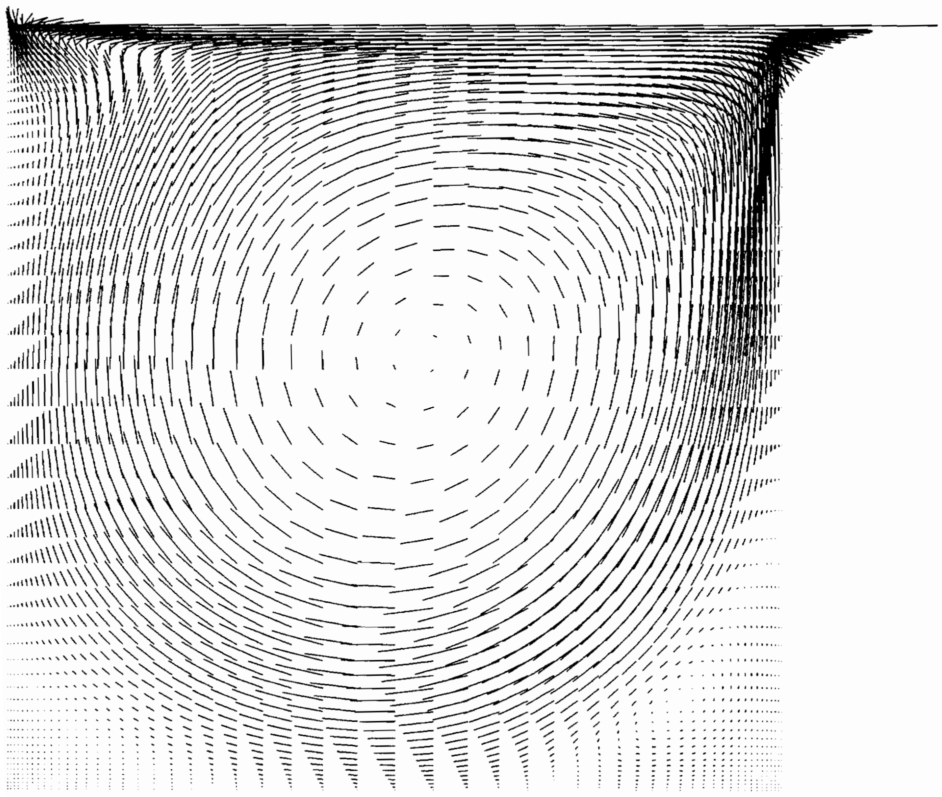

Figure 3.22: Velocity vectors for Re=1000

100 and 1000. A comparison to the numerical solutions of Ghia [61] is shown in Figures 3.17, 3.18, 3.19, and 3.20. As shown in these figures, the finite element solution approaches the solutions of Ghia as each grid becomes finer. This shows a convergence to the same solution as that obtain from Ghia. A velocity vector plot for the 55 x 55 grid is shown in Figures 3.21, and 3.22.

3.3 Conclusions

The work of this chapter enforces the belief that the segregated method is indeed a viable method of choice in the solution of incompressible fluid flow. As compared to the penalty method, the segregated method requires less time for higher Reynolds numbers, and more computer time for the lower Reynolds numbers. Also, the segregated method requires less time within the LINPACK routines for all cases per time step.

In the last section, the segregated solution method is used to converge cavity flow to previously published data of Ghia [61]. This section shows the ability of the method to converge on an increasingly finer mesh. Hence, the method is shown to be consistent.

Chapter 4

Iterative Solutions

4.1 Matrix Condition Number

The condition number of a matrix plays a crucial role in the solution behavior of an iterative scheme. The condition number is defined as the ratio of the largest over the smallest eigenvalue of the matrix. As the condition number becomes larger, iterative solution methods require larger amounts of computer time. Iterative solution methods are far more susceptible to high condition number matrices than are direct solution methods. Computer time required to obtain a direct solution of a matrix is not a function of the condition number. However, the accuracy of the direct solution of a matrix is a function of the condition number. Hence, a direct solution may be able to obtain an acceptable solution whereas an iterative solution method will not be able to obtain a solution at all due to computer time limitations.

In the penalty method, the constraint portion of the system matrix is made singular by reduced integration techniques. This singular matrix is multiplied by the large penalty number, and added to the remaining portion of the system matrices. This procedure yields a system matrix which approaches a singular or ill conditioned matrix as the penalty number increases. Iterative methods are very sensitive to this condition number as shown in equation 2.13. Hence, as the condition number increases, so too does the number of iterations required to obtain an acceptable solution. Carey *et. al* [66] studied this problem of the penalty method and in his conclusions he states; "*Preconditioning is seen to be an important and sensitive issue and iterative performance for the linear problem is poor*". In reading through this paper it becomes apparent just how difficult an iterative solution to the penalized Navier-Stokes equations is to obtain. Actually, unless preconditioning is used in one form or another, the iterative penalty solution will not converge.

As an example, the 20x20 cavity problem is run at three different penalty numbers, and the condition number is examined at each iteration to obtain the plot in Figure 4.1. Reddy [15] suggests a penalty number of $\mu \times 10^6 \leq 10^{13}$. However, as the penalty number becomes large, the condition number of the matrix increases and an iterative solution requires larger amounts of computer time. For this reason the iterative penalty solution is not attempted in the present work. Interested readers may find the work of Gunzburger [67] and Reddy [68] of interest in relation to the iterative penalty methods.

The condition number for the various steps involved in the segregated solution method is shown in Figure 4.2. Note in this figure that there is a low condition number for the momentum and velocity correction equations, but

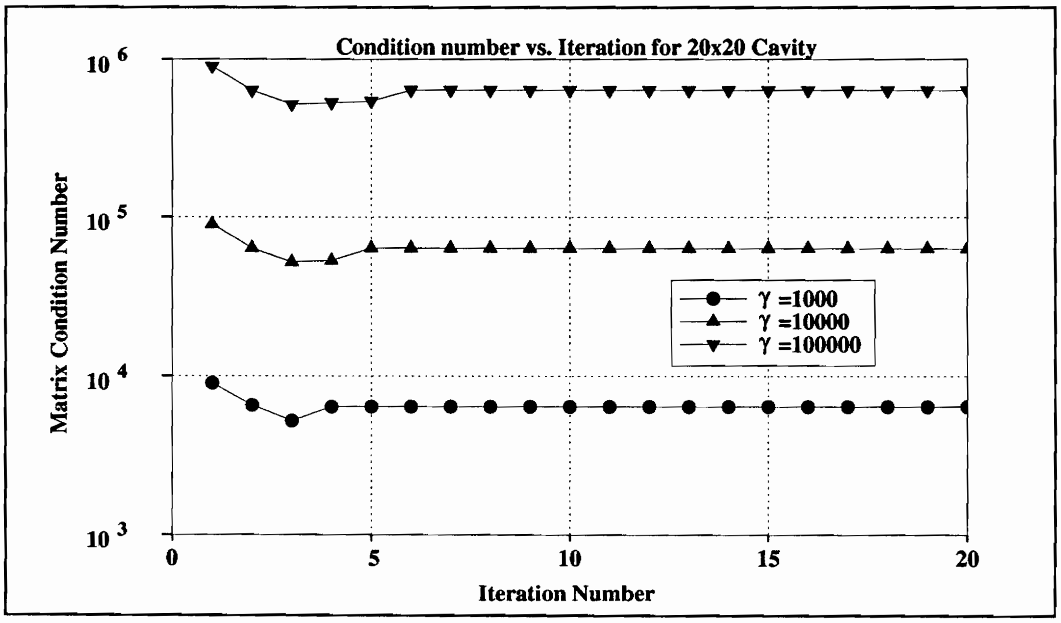

Figure 4.1: Matrix condition verses iteration, penalty method

a relatively large condition number for the pressure correction equation. It is expected that the pressure correction equation may supply more problems to an iterative solution method than either the momentum or velocity correction equations.

The present chapter is dedicated to the investigation of the use iterative methods applied to the segregated finite element formulation. In this chapter an investigation is carried out using several types of iterative methods to solve each of the equations within the segregated solution method.

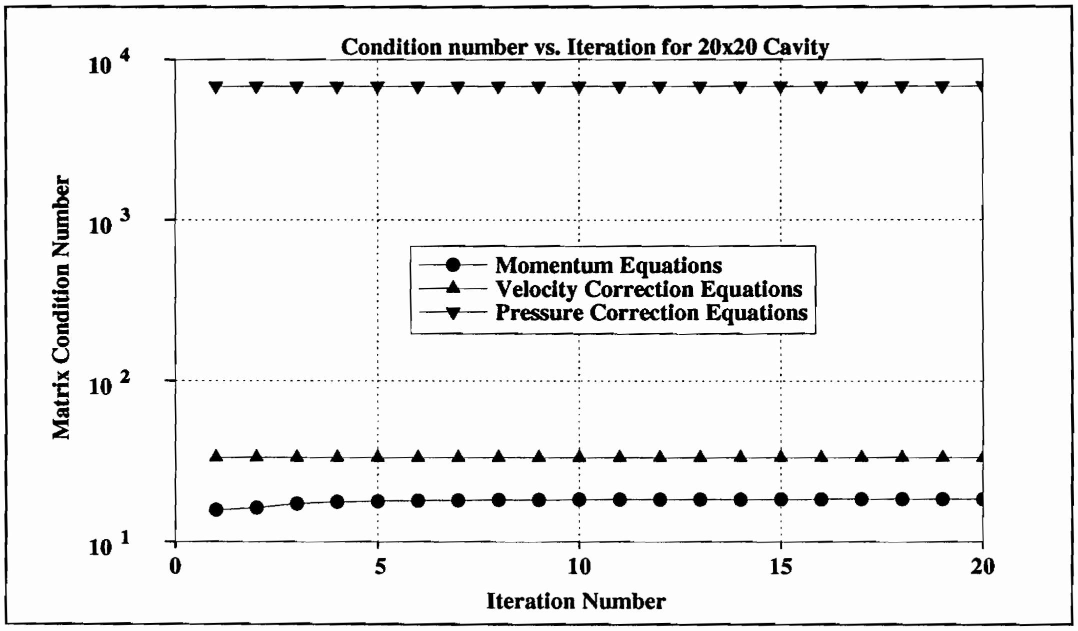

Figure 4.2: Matrix condition verses iteration, segregated method

4.2 Iterative Methods

The iterative methods used in the present work are of the gradient type. These methods can be looked at as a minimization of the norm of the residual, $r = Ax - b$, in some sense. The type of method used depends on the properties of A. If A is symmetric and positive definite, then the standard conjugate gradient method is used. If A is unsymmetric then the biconjugate gradient method is used.

4.2.1 Conjugate Gradient Method

The conjugate gradient method of Hestenes and Stiefel [38] is used for symmetric positive definite matrices. The algorithm is presented in equation 4.1.

$$
\begin{aligned}
\{x^0\} = \; & \text{arbitrary} \\
\{p^0\} = \; & \{r^0\} = \{b\} - [A]\{x^0\} \\
\texttt{Do k} = \; & 0, 1, 2, 3, 4, \ldots. \\
& \left\{u^k\right\} = [A]\left\{p^k\right\} \\
& \alpha_k = \frac{\left\{r^k\right\}^T\left\{r^k\right\}}{\left\{p^k\right\}^T\left\{u^k\right\}} \\
& \left\{x^{k+1}\right\} = \left\{x^k\right\} + \alpha_k\left\{p^k\right\} \\
& \left\{r^{k+1}\right\} = \left\{r^k\right\} - \alpha_k\left\{u^k\right\} \\
& \beta_k = \frac{\left\{r^{k+1}\right\}^T\left\{r^{k+1}\right\}}{\left\{r^k\right\}^T\left\{r^k\right\}} \\
& \left\{p^{k+1}\right\} = \left\{r^{k+1}\right\} - \beta_k\left\{p^k\right\} \\
\texttt{EndDo} &
\end{aligned}
\tag{4.1}
$$

4.2.2 Biconjugate Gradient Method

This method was displayed by Jea and Young [45] as a method for solving nonsymmetrizable linear equations. This method is actually the "biconjugate gradient" method of Fletcher [69].

```
$\{x^0\} =$ arbitrary
$\{\tilde{r}^0\} =$ arbitrary
$\{p^0\} = \{r^0\}$
$\{\tilde{p}^0\} = \{\tilde{r}^0\}$
Do k = 1, 2, 3, 4, ....
        $\left\{u^k\right\} = [A]\left\{p^k\right\}$
        $\left\{\tilde{u}^k\right\} = [A]^T\left\{\tilde{p}^k\right\}$
        $\left\{x^{k+1}\right\} = \left\{x^k\right\} + \lambda_k\left\{p^k\right\}$                (4.2)
        $\left\{r^{k+1}\right\} = \left\{r^k\right\} - \lambda_k\left\{u^k\right\}$
        $\left\{\tilde{r}^{k+1}\right\} = \left\{\tilde{r}^k\right\} - \lambda_k\left\{\tilde{u}^k\right\}$
        $\left\{p^k\right\} = \left\{r^k\right\} - \alpha_k\left\{p^{k-1}\right\}$
        $\left\{\tilde{p}^k\right\} = \left\{\tilde{r}^k\right\} - \alpha_k\left\{\tilde{p}^{k-1}\right\}$
        $\lambda_k = \frac{\left\{\tilde{r}^k\right\}^T\left\{r^k\right\}}{2\left\{u^k\right\}^T\left\{\tilde{p}^k\right\}}$
        $\alpha_k = \frac{\left\{\tilde{r}^k\right\}^T\left\{r^k\right\}}{2\left\{r^{k-1}\right\}^T\left\{\tilde{r}^{k-1}\right\}}$
EndDo
```

4.2.3 Preconditioning

Theoretical and numerical evidence show that these methods converge fairly rapidly when the system matrix eigenvalues are clustered together on the real axes. As seen in Figure 4.2, this is the case for the momentum and velocity correction equations, but not the case for the pressure correction equation. There is no guarantee that the momentum or velocity corrections equation always has a small cluster of eigenvalues for every problem. This brings up the question of how to change the matrices such that the eigenvalues become clustered without changing the system of equations. Answering this question is the entire focus of preconditioning.

There are fundamentally three forms of preconditioners; left, right, and two–sided. If you start with the system

$$[A]\{x\} = \{b\}. \tag{4.3}$$

Then left preconditioning is obtained by:

$$[Q]^{-1}[A]\{x\} = [Q]^{-1}\{b\}. \tag{4.4}$$

Right preconditioning is obtained by:

$$\left([A][Q]^{-1}\right)[Q]\{x\} = \{b\}. \tag{4.5}$$

And two–sided preconditioning is obtained by:

$$\left([Q_L]^{-1}[A][Q_R]^{-1}\right)([Q_R]\{x\}) = [Q_L]^{-1}\{b\}. \tag{4.6}$$

If $[Q]^{-1} = [A]$, then $[Q]^{-1}[A] = [A][Q]^{-1} = [I]$, and all of the eigenvalues are unity. In this case the conjugate gradient methods will converge in one iteration. The goal in two sided preconditioning is to obtain $[Q_L]^{-1}[A][Q_R]^{-1}$ $= [I]$. In short, the goal is to obtain a matrix $[Q]$ which is an approximation to $[A]$ and is easy to invert. One of the simplest forms of preconditioning is the Jacobi preconditioning. This method is also called scaling due to the fact that the new matrix is unity on the diagonal, and less than one for all off diagonal terms. The two-sided form of Jacobi preconditioning is:

$$\left([D]^{-\frac{1}{2}}[A][D]^{-\frac{1}{2}}\right)\left([D]^{\frac{1}{2}}\{x\}\right) = [D]^{-\frac{1}{2}}\{b\}. \tag{4.7}$$

Where $[D]$ is simply the diagonal of $[A]$, *ie.* $D_{ij} = A_{(i)(i)}\delta_{ij}$. In effect the left and right preconditioners are, $[Q_L] = [Q_R] = [D]^{-\frac{1}{2}}$. Other methods include incomplete factorization methods, successive overrelaxation methods, and polynomial methods. Only the Jacobi preconditioning method is used throughout this work.

4.3 Direct vs. Iterative Solutions

Is the iterative method of solving equations any better than the direct method of solving a set of simultaneous equations? This is an important question when a programmer is faced with solving a large set of equations. In the next section, results of several comparisons are presented.

4.3.1 3-D Cavity Flow

The three-dimensional cavity problem is run with several different grids. In these runs the iterative preconditioner of the Jacobi type is used. Due to the large computer core required for solving three-dimensional problems, a frontal type solution is used as the direct solution method. The problem chosen for comparison is that of a three-dimensional driven cavity flow. The cavity is a cube with all side dimensions equal to one. The grids are of the sizes, $10 \times 10 \times 5$, $15 \times 15 \times 10$, and $20 \times 20 \times 15$. The flow is driven by a lid at y=1.0 as shown in Figure 4.3. The singular points, where the top of the cavity join the sides of the cavity, are assigned a velocity of $u = 1$, $v = 0$, and $w = 0$. The grid shown in Figure 4.3 contains 6000 elements, and 7056

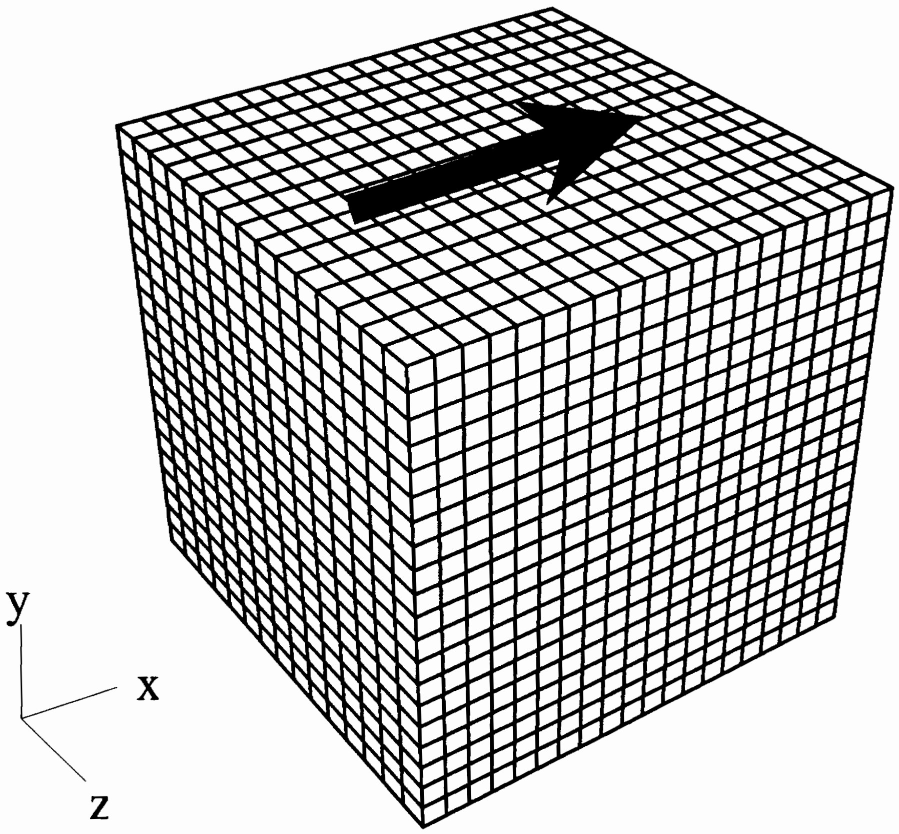

Figure 4.3: 3-D cavity grid

nodes (20 × 20 × 15).

In order to make comparisons between the direct and iterative solution methods the different size grids were run through 5 time steps. Such a test is universal in the sense that, for any Reynolds number, the same number of time steps will be required of the direct and iterative methods in order to obtain the steady state solution. The iteration type solutions have one parameter which determines if a solution is converged. The convergence

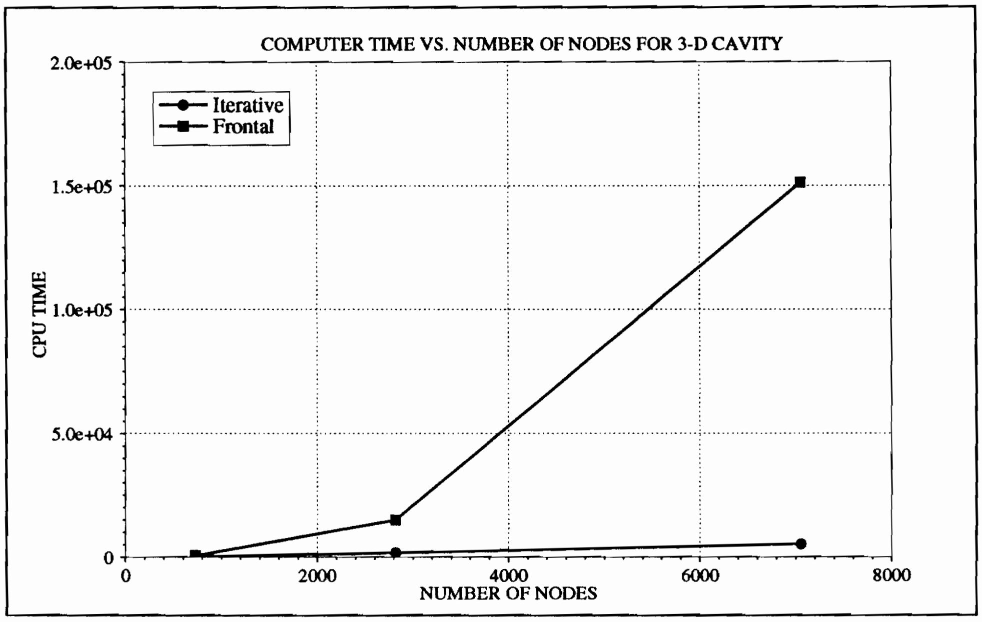

Figure 4.4: Computer solution times

parameter in the following example is expressed in equation 4.8.

$$\sqrt{\frac{r_i r_i}{b_j b_j}} = \zeta \leq 10^{-5} \tag{4.8}$$

Such high accuracy is important in obtaining good conclusions with regards to computer timings. Figure 4.4 shows the computer times for each grid and five time steps. The iteration method uses Jacobi preconditioning throughout the simulation. The computer times (in seconds) were taken from a NeXT computer. A further expansion of the plot of Figure 4.4 is shown in Figure 4.5. In both plots the nature of each solution method can be observed. The numerical data seems to validate the theory that the iterative conjugate gradient methods tend to follow a linear path of convergence, where the elimination methods tend to follow a nonlinear path [37] (see equation 4.9).

$$CPU_{time} = A \times N^{P}_{equations} + B \tag{4.9}$$

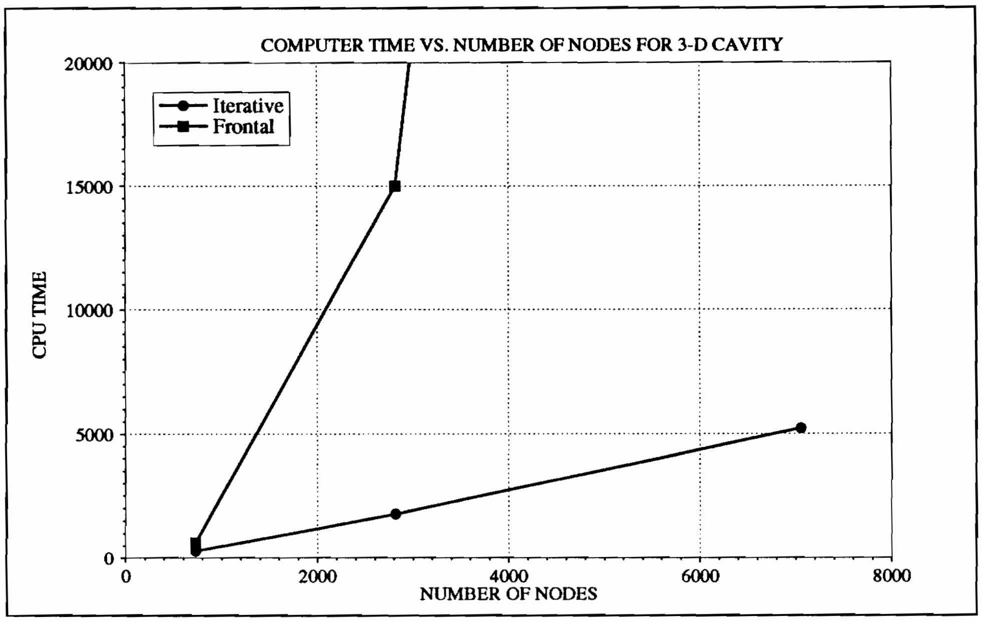

Figure 4.5: Computer solution times, expanded

Where $P \approx 1$ for conjugate gradient methods, and $P > 1$ for direct elimination methods. (Note: A and B are constants). Such behavior has also been shown previously by Reddy [68].

Results for a Reynolds number of 100 and 1000 are shown in Figures 4.6 thru 4.11. Figures 4.6 and 4.7 show the velocity vectors at the Z centerline plane in the cavity. The velocity distribution looks much the same as with two-dimensional flow. The Reynolds number 1000 flow tends to show a much smaller boundary layer on the impinging wall than the Reynolds number 100 case. Figures 4.8 and 4.9 also show this trend in the vorticity profiles. The pressure force driving this flow down the impinging wall is shown in Figures 4.10 and 4.15. The Reynolds number 1000 flow shows a very large pressure gradient down the wall, as compared to the Reynolds number 100 flow. This same distribution can be seen from Figures 4.12 and 4.13. Figures 4.14 and 4.11 show surfaces of constant pressure. Every point on the surfaces is of constant pressure. These isopressure surfaces show a pressure distribution which is definitely three-dimensional in nature. Pressure distributions such as these cause secondary recirculation zones at the bottom back of the cavity. This secondary recirculation zone is a result of the of a pressure gradient perpendicular to the direction of flow.

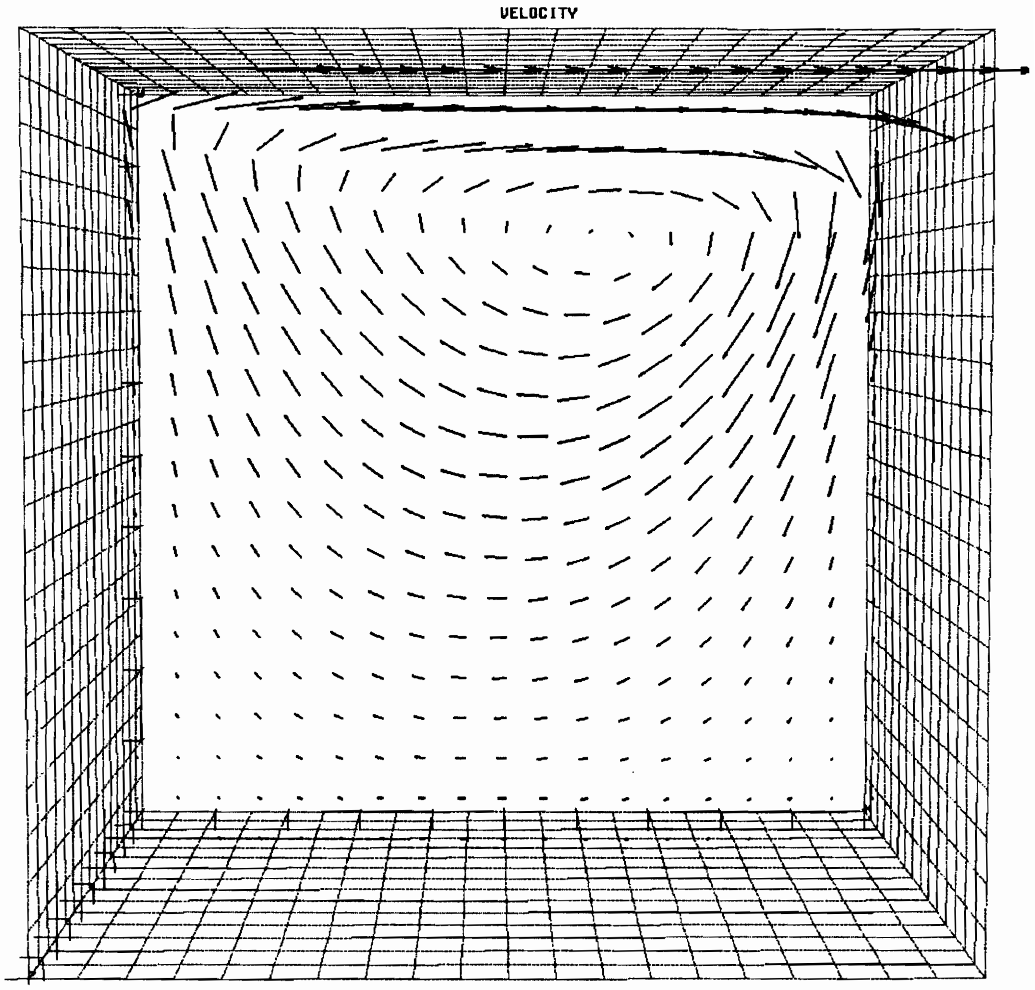

Figure 4.6: Velocity at Z=0.5 for Re=100

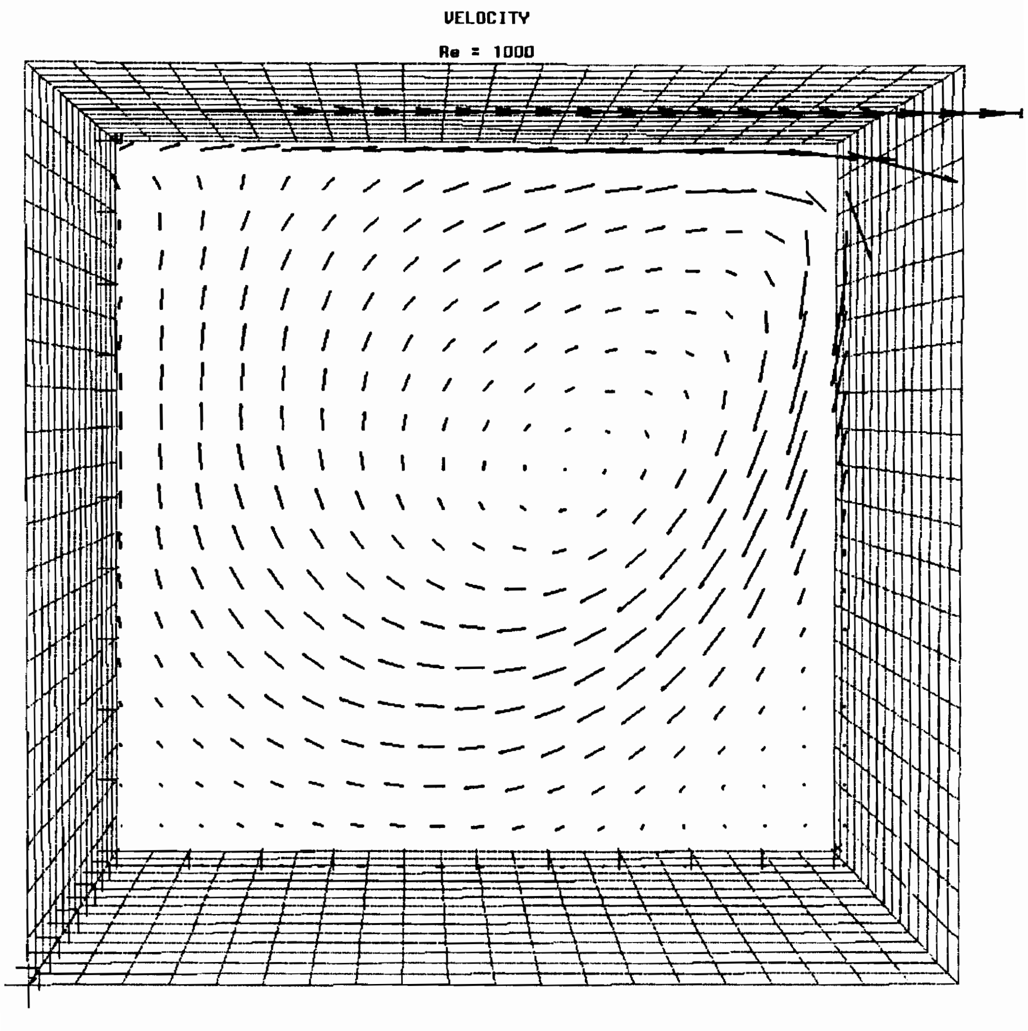

Figure 4.7: Velocity at Z=0.5 for Re=1000

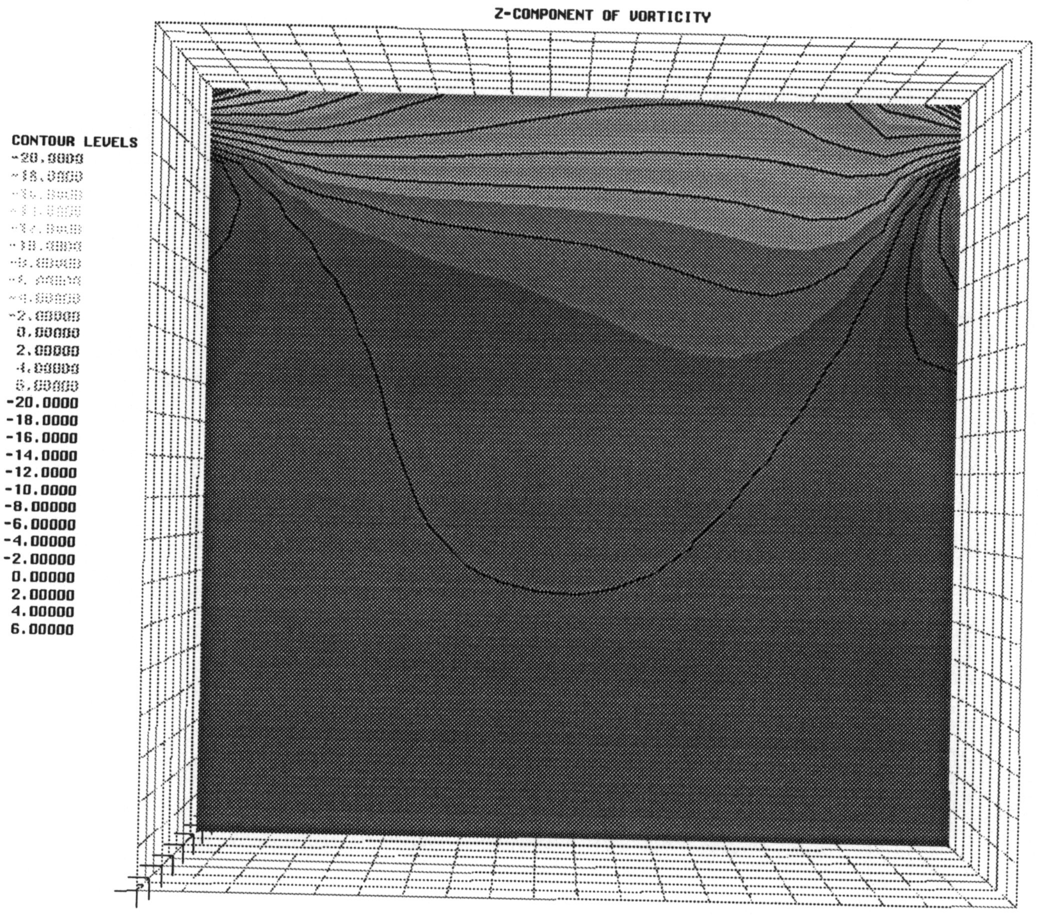

Figure 4.8: Vorticity at Z=0.5 for Re=100

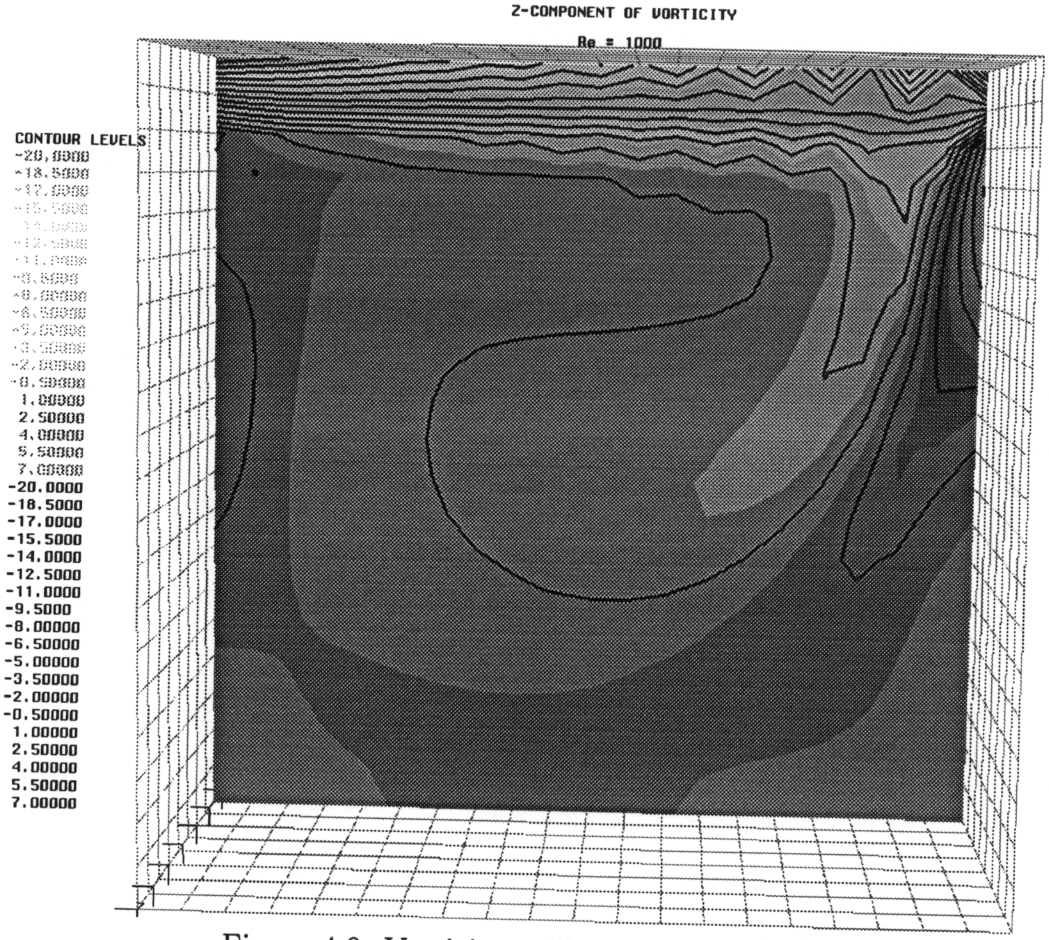

Figure 4.9: Vorticity at Z=0.5 for Re=1000

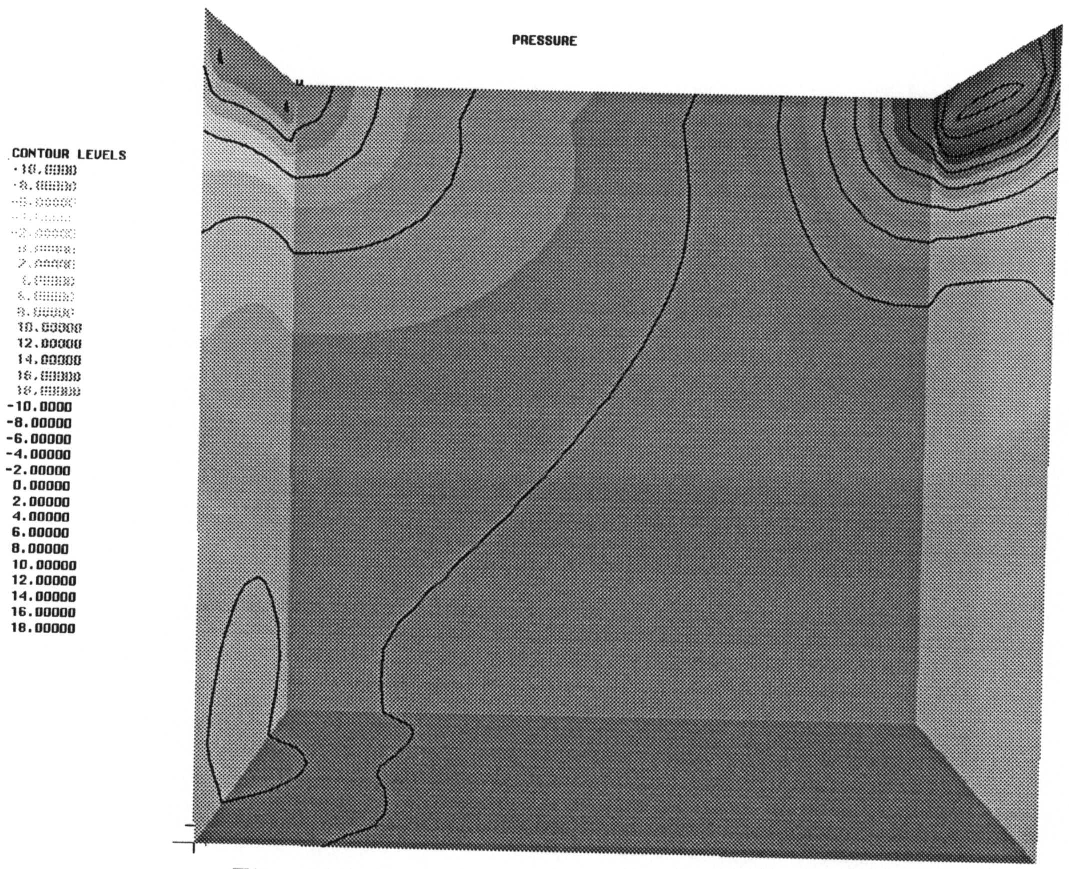

Figure 4.10: Pressure on walls for Re=100

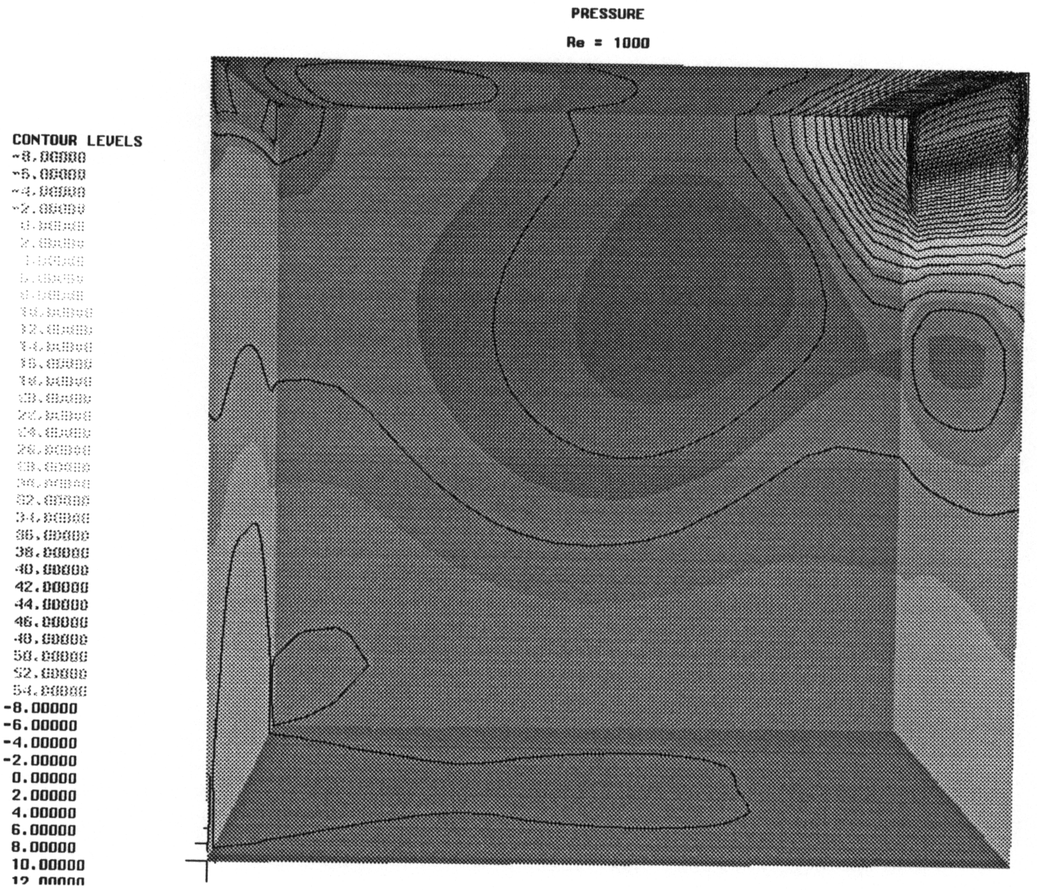

Figure 4.11: Pressure on walls for Re=1000

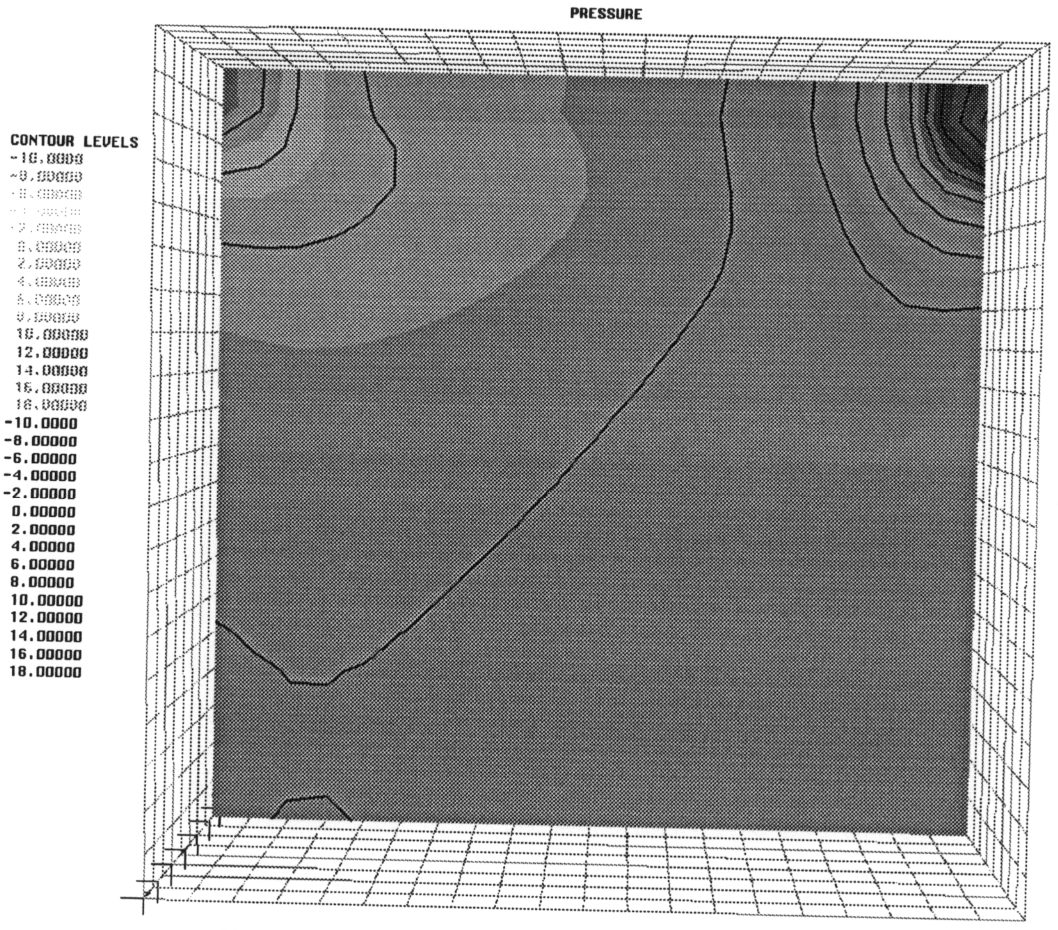

Figure 4.12: Pressure at Z=0.5 for Re=100

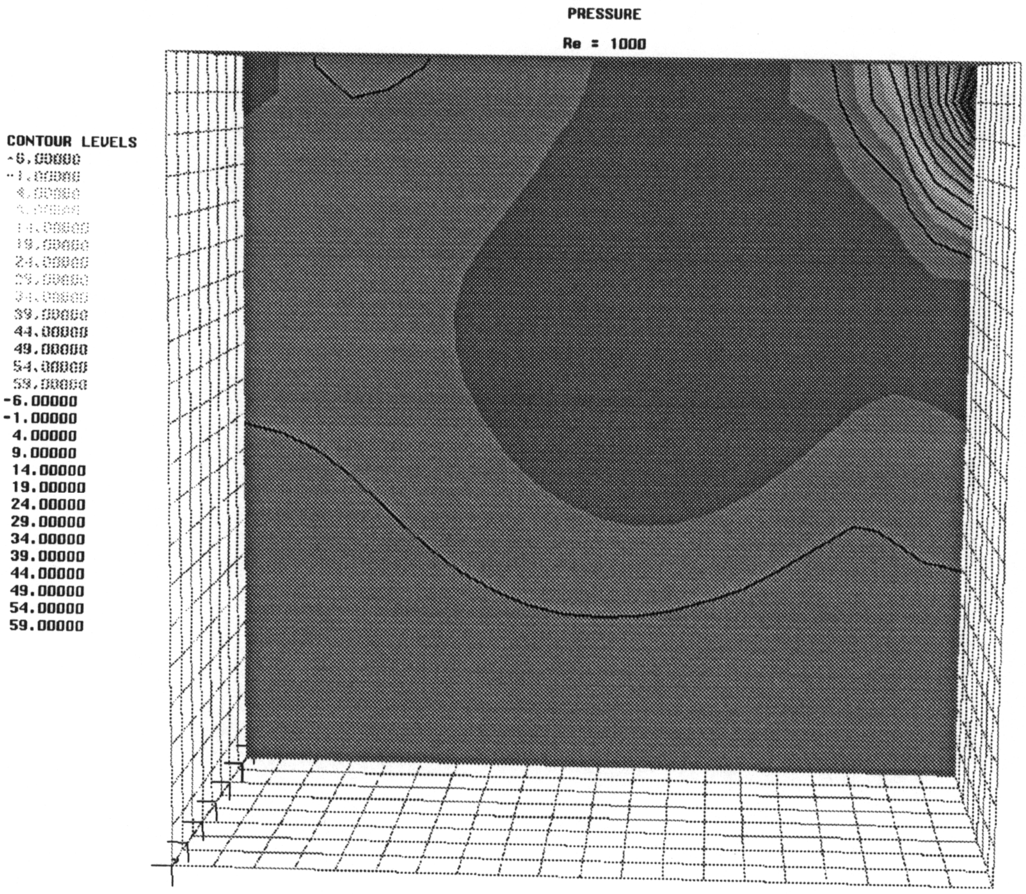

Figure 4.13: Pressure at Z=0.5 for Re=1000

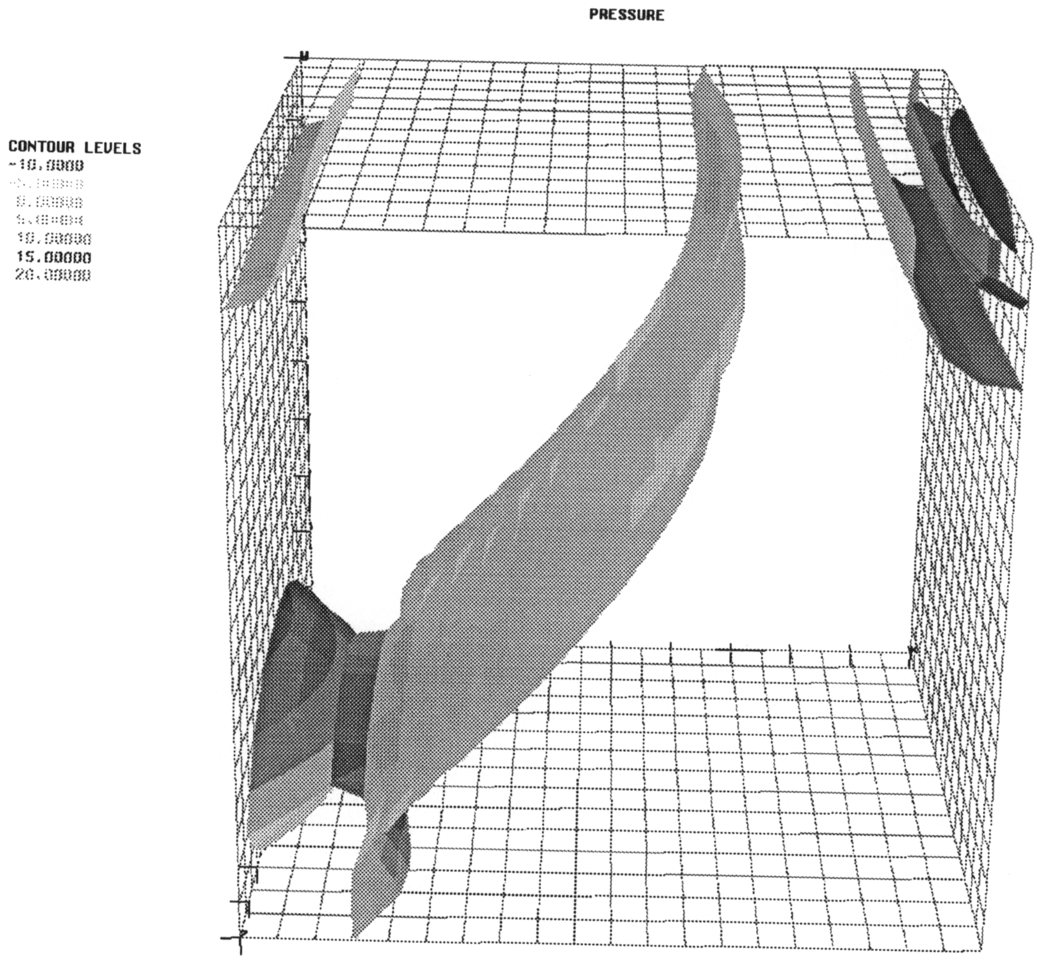

Figure 4.14: IsoPressure surfaces for Re=100

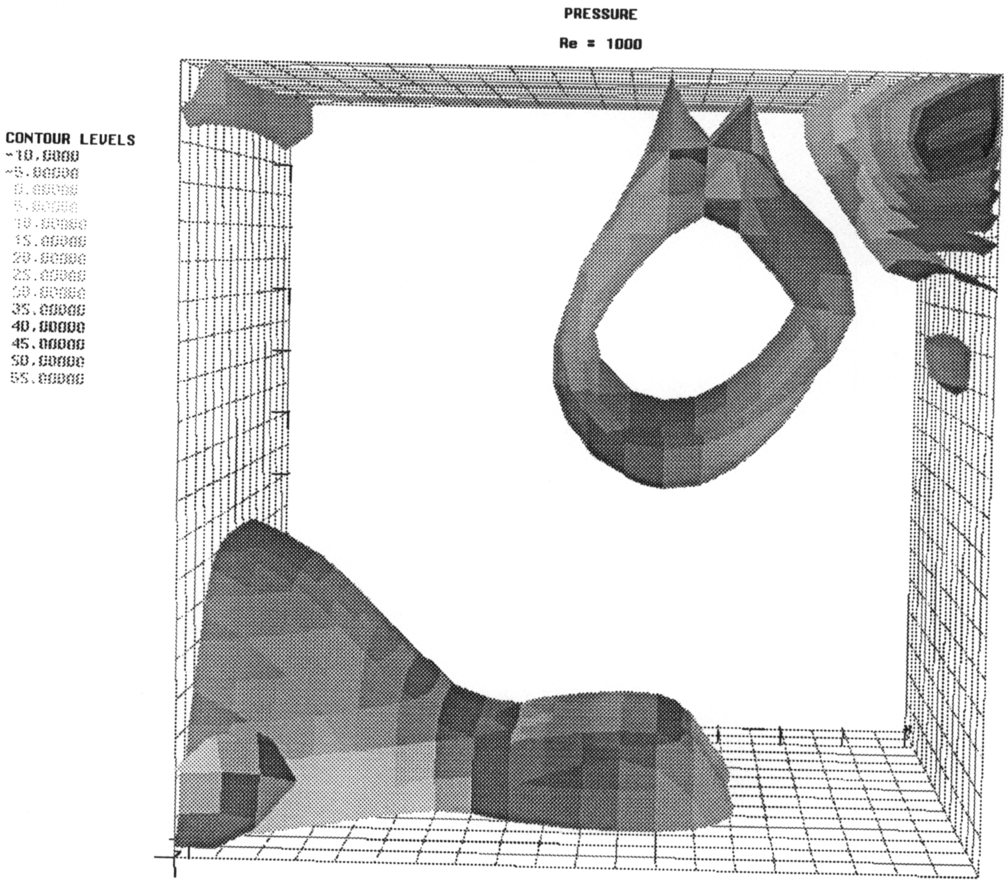

Figure 4.15: IsoPressure surfaces for Re=1000

4.3.2 3-D Curved Duct Flow

The next example involves flow through a curved duct. The Reynolds number based on the wetted perimeter is 900. The geometry is shown in Figure 4.16. The dimensions of the duct are 1 by $\frac{1}{2}$. The inlet and outlet sections of the

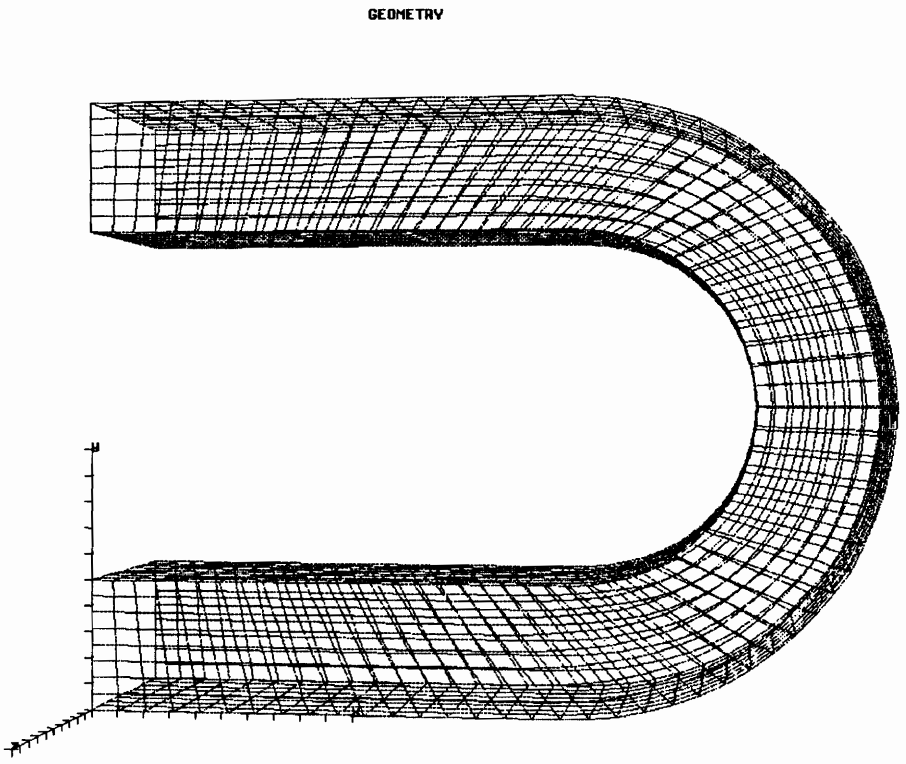

Figure 4.16: Grid for duct flow

duct are 1.8 units long, and the radius of curvature of the duct is $\frac{2}{3}$ units. The grid size is 9 x 9 x 71, or 5751 nodes, and 4480 elements. The inlet profile is a uniform distribution of velocity in the streamline direction, and zero velocity otherwise.

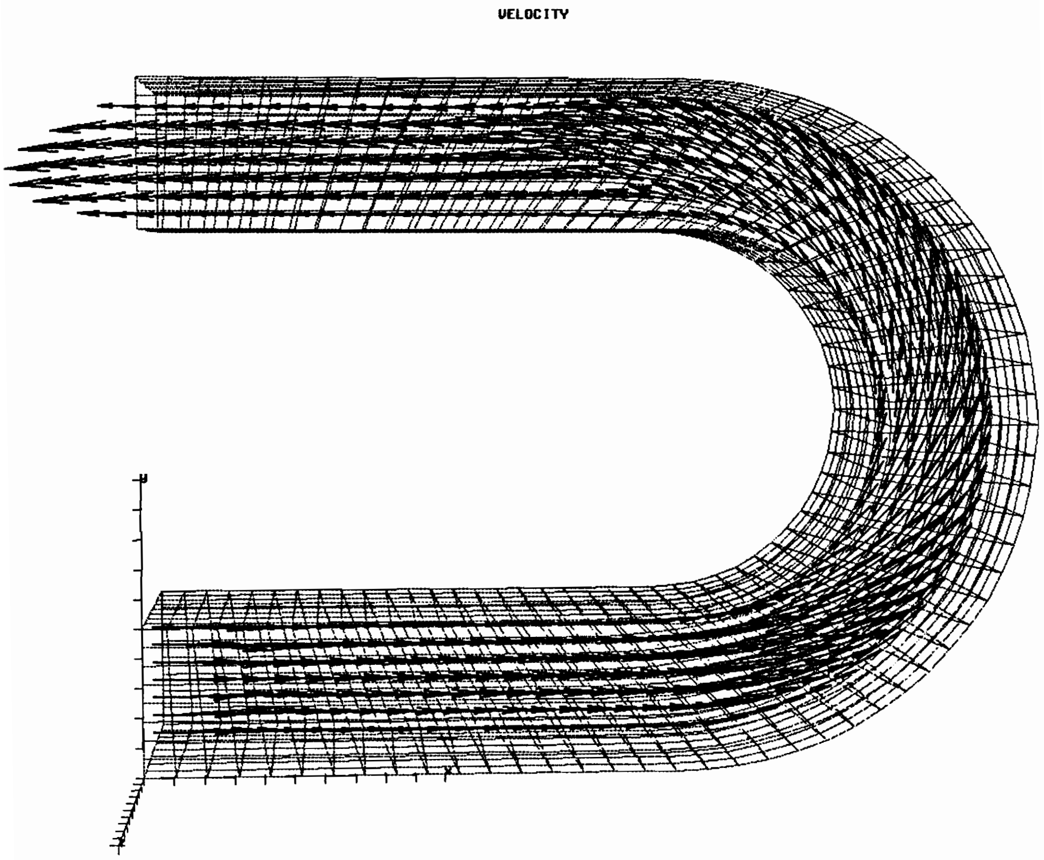

Figure 4.17: Centerline velocitys for duct flow

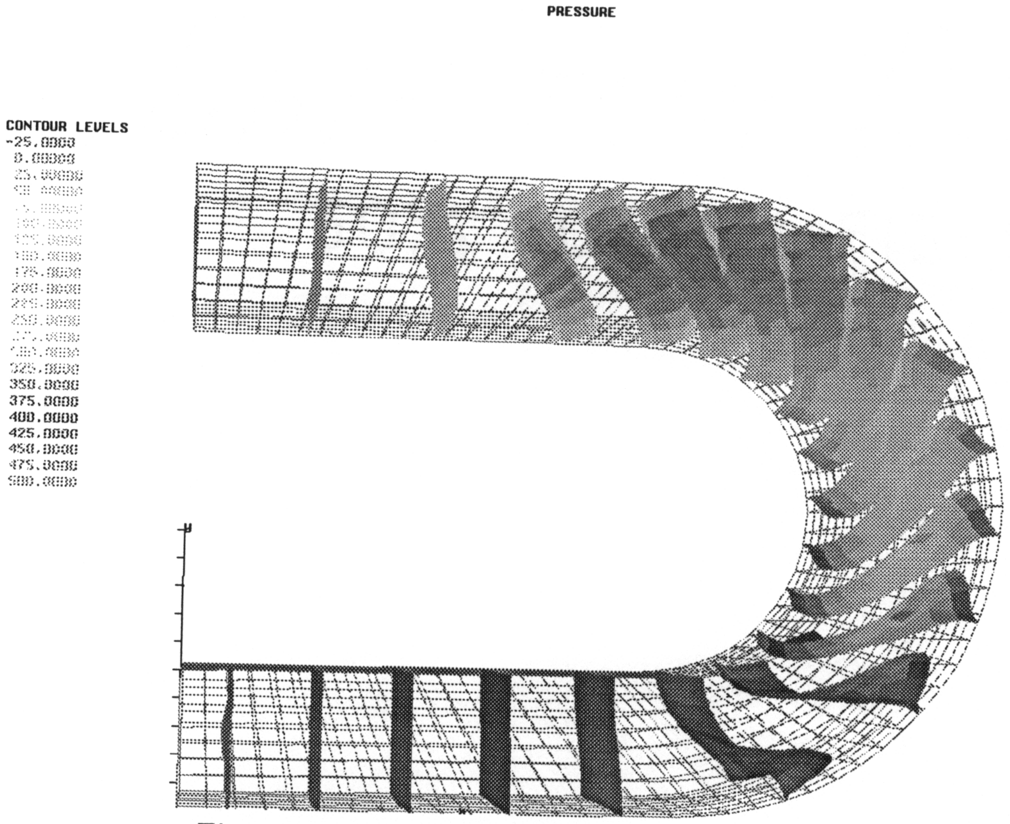

Figure 4.18: Isopressure Surfaces for duct flow

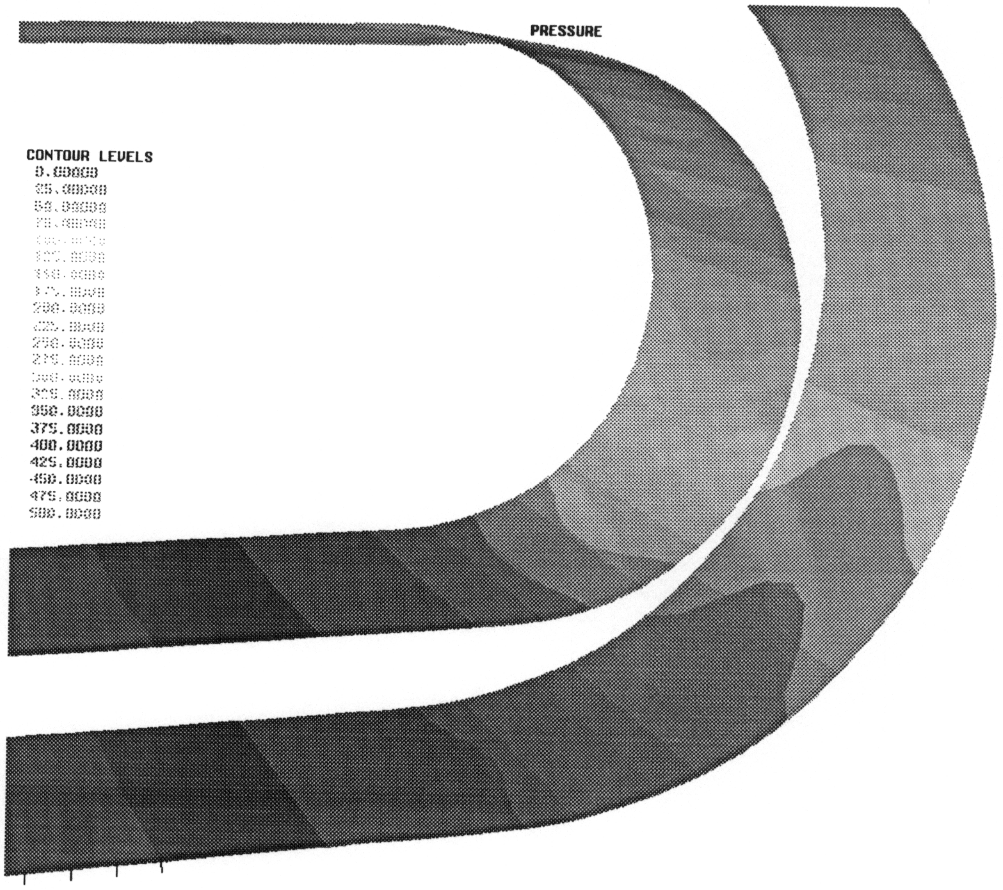

Figure 4.19: Pressure contours on duct walls

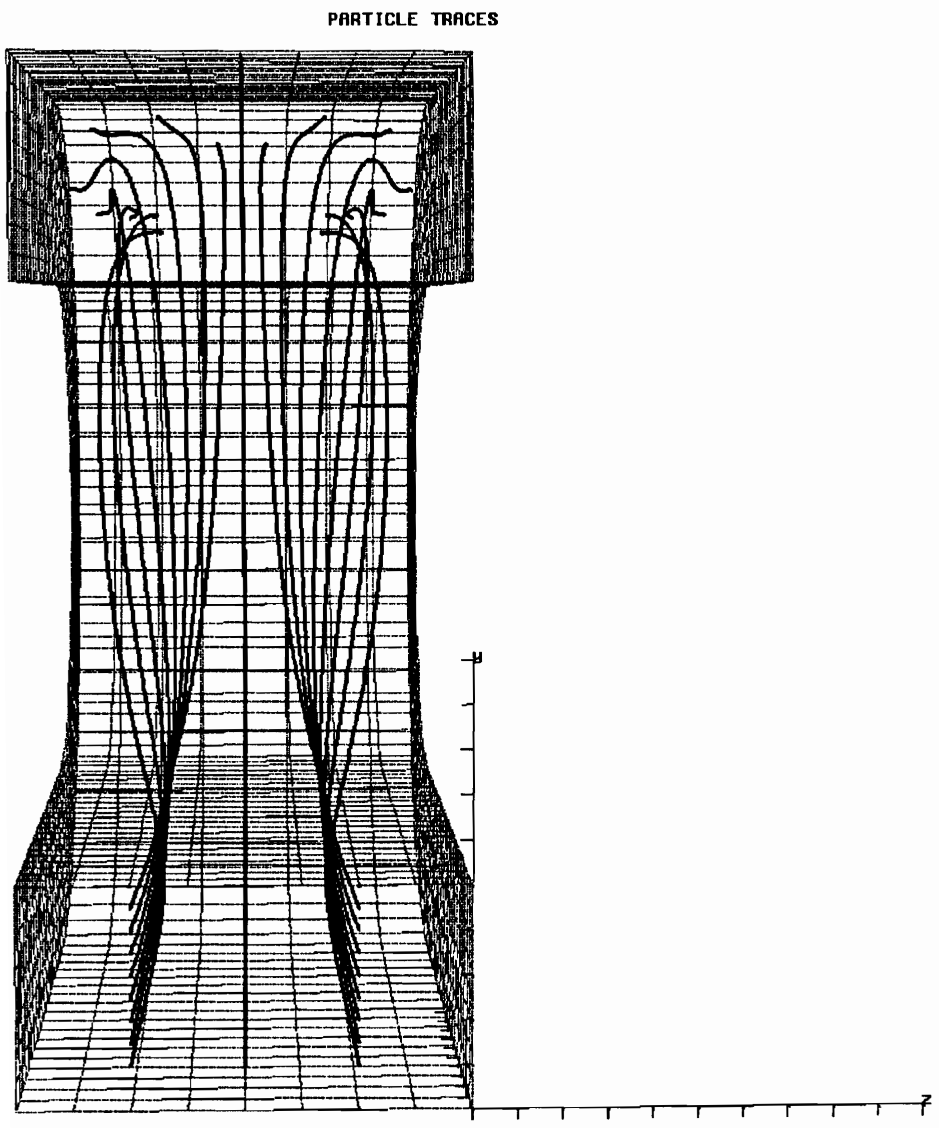

Figure 4.20: Particle traces through duct

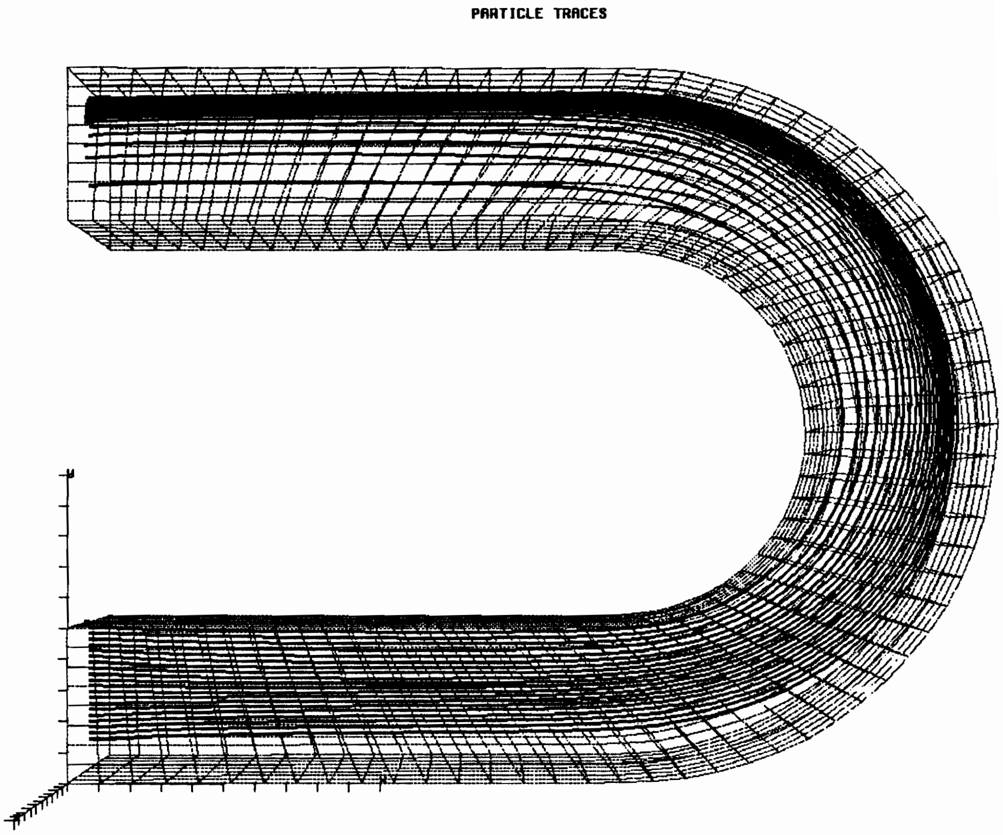

Figure 4.21: Particle traces down the centerline

The velocity down the center section of the duct is shown in Figure 4.17. Isopressure surfaces are shown in Figure 4.18. The shape of the pressure surfaces is a function of the distance down the duct. At the inlet and exit to the duct the pressure surfaces are flat. However, at the beginning of the curved portion of the duct, the pressure surfaces begin to distort and are no longer flat. This variation in pressure can be seen further in Figure 4.19, which shows a pressure variation perpendicular to the direction of flow. This pressure variation causes secondary recirculation zones down the length of the duct. This swirling flow can be seen in Figure 4.20, where a line of particles are sent into a spiraling pattern as they flow down the duct. Figure 4.20 is produced by performing a 2^{nd} order Runge–Kutta integration of particles under the influence of the velocity field. Figure 4.21 is produced this same way. In this figure the particles down the centerline of the duct are "pushed" towards the outer side of the duct as they flow downstream.

Chapter 5

P-Adaptive Solution Method

In the traditional finite element method, higher order elements are obtained through two types of interpolation, either Lagrange or Hermitian elements. The reasons for the popularity among Lagrange and Hermitian element formulations are the ease in which interelement continuity can be obtained and the significance of the basis function coefficients. Any order element may be traced back to Pascal's triangle where all possible monomial combinations are listed. If the basis functions are grouped into levels of increasingly higher polynomial orders, these basis functions may be used in p-adaptive finite element methods. In the present work tensor products of one dimensional basis functions will be used to generate two and three dimensional basis functions. With this in mind, it is instructive to look at a few one-dimensional examples. In Figure 5.1 linear Lagrange basis functions are shown. Figures 5.2 and 5.3 show quadratic and cubic Lagrange basis functions.

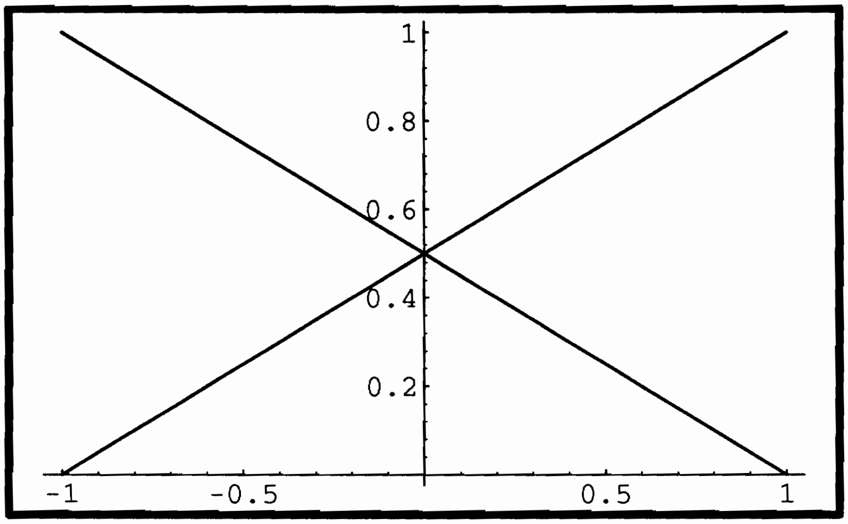

Figure 5.1: Linear Lagrange basis functions

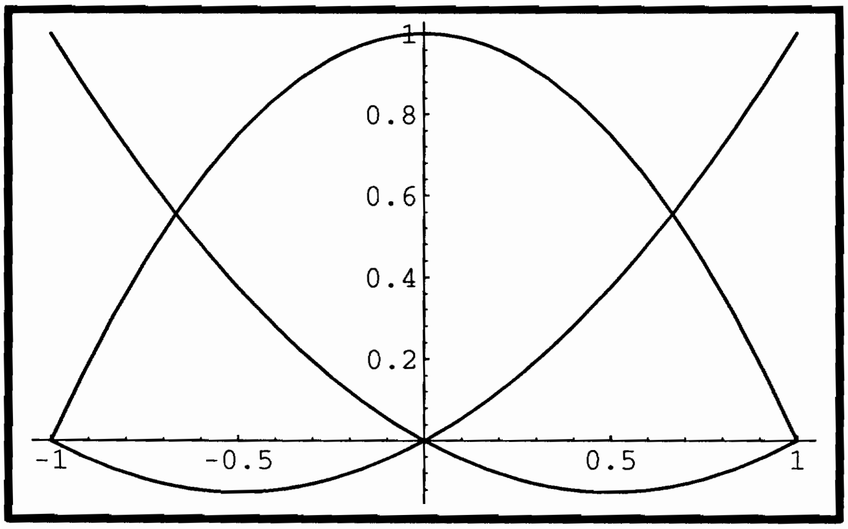

Figure 5.2: Quadratic Lagrange basis functions

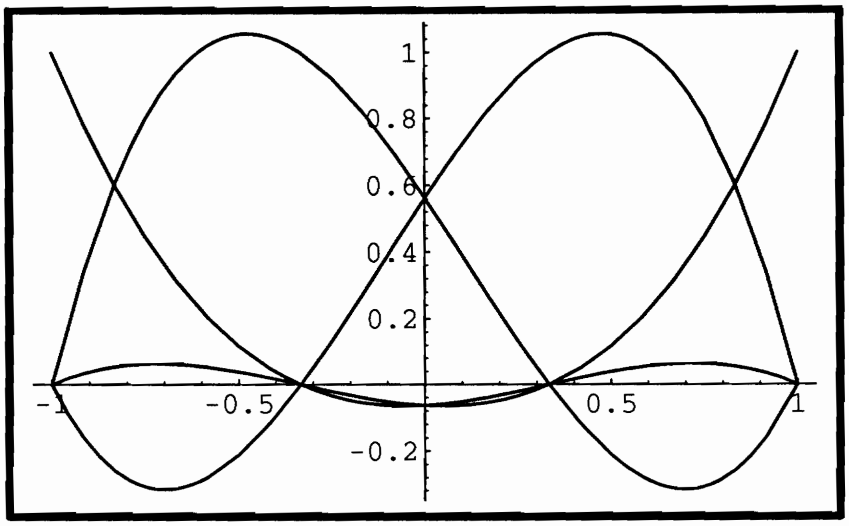

Figure 5.3: Cubic Lagrange basis functions

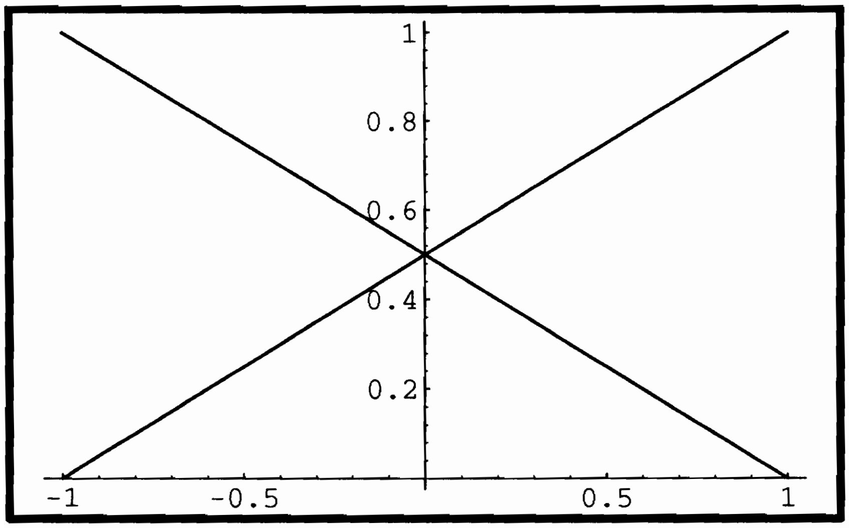

Figure 5.4: Linear basis functions

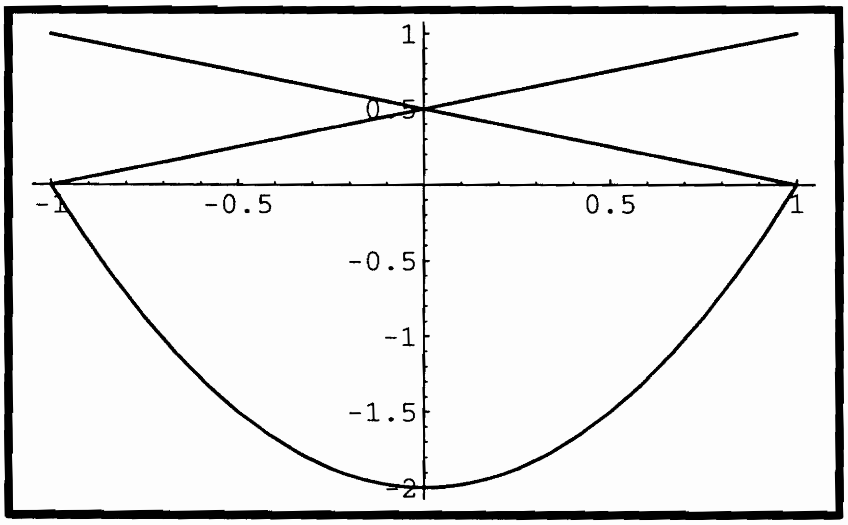

Figure 5.5: Quadratic basis functions

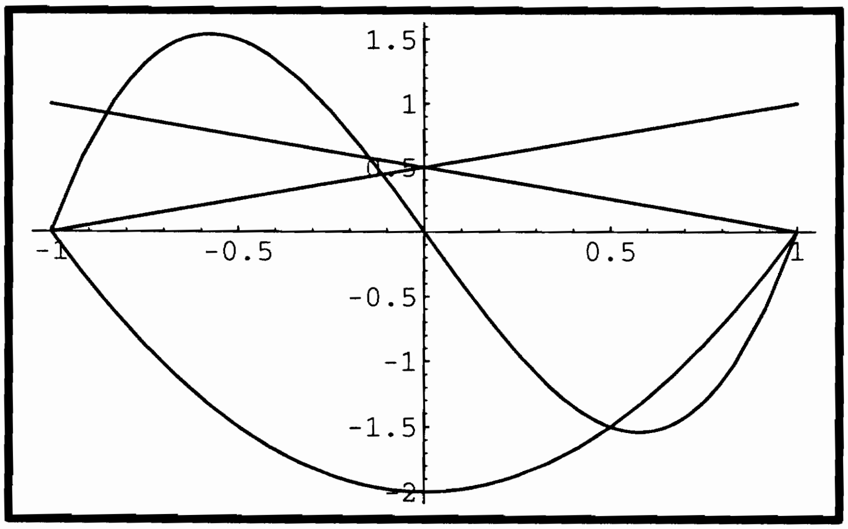

Figure 5.6: Cubic basis functions

Heiarachial basis functions prove very useful in the construction of the p-adaptive basis functions. Three examples are shown in Figures 5.4, 5.5, and 5.6. The overall approach to obtain heiarachial basis functions is the same. Start with the linear Lagrange basis functions, Figure 5.4, add a new basis function which is zero at the ends of the element, and increases the polynomial order by one. In this type of approach, all lower order basis functions will directly exist in a higher order element. For example, the set of basis functions in Figure 5.6 contain all basis functions existing in Figure 5.5. However, in the Lagrange approach this embedding of the lower degree basis functions within higher order basis functions does not exist. One advantage to such an heiarachial approach is the ease in which higher order solutions may be obtained. If a solution is obtained using linear basis functions throughout all elements, then going to second order fuctional representation simply means the addition of one polynomial degree per element. An approximate solution will easily exist in the form of the linear solution plus zero times the quadratic basis functions. Several forms of the heiarachial basis function exist. The only requirements that added basis function must satisfy is they must be zero at the ends of the elements, complete, and linearly independent.

As with two–dimensional Lagrange elements, two–dimensional heiarachial elements are formed from the tensor product of one–dimensional basis functions. Such tensor products give nine basic node types. The first four nodes will be on all four corners of the quadrilateral element. These four nodes all have one degree of freedom per node, and give the four basis functions of a linear Lagrange element as shown in Figure 5.7. The second type of node positions are those along the side of the element. These nodes can

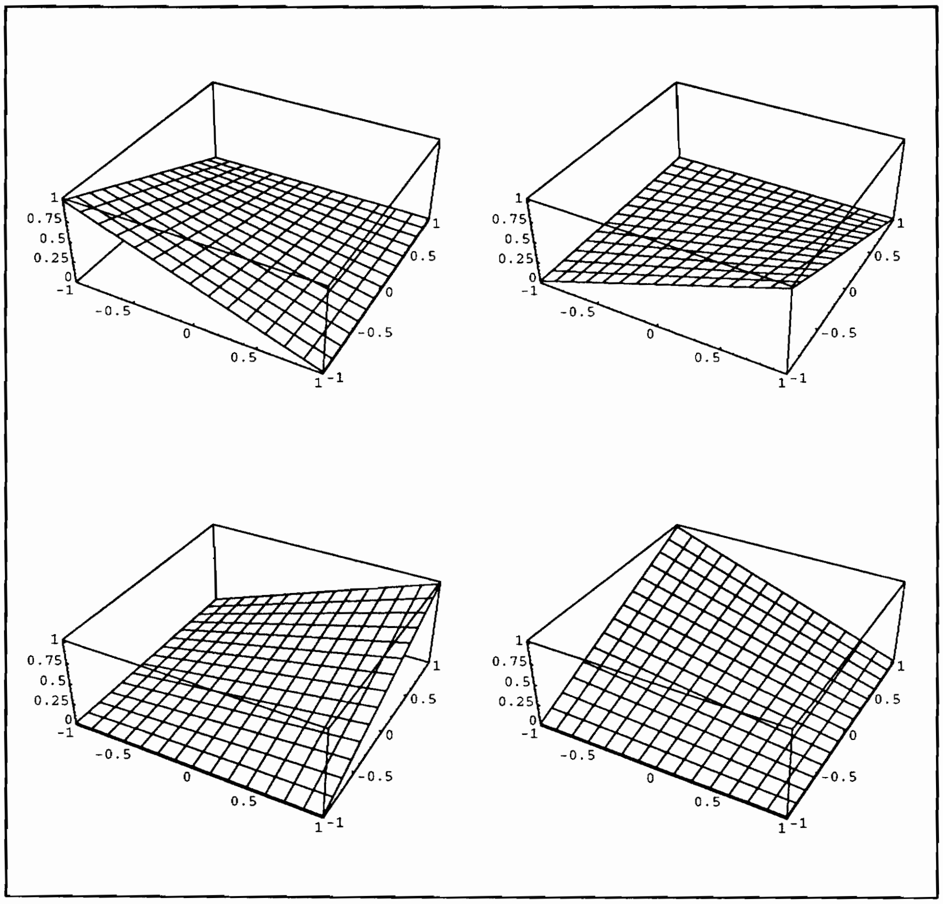

Figure 5.7: First four two-dimensional basis functions

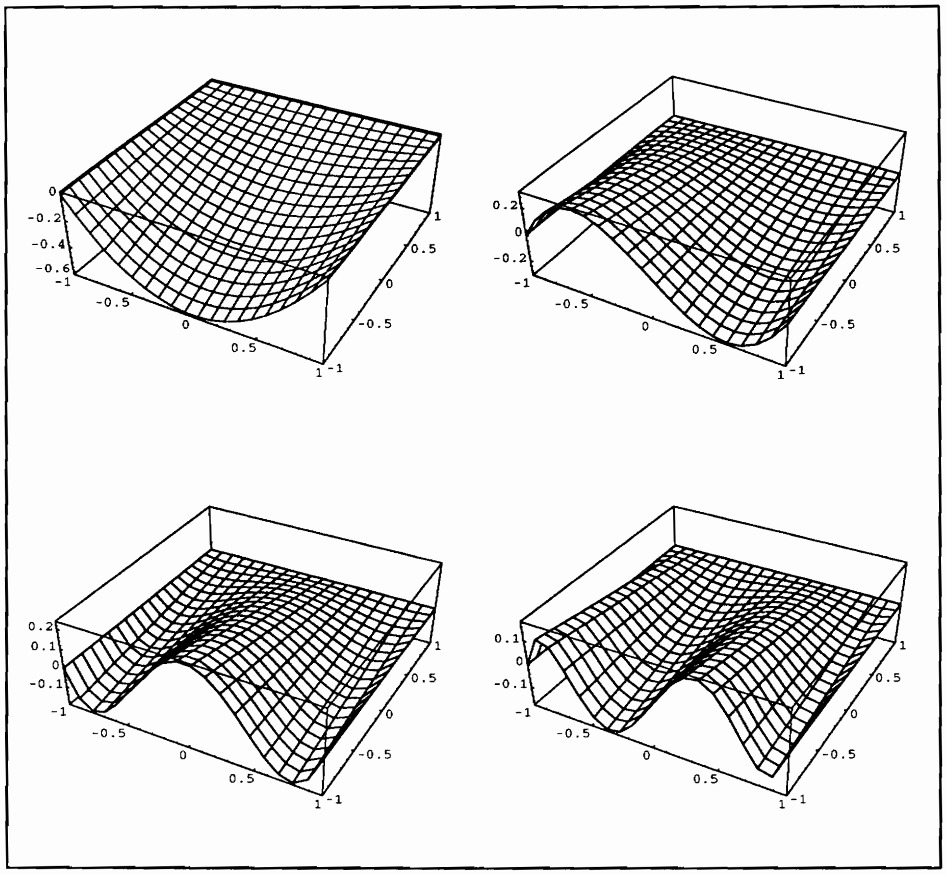
Figure 5.8: Two-dimensional side basis functions

have many degrees of freedom. However, these degrees of freedom may or may not have any physical meaning. Four of these basis functions are shown in Figure 5.8. The basis functions on the other three sides can be seen by mentally rotating the figures 90 degrees at a time in the plane of the element. The final group of basis functions are those associated with only the internal degrees of freedom. These basis functions come from the tensor products of one-dimensional basis functions of second order or higher. Some of the

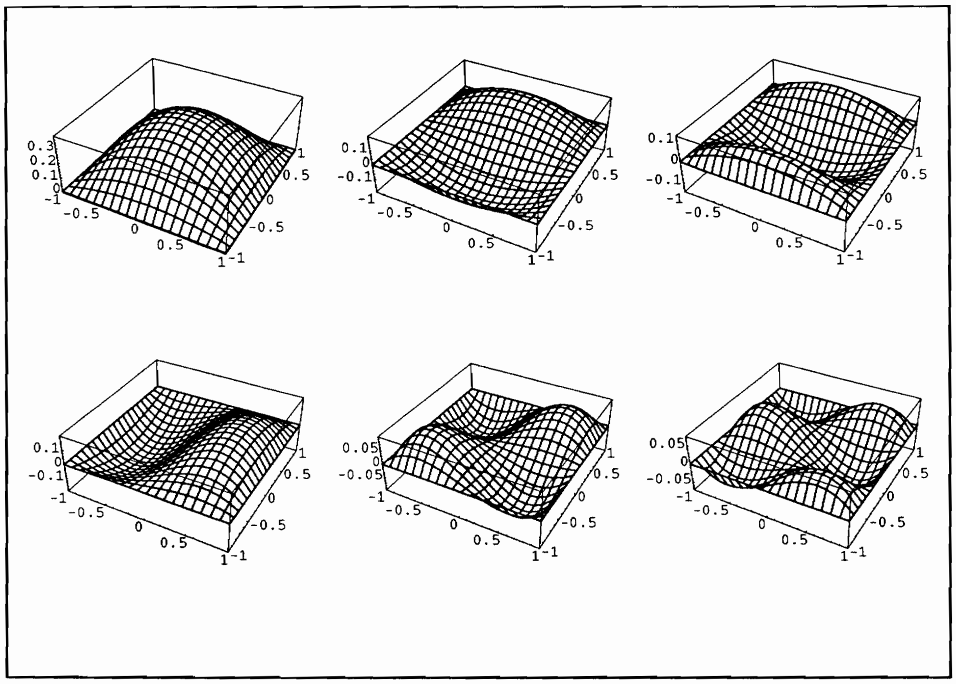

Figure 5.9: Two-dimensional internal basis functions

endless numbers of these basis functions are shown in Figure 5.9

The available degrees of freedom are shown in Table 5.1. This table shows the degrees of freedom for each node shown in Figure 5.10. From this table can be seen the following: if the polynomial order of the basis function is 4^{th} order, then nodes 1 to 4 have one degree of freedom; nodes 5 to 8 have 3 degrees of freedom; and node 9 has 9 degrees of freedom. The actual degrees of freedom for nodes 5 to 9 may not hold any physical significance. However, one can choose to give them significance by selecting the higher order basis function to be such that the derivatives are given at the side and center nodes. Throughout this work the nodal degrees of freedom will not be given any specific physical significance. The basis functions used are those

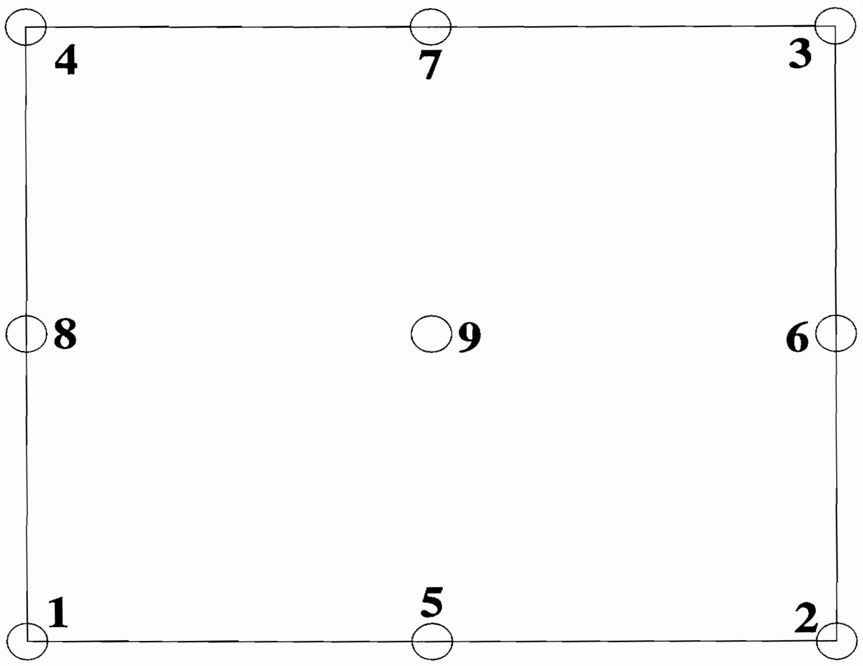

Figure 5.10: Two-dimensional nodal numbering

Table 5.1: Degrees of freedoms vs polynomial order

	Node number								
Order	1	2	3	4	5	6	7	8	9
1	1	1	1	1	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1
3	1	1	1	1	2	2	2	2	4
4	1	1	1	1	3	3	3	3	9
5	1	1	1	1	4	4	4	4	16
6	1	1	1	1	5	5	5	5	25
$\vdots$	1	1	1	1	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

suggested by Devloo [70].

$$\psi_1 = \frac{1}{2}(1-\xi) \tag{5.1}$$

$$\psi_2 = \frac{1}{2}(1+\xi) \tag{5.2}$$

$$\psi_n = T_n(\xi) - \psi_2(\xi) - (-1)^{n-1}\psi_1(\xi). \tag{5.3}$$

Where $T_n(\xi)$ is the Chebyshev polynomials defined by:

$$T_0(\xi) = 1 \tag{5.4}$$

$$T_1(\xi) = \xi \tag{5.5}$$

$$T_{n+1}(\xi) = 2\xi T_n(\xi) - T_{n-1}(\xi) \qquad \forall\, n \geq 2. \tag{5.6}$$

These basis functions (up to fifth order) are shown in Figure 5.11. This type of functional representation has been used to a considerable extent in the area of spectral methods [71].

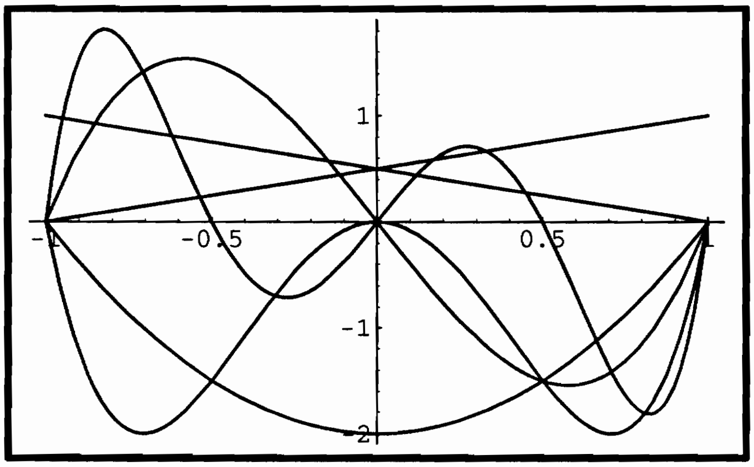

Figure 5.11: Chebyshev basis functions

5.1 Interelement continuity

In order to obtain an optimal representation of the given dependent variable over the domain, each element must be capable of obtaining its own polynomial level. This brings up the question: "How should two adjacent elements have differing polynomial order, while maintaining interelement continuity?". This can be answered by allowing the adjoining elements to have the same number of degrees of freedom at the joining side node. For example, two elements side by side are shown in Figure 5.12. The element on the left is of order 2 and the element on the right is of order 5. From Table 5.1 we see for order 2 the side nodes has 1 degree of freedom, and for order 5 the side nodes have 4 degrees of freedom. To assure interelement continuity we must assign the joining node between the two elements to have the same number of degrees of freedom. In the present work all lower order elements will be increased up to the higher order elements. In the example shown in

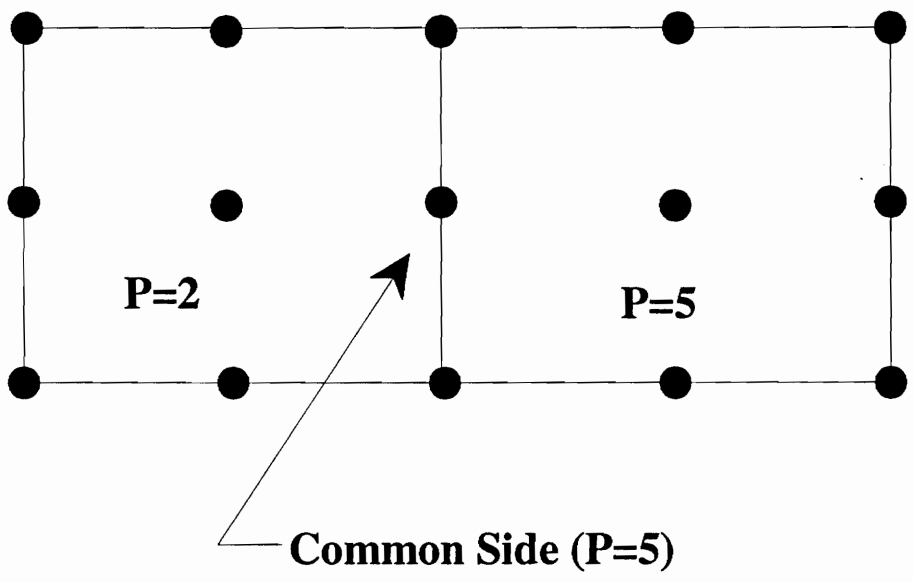

Figure 5.12: Two adjacent elements

Figure 5.12, the common node will be assigned 4 degrees of freedom. The functional variation of such an example is shown in Figure 5.13. This plot shows the velocity corrections at time equal to 20 seconds. In order to dynamically change the degrees of freedom during a particular run, the data structure of Demkowicz and Oden is used [72].

5.2 Backstep Flow

In this section flow over a backstep is investigated. The geometry and conditions of Figure 3.7 are used. The grid for this particular case is shown in Figure 5.14. Notice the difference between this grid and the one used in

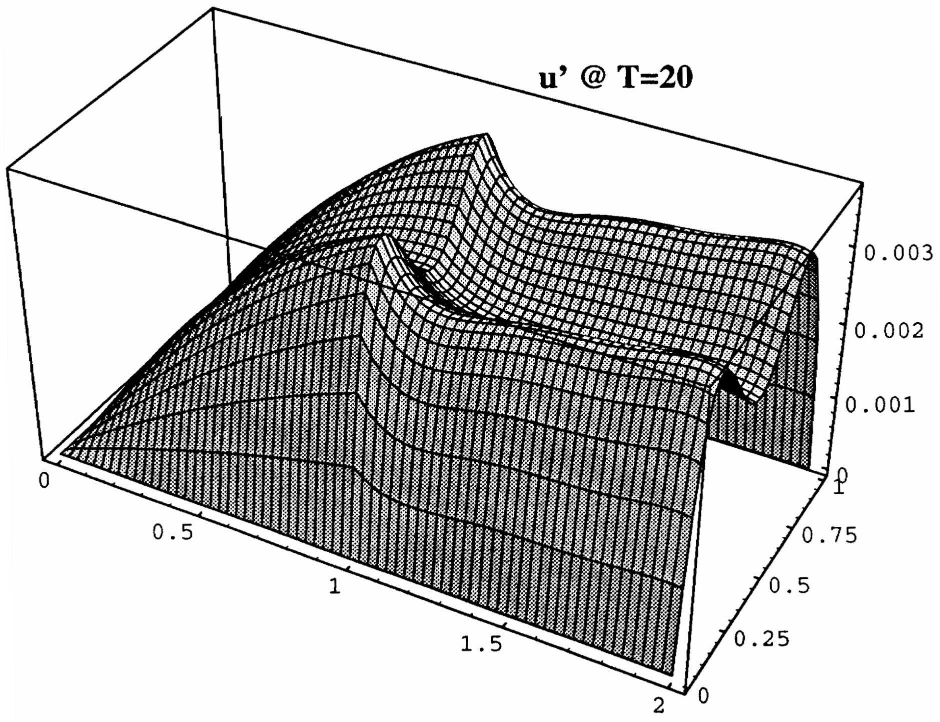

Figure 5.13: Example of interelement continuity

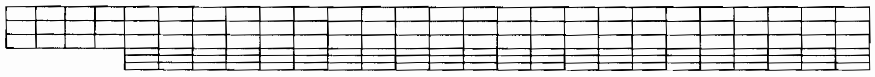

Figure 5.14: Grid for backstep problem

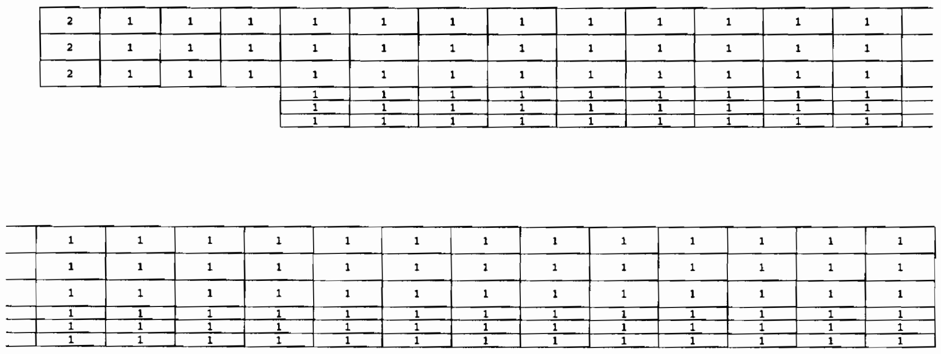

Figure 5.15: Grid with polynomial orders (step 1)

Figure 3.8 (Chapter 3) is that the grid used in this example is very coarse.

The first step of this backstep solution is the selection of polynomial orders. The lowest possible values are selected, giving the polynomial order shown in Figure 5.15. In this figure the numbers within each element are the polynomial orders. Note: the elements at the inlet were selected to be second order to allow the exact parabolic inlet distribution to be specified.

Through the adaptive process, the grids in Figures 5.16 and 5.17 were obtained. Notice the increase in the polynomial order of the elements. The velocity vectors from the first solution are shown in Figure 5.18. These velocity vectors are in the vicinity of the backstep and are shown at each

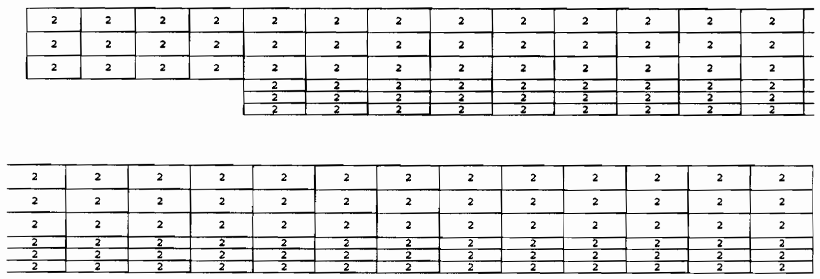

Figure 5.16: Grid with polynomial orders (step 2)

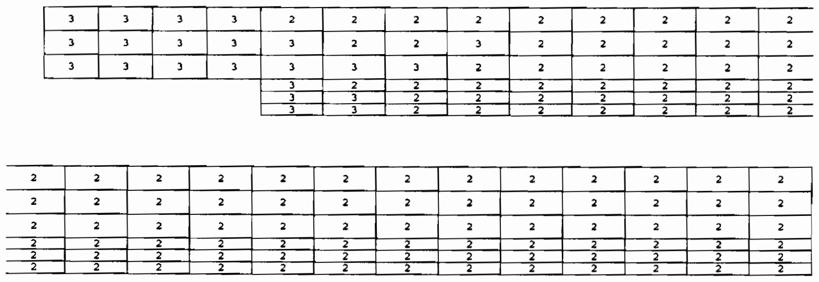

Figure 5.17: Grid with polynomial orders (step 3)

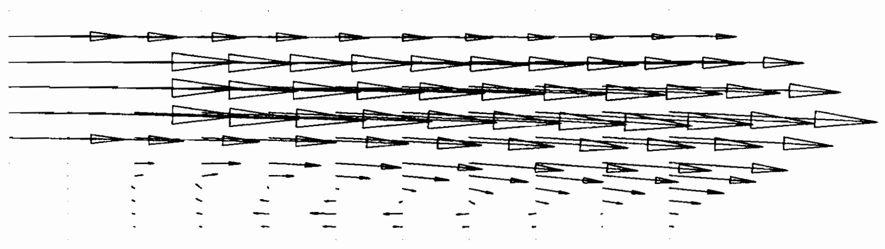

Figure 5.18: Velocties over backstep

node within the elements. In Figure 5.18 the general quantitative nature of the flow can be seen. However, the solution may not be as exact as expected. So a very good question is: "What elements need their polynomial order increased to obtain a more accurate solution?". To answer this question an error estimate must be obtained within each element. This in turn will supply a guide to polynomial refinments. There are two equally valid approaches to obtain an error estimate. The a priori method seeks to obtain an error analysis before the next solution is completely finished, while the a posteriori method seeks to estimate the errors after the next solution step is taken. In the present example, the a priori error is the difference between two solutions, the first being complete, and the second being an incomplete solution with an increase in the polynomial order. The a posteriori method is executed by comparing the difference between two successive complete solutions. This process of convergence can be seen in Table 5.2. These tables can be seen in graphical form in Figures 5.19, and 5.20. In the table and figures the L_2 error per unit area is shown. This error is actually the L_2 measure between the two solutions, either an incomplete one (a priori) or a complete one (a posteriori). With these facts in mind we proceed to display the errors

Table 5.2: L_2 error per unit area

L_2 per unit area				
Solution Step		1	2	3
A priori	U	0.205539	0.021029	0.011045
	V	0.051994	0.012853	0.005635
	P	32.49240	1.196550	1.023490
A posteriori	U	0.575857	0.093857	...
	V	0.078205	0.015796	...
	P	41.38130	8.534350	...

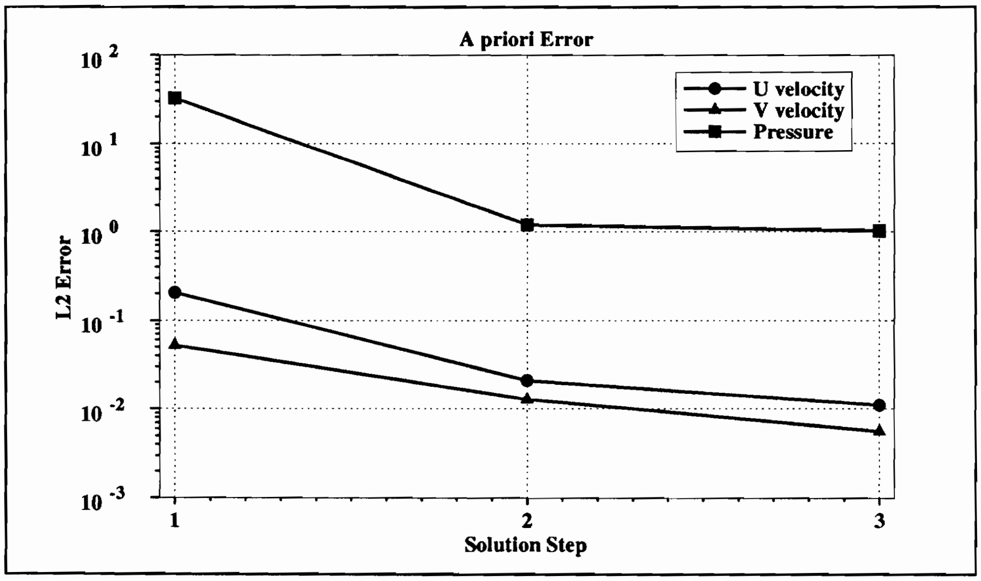

Figure 5.19: A priori errors

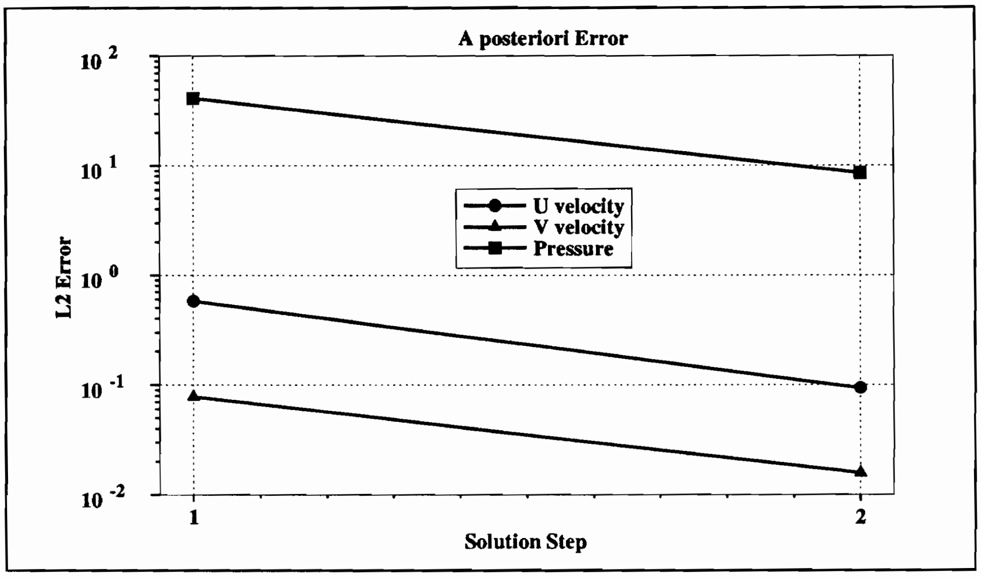

Figure 5.20: A posteriori errors

graphically. Figures 5.21, 5.22, and 5.23 show the a priori errors in u–velocity. Notice the error decreases as the polynomial order increases. Figures 5.24, 5.25, and 5.26 show the a priori errors in v–velocity and Figures 5.27, 5.28 and, 5.29 show the a priori errors in pressure. Notice all these errors decrease as the polynomial order increases.

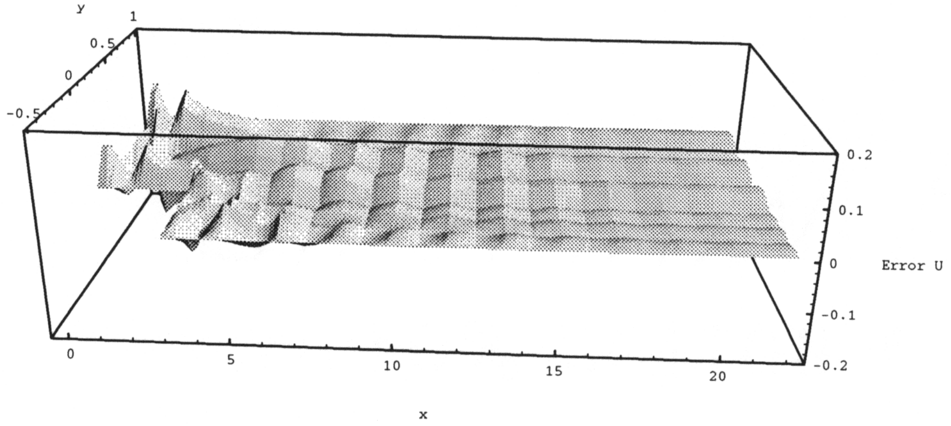

Figure 5.21: A priori errors in U, (step 1)

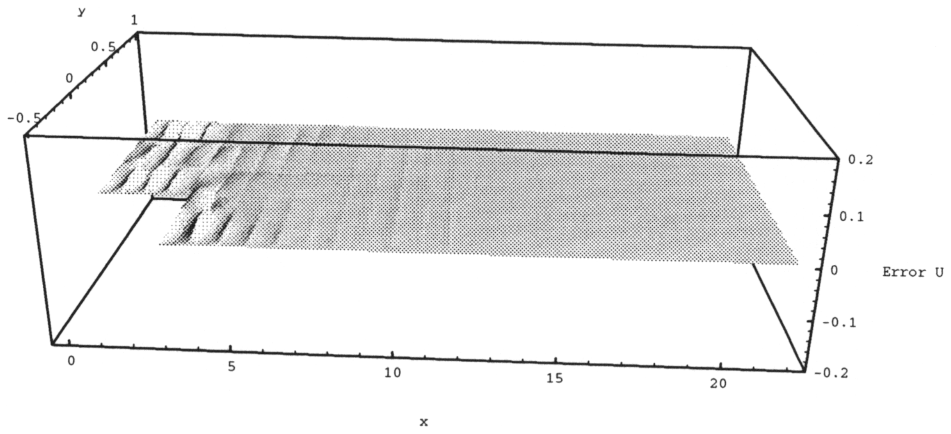

Figure 5.22: A priori errors in U, (step 2)

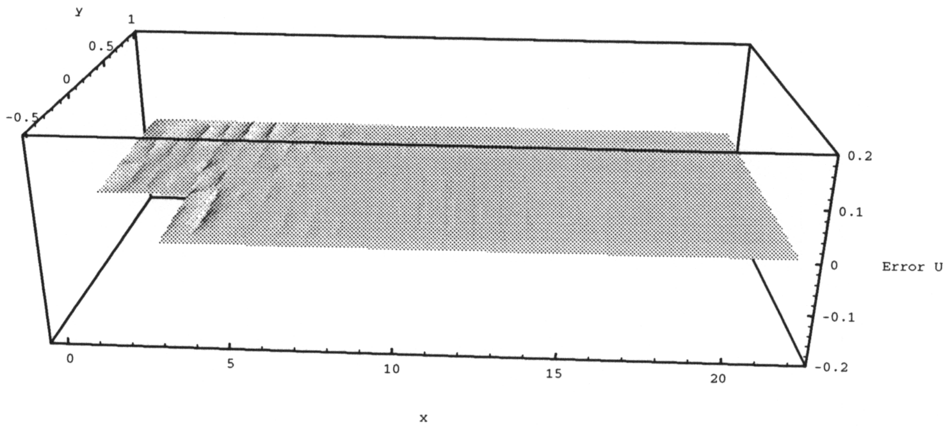

Figure 5.23: A priori errors in U, (step 3)

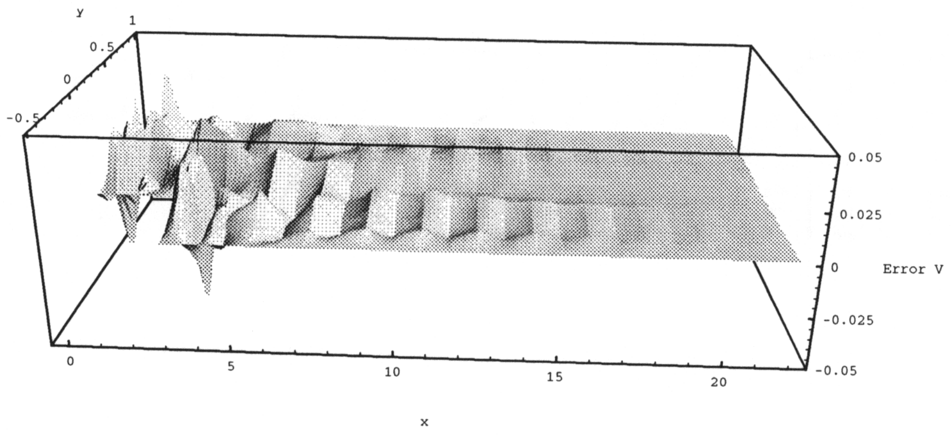

Figure 5.24: A priori errors in V, (step 1)

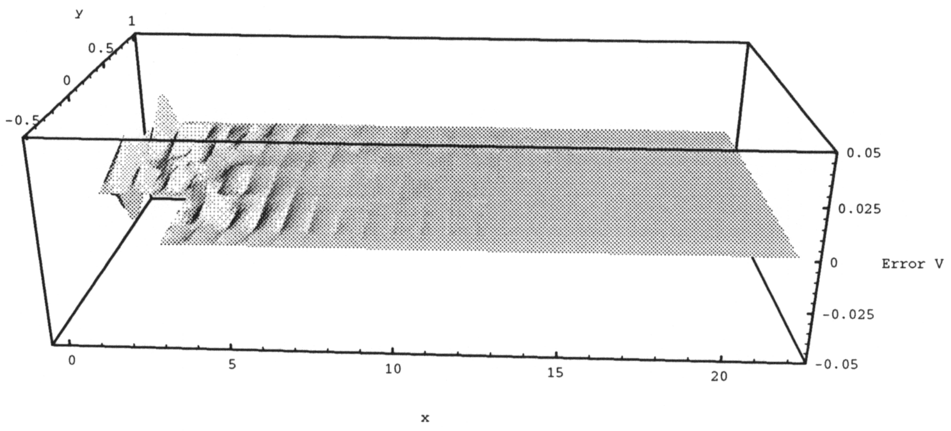

Figure 5.25: A priori errors in V, (step 2)

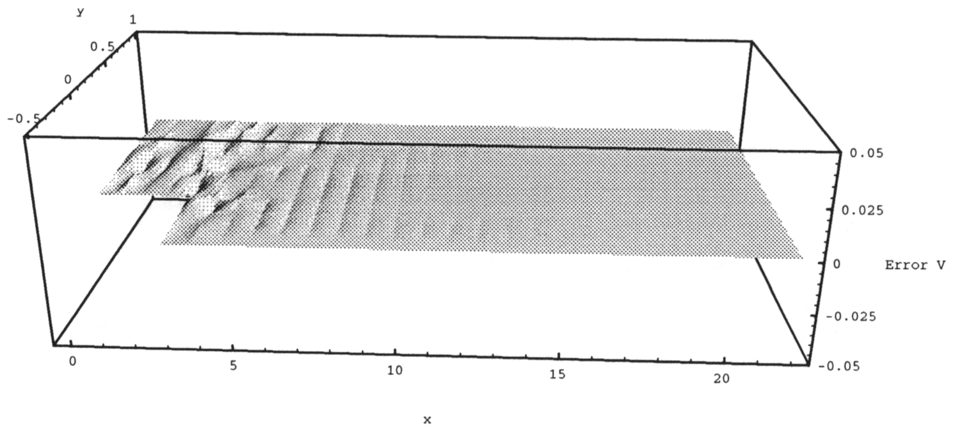

Figure 5.26: A priori errors in V, (step 3)

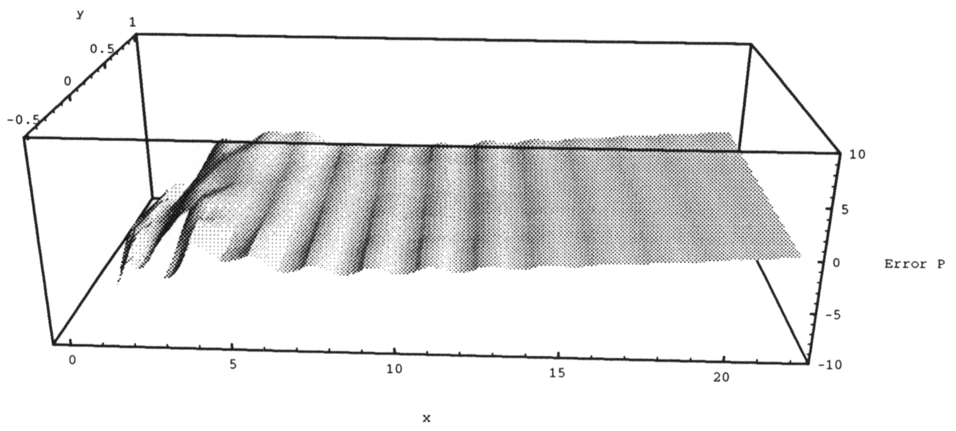

Figure 5.27: A priori errors in P, (step 1)

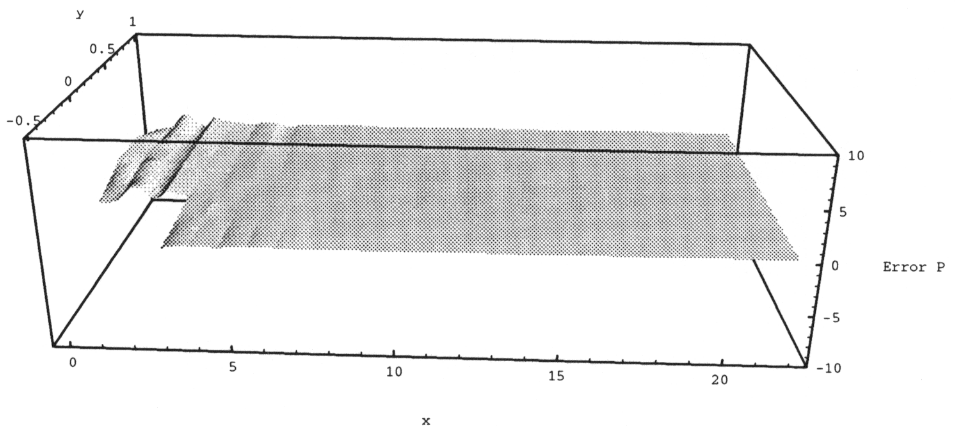

Figure 5.28: A priori errors in P, (step 2)

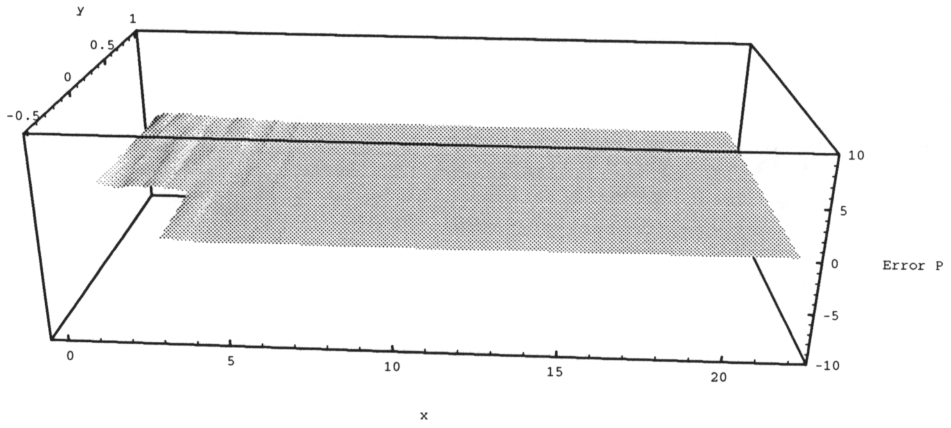

Figure 5.29: A priori errors in P, (step 3)

With these errors a desicion about further refinements can be made, if needed. The a posteriori error analysis is simply the difference between solutions. This error analysis method is a bit more realistic in the sense that the differences are real. The errors are the differences between two converged solutions. These errors are shown in Figures 5.30, 5.32, 5.34, 5.31, 5.33, and 5.35. Notice the decrease in the overall error through the enrichment process.

The velocity vector representations of the first and second solutions are shown in Figure 5.18 and Figure 5.36. In order to add validity to the overall numerical method, the work of Kueny and Binder [64] is used for a comparison in Figures 5.37, 5.38, and 5.39. Notice that as the polynomial order is increased, the solution converges to the experimental solution.

5.3 Three Dimensional Backstep

By extending this method to the third dimension, several limitations become evident. It is the goal of the author to show these limitations in order to spur new research work into these areas. The problem chosen is that of flow over a three–dimensional backstep. The geometry of the configuration is shown in Figure 5.40. The Reynolds number based on the inlet perimeter is 400. Two grids were developed, a fine grid and a coarse grid. The purpose behind the development of two grids is comparative. The actual grids used in the computation are shown in Figures 5.41, and 5.42. The grid in Figure 5.41 contains 621 elements, and is used to calculate the higher order solution. The grid in Figure 5.42 contains 3960 elements, and is used to calculate the

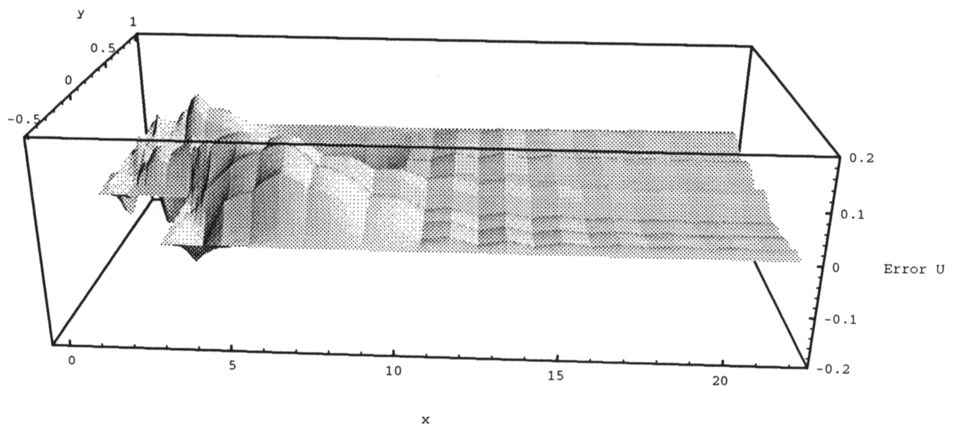

Figure 5.30: A posteriori errors in U (step 1)

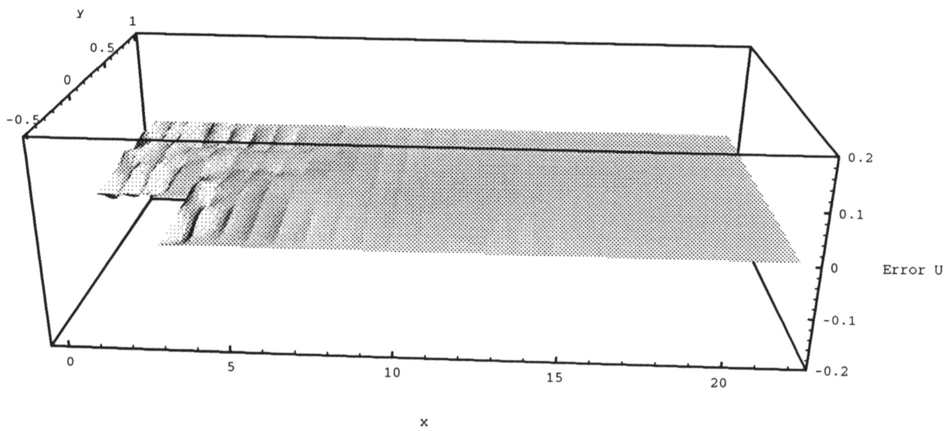

Figure 5.31: A posteriori errors in U (step 2)

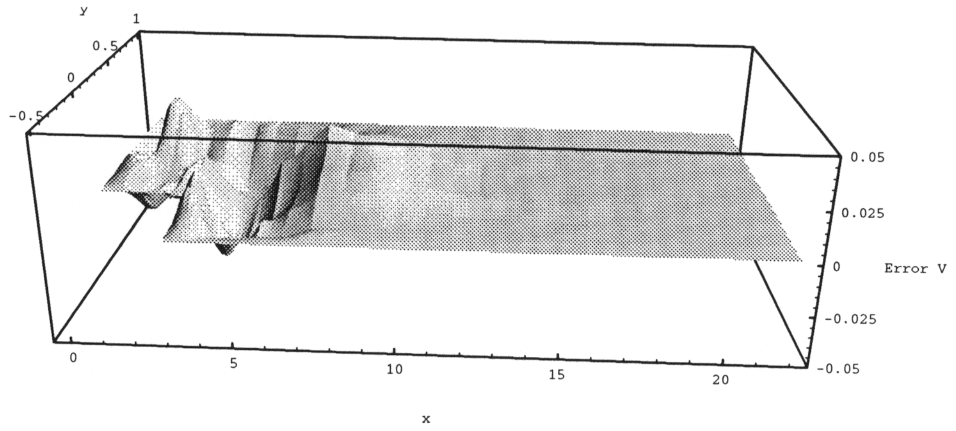

Figure 5.32: A posteriori errors in V (step 1)

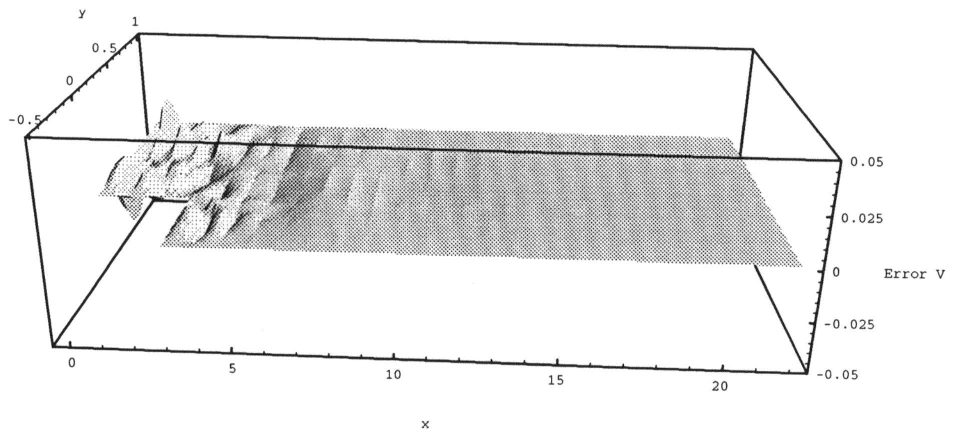

Figure 5.33: A posteriori errors in V (step 2)

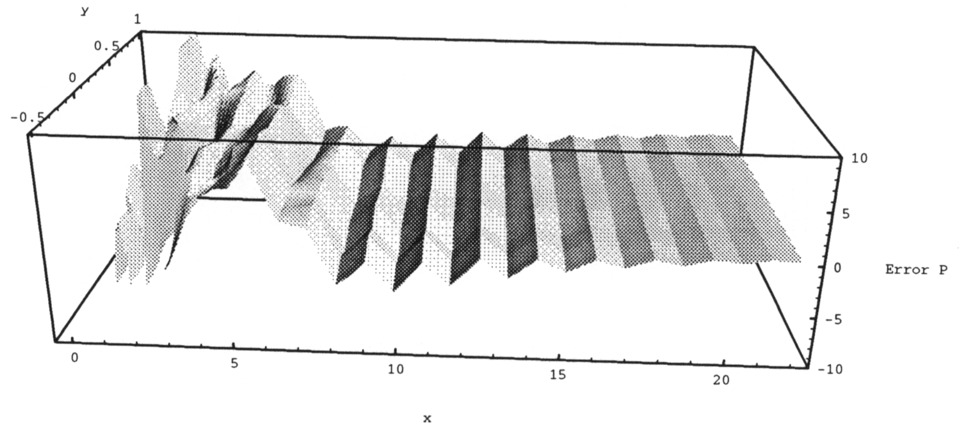

Figure 5.34: A posteriori errors in P (step 1)

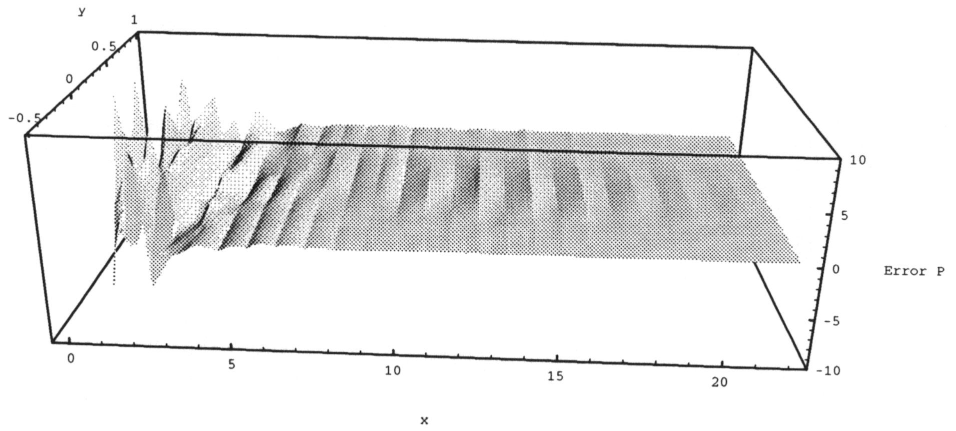

Figure 5.35: A posteriori errors in P (step 2)

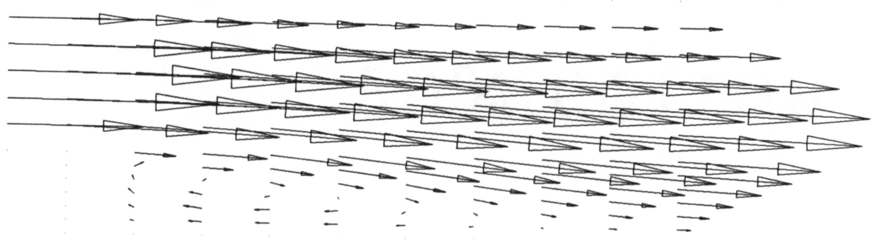

Figure 5.36: Velocities for solution step two

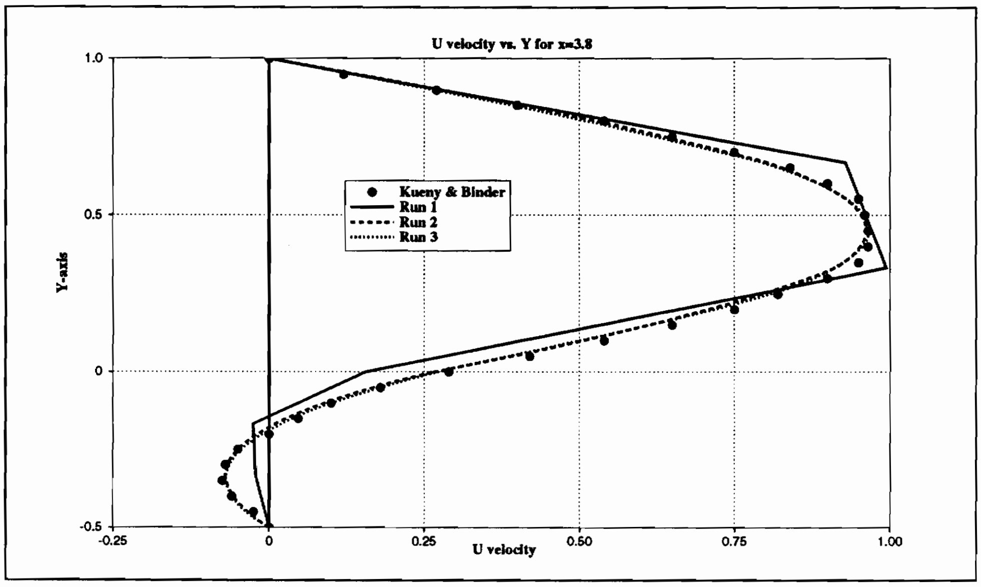

Figure 5.37: Solution at x=3.8

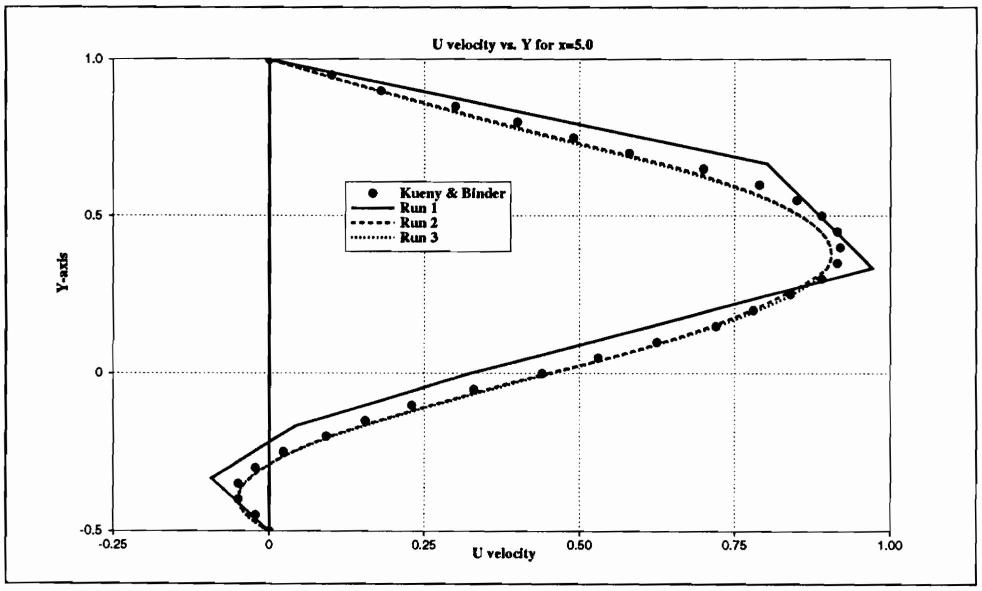

Figure 5.38: Solution at x=5.0

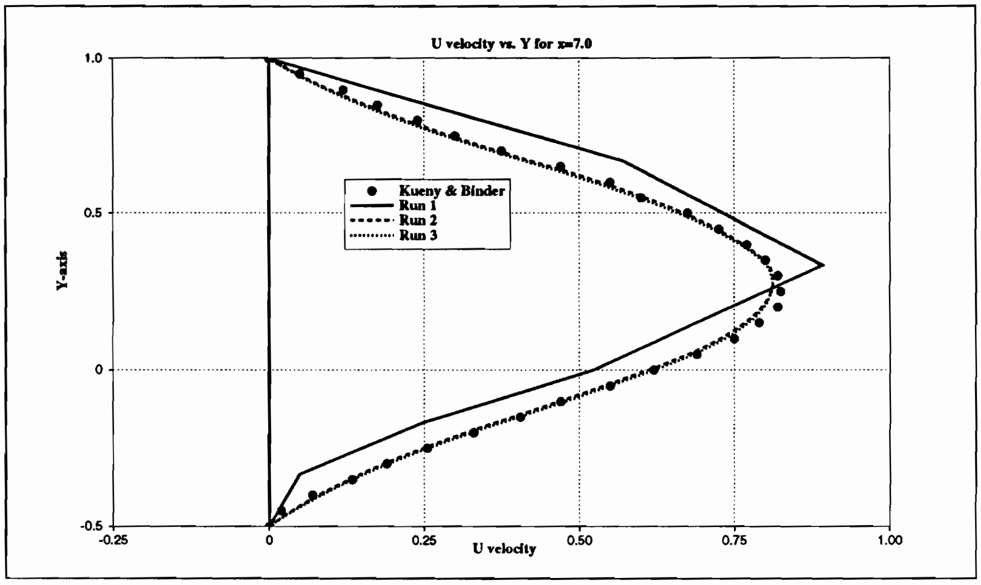

Figure 5.39: Solution at x=7.0

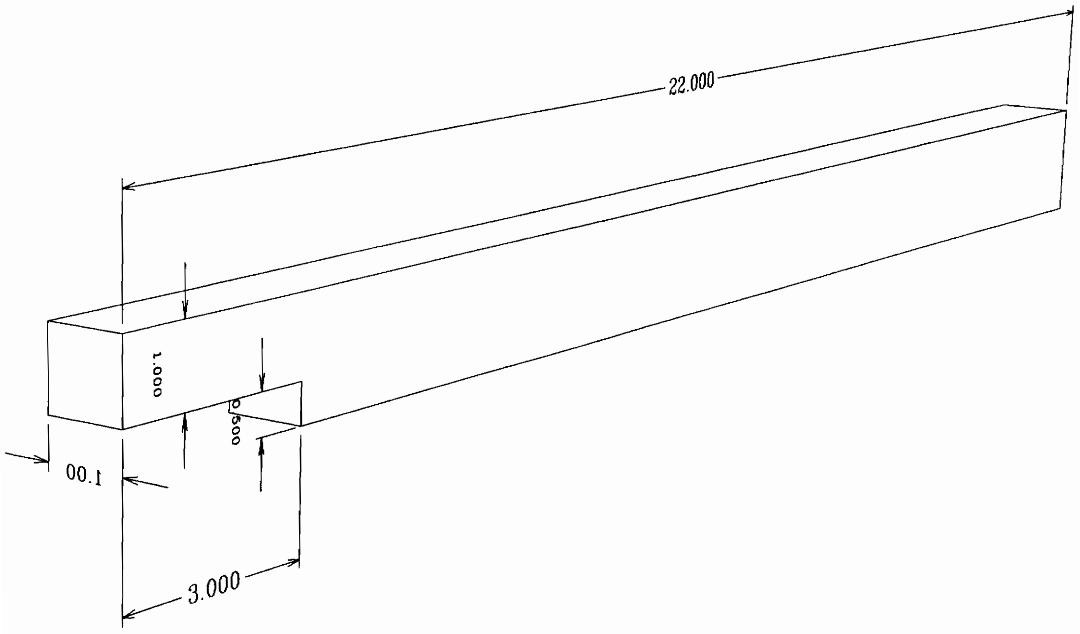

Figure 5.40: Three dimensional backstep geometry

linear solution using 8 noded brick elements. The inlet boundary conditions are calculated assuming fully developed flow. Thus, the inlet profiles are calculated using equation 5.7.

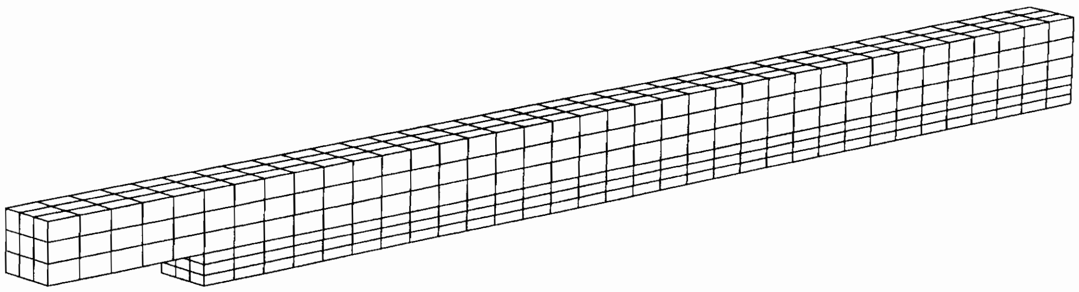
Figure 5.41: Coarse grid for three dimensional backstep

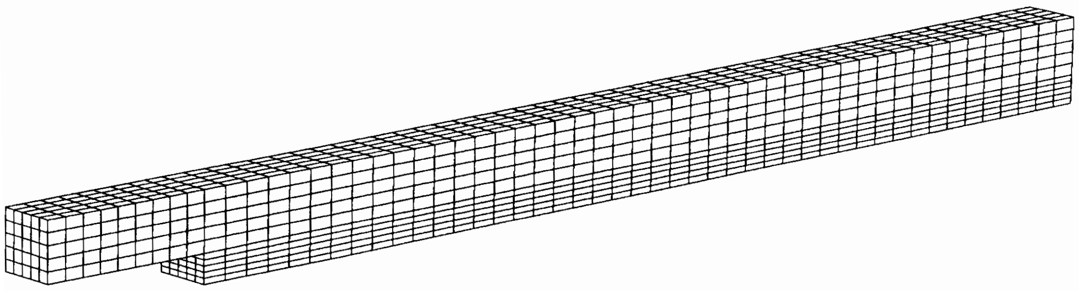
Figure 5.42: Fine grid for three dimensional backstep

$$\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -\frac{1}{\mu}\frac{\partial p}{\partial x} \tag{5.7}$$

For the geometry at hand the solution becomes:

$$u = -\frac{16}{\mu}\frac{\partial p}{\partial x} \sum_{i=1,3,5,\ldots}^{\infty} \sum_{j=1,3,5,\ldots}^{\infty} \frac{a^2}{ij\pi^4(i^2+j^2)} \sin(i\pi)\sin(j\pi)\,. \tag{5.8}$$

With this inlet profile we choose a pressure difference which gives a maximum velocity of 1 (one) at the center of the inlet. With this inlet profile we may calculate the Reynolds number as shown in equation 5.9.

$$\begin{aligned} Re &= \frac{Ul\rho}{\mu} \\ U &= 1.0 \\ l &= 4.0 \end{aligned} \tag{5.9}$$

5.3.1 Solution

Fine Grid

The fine grid solution gives the U velocity contours as shown in Figure 5.43. These contours are produced at various X locations down the length of the backstep. At X=5.0 the flow has returned to a fully developed flow. This phenomenona is attributed to the relatively low Reynolds number of the flow. However, as with most flows over a backstep, there is a recirculation zone

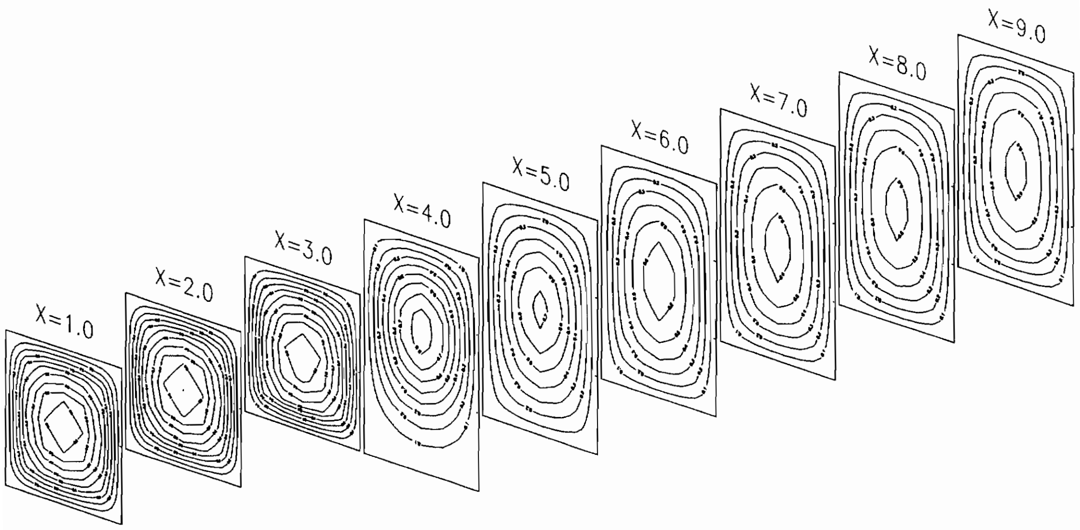

Figure 5.43: U velocity contours, fine grid

just behind the step. In order to show this, U velocity contours are obtained for X = 3.2,3.4,3.6, and 3.8 as shown in Figure 5.44. This solution should be approachable from the coarse grid solution by the use of higher order elements.

Coarse Grid

The adaptive solution begins at a polynomial level equal to 1 throughout the domain. This interpolation is equivalent to the standard eight noded brick elements. The inlet boundary conditions are set such that the velocities at the nodes correspond to values dictated by equation 5.8. The final solution for polynomial level equal to 1 is shown in Figure 5.45. This solution is not at the same level of accuracy as the solution with 3960 elements. However, the general nature of the flow is captured.

The next step in the adaptive process is simply increasing the polynomial order by 1, and repeating the solution process. It is instructive to look at a plot of the a priori error in the u velocity at this point. This error plot is obtained from the difference in the converged linear solution, and a one step quadratic solution. This error plot may be seen in Figure 5.46. Figure 5.46 shows large errors at the center of each element. With these errors as a guide, each element will be extended to include quadratic basis functions. This will give 27 basis functions per element.

At each level the error indicator per element is determined. This error indicator is based on the relation shown in equation 5.10.

$$E_{vel} = \sqrt{\frac{\int_e (\delta u^2 + \delta v^2 + \delta w^2)\, d\Omega}{\int_e (u^2 + v^2 + w^2)\, d\Omega}} \tag{5.10}$$

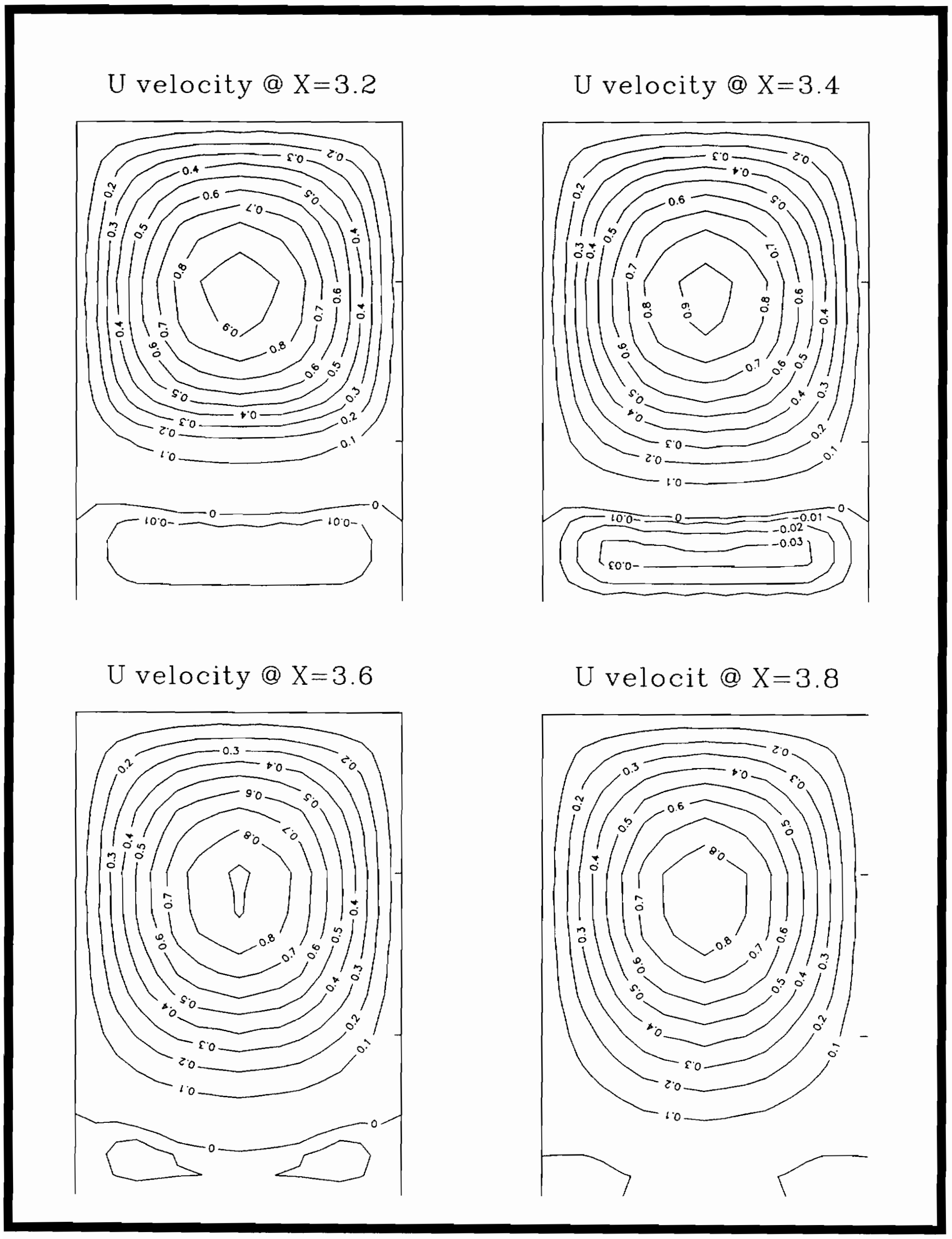

Figure 5.44: U velocity contours behind backstep

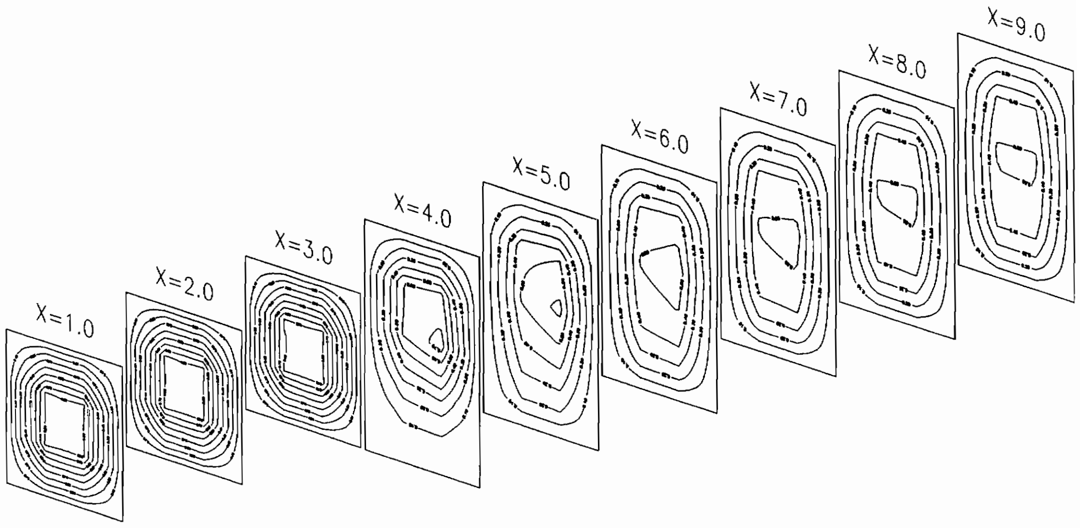

Figure 5.45: U velocity contours for step one

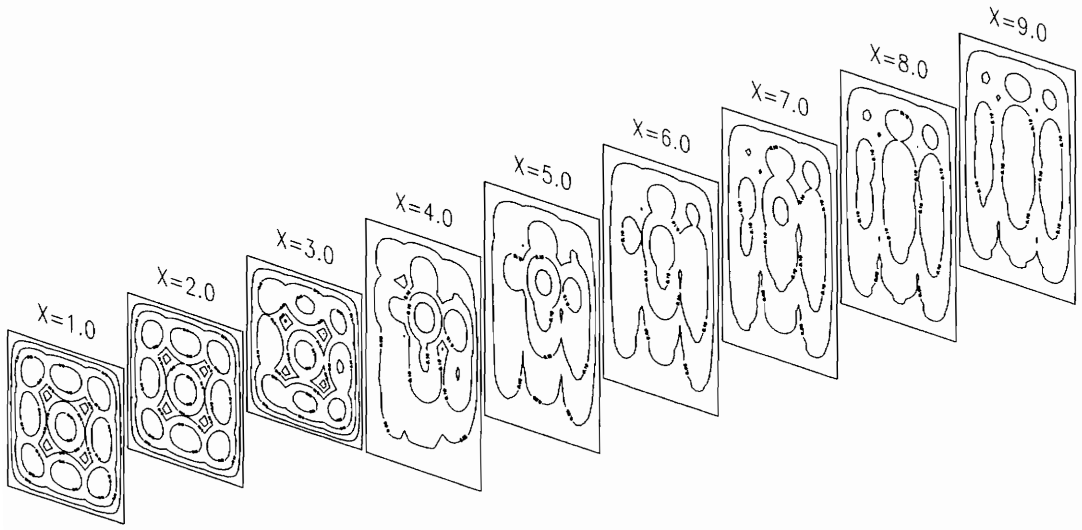

Figure 5.46: A priori error in U velocity

Using this equation an error estimate for each element is determined. This estimate may be considered to be a percentage change for each element. Figure 5.47 shows the geometry for the coarse grid, where elements with a ten percent or higher error are highlighted. This plot inticates that

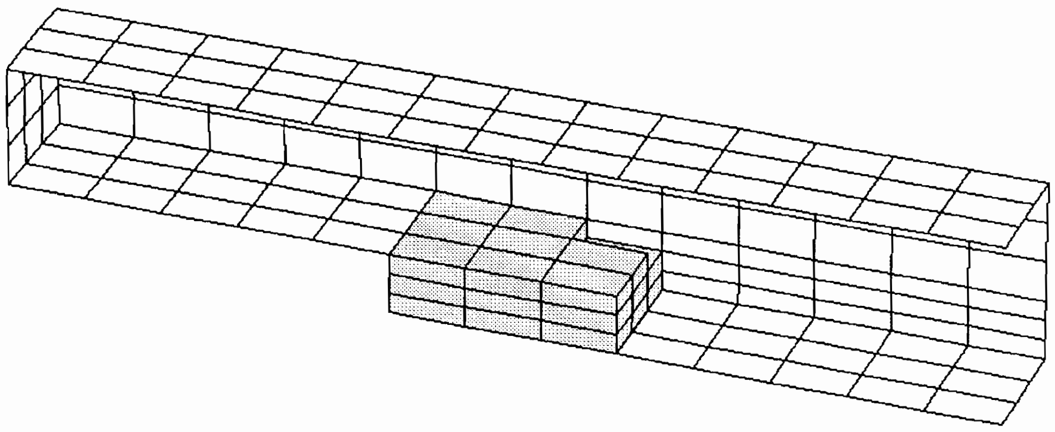

Figure 5.47: High error elements for coarse grid

increasing the polynomial order of the elements just behind the step is in order. However, for the moment, all elements are increased to second order, and converged to the next solution. This gives the opportunity to observe the a posteriori error as compared to the a priori error estimates. Converging the second order solution gives the u velocity contours shown in Figure 5.48. These contours are actually very similar to the contours shown in Figure 5.43. However, at this stage of the convergence process the velocities are not correct behind the backstep.

It is instructive to observe the a posteriori error analysis of the solution between steps one and two. The a posteriori error contour plots of the

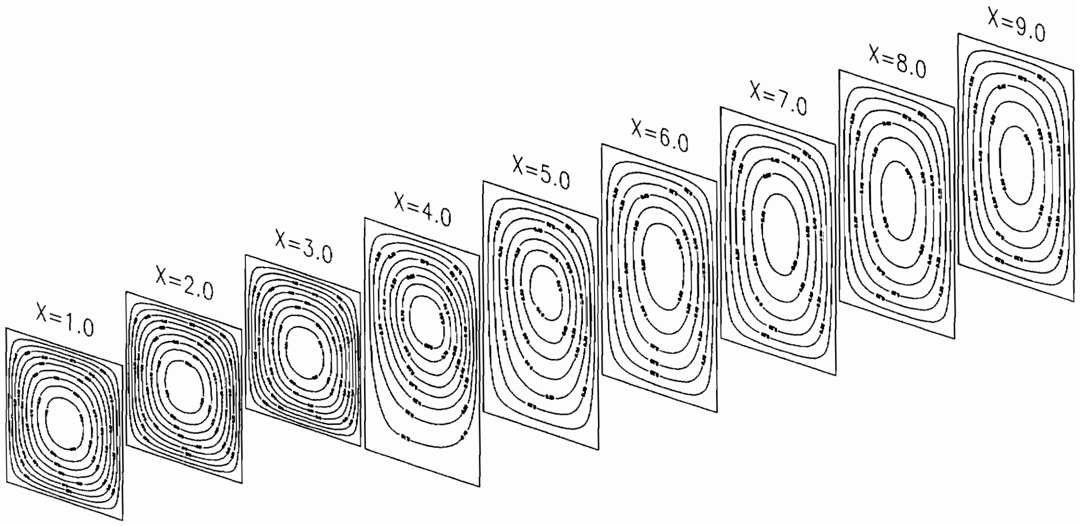

Figure 5.48: U velocity contour for step two

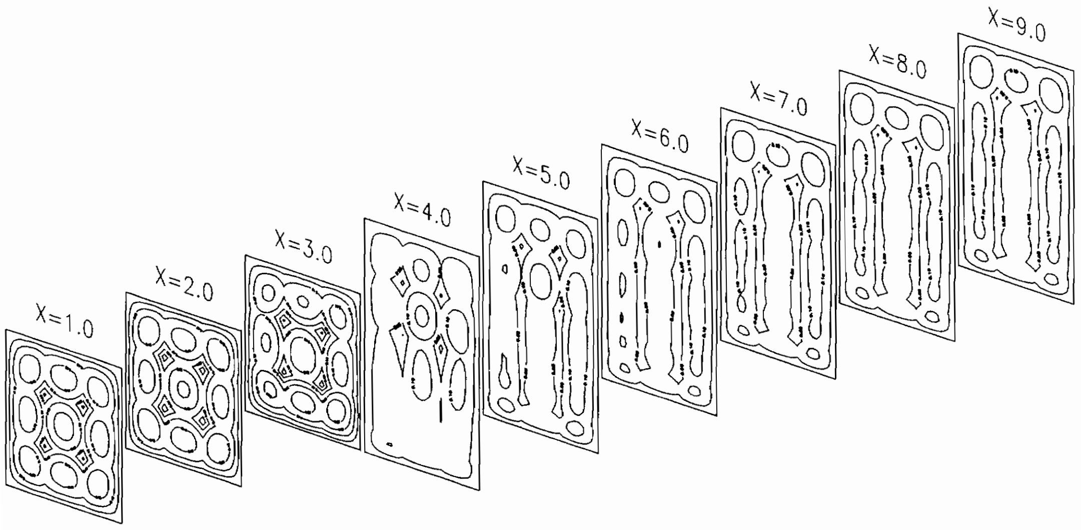

Figure 5.49: A posteriori error in U velocity

u velocity are shown in Figure 5.49. The error per element is shown in Figure 5.50. This figure shows all elements with a twenty percent or greater error in the velocity. Again the error seem to be mostly located just behind the backstep. The error per element is shown in Figure 5.51. This figure

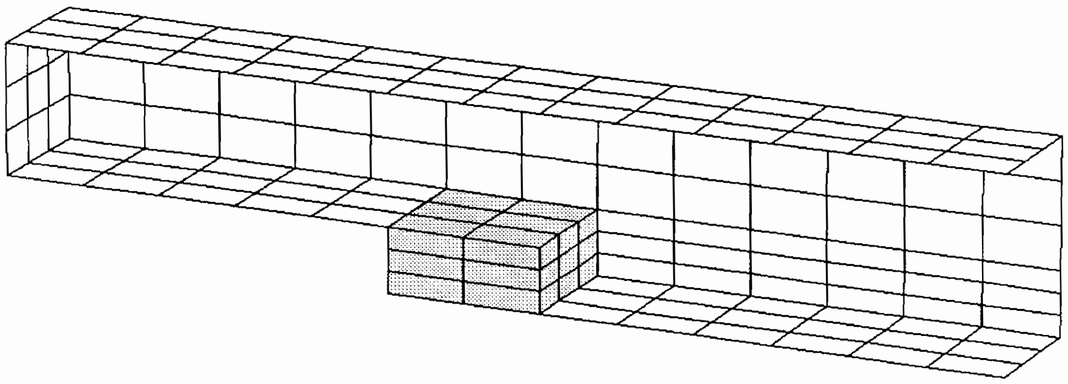

Figure 5.50: Elements with 20% or greater error

shows all elements with a thirteen percent or greater error in the velocity.

The next step in the convergence process is simply to increase the polynomial level to 3 throughout the domain. This is where the actual limitations of this method begins. The details of these limitations will be discussed in the next section.

5.3.2 Polynomial Limitations

Up to this point very little has been said about the integration of the element matrices. However, this is one of the major drawbacks of the p adaptive method. To get a better feel for this limitation, Table 5.3 can be used to observe the number of integration points required for each element integration

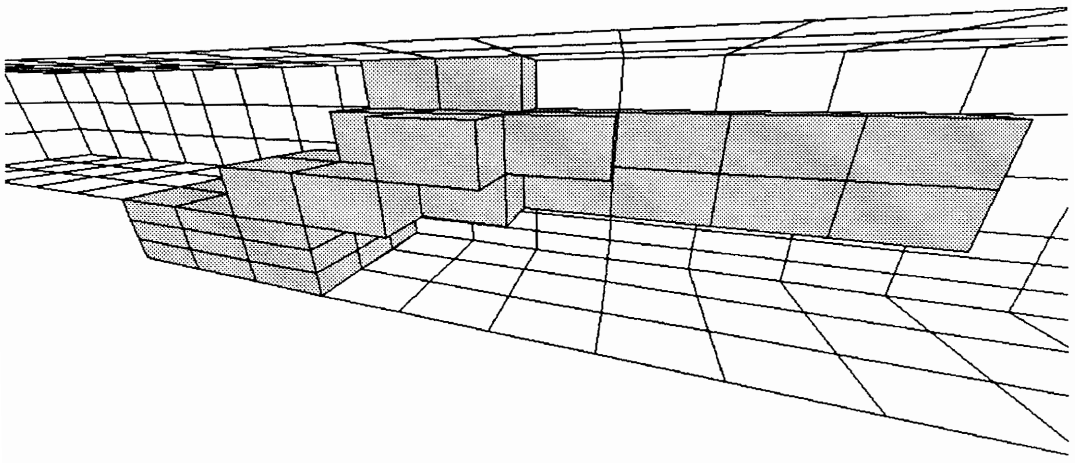

Figure 5.51: Elements with 13% or greater error

Table 5.3: Integration order for the mass matrix

Poly.	Order	Integration Points		
Order	(mass matrix)	(1-D)	(2-D)	(3-D)
1	2	2	4	8
2	4	3	9	27
3	6	4	16	64
4	8	5	25	125
5	10	6	36	216
6	12	7	49	343
7	14	8	64	512
8	16	9	81	729
9	18	10	100	1000

for a typical mass matrix. This is still not the complete story. In Table 5.4 note not only the high number of integration points, but the increasingly high number of basis functions which must be integrated, and the high number of matrix terms. The column labeled 'Function Evaluations' can be explained by observing the actual formula for Gauss integration. To integrate a three-dimensional element we use the relation in equation 5.11.

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(x,y,z)\,dxdydz = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_i w_j w_k f(x_i, y_j, z_k) \qquad (5.11)$$

The number of functional evaluations is simple, the number of times the summation terms are evaluated multiplied by the number of functions to be

Table 5.4: Integration for three dimensional mass matrix

Poly. Order	Order (mass matrix)	Basis functions	Integration points	Function Evaluations
1	2	8	8	64
2	4	27	27	729
3	6	64	64	4096
4	8	125	125	15625
5	10	216	216	46656
6	12	343	343	117649
7	14	512	512	262144
8	16	729	729	531441
9	18	1000	1000	1000000

evaluated.

To obtain a feel for the amount of computer time required, a simple computer simulation was constructed. The runs consisted of one element. The computer times were made nondimensional by the time required to construct the matrices for an eight noded linear brick element. The actual time required to construct this eight noded linear brick element on the NeXT computer was 1.7 seconds. The times are shown in Figure 5.52. One important note should be made, the time given included the time required to read in and construct the connectivity and to integrate linear portions of all of the matrices. However, the overhead time for one element should be very small. From this

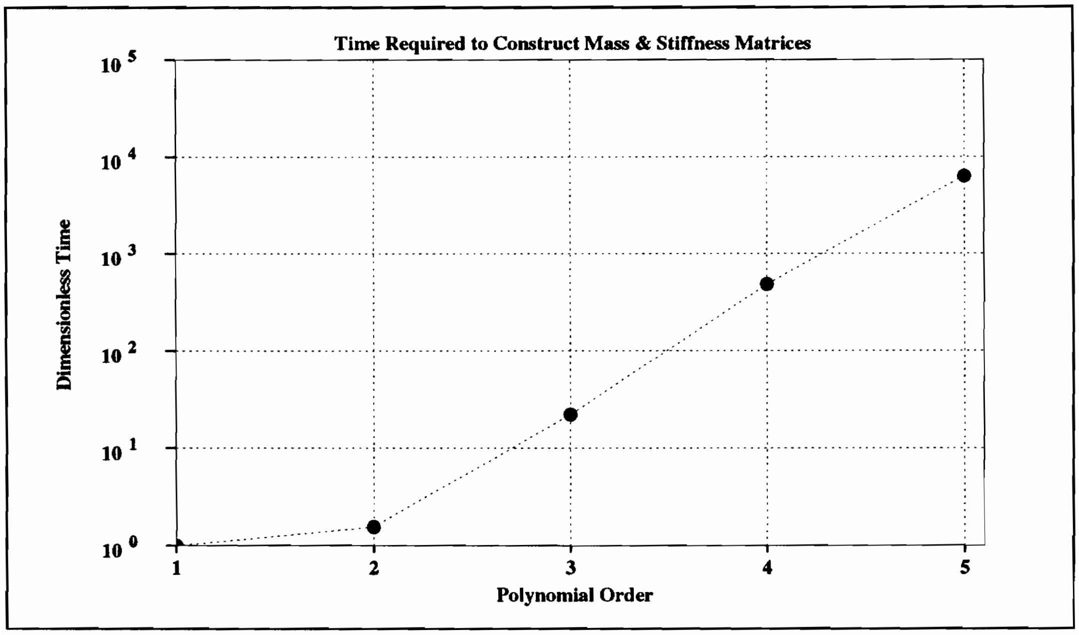

Figure 5.52: Times required at various polynomial orders

plot, the approximate time required to construct the 621 elements can be determined. For a polynomial order of 5 the approximate time required to construct these matrices is 1864 hours or about 77 days. This time does not include the time required to construct the convection portion of the matrix, or the time required to solve the system of equations.

Chapter 6

Conclusion

An element by element iterative solution method has been developed using a segregated solution method for the incompressible Navier-Stokes equations. Using direct solvers, the segregated solution method is compared to the penalty method for incompressible flows. The segregated solution methods compare very favorably with the penalty methods. Next, iterative methods are compared with direct solution methods. For large scale three-dimensional problems, the iterative methods have a significant advantage over direct solution methods. However, for thinly banded two-dimensional problems, the iterative methods lose their advantages as far as computer solution times are concerned. P adaptive element formulations are introduced at this point in the work. These elements are free to reach a polynomial level independent of all adjacent elements. Since the elements are hierarchical in form, they can be conveniently used in error analysis and convergence strategies. Both two and three-dimensional flow over a backward facing step is considered. The limitations of such element formulation were brought out in the three-dimensional flow situations.

6.1 Segregated Formulation

The segregated solution method used is that of Comini and Del Giudice [11, 12]. This segregated solution method gives direct solution times of the same order as that of the times for the penalty formulation. In the context of iterative solvers, segregated solution methods hold an advantage due to their low conditioned matrices. By contrast, the penalty methods matrix condition number is high making iterative solution methods very difficult.

6.2 Matrix Iterative Methods

As compared to direct solution methods, the conjugate gradient type methods hold a significant advantage over the direct methods. The use of such methods have allowed problems which were previously restricted to a supercomputer to become solvable on high speed inexpensive workstations.

These iterative methods are classified as element by element methods. The element by element methods never need assemble the global stiffness matrix. Therefore, a large amount of computer storage is saved. In the present work all symmetric matrices are solved using the conjugate gradient method, and all nonsymmetric matrices are solved by use of the bi-conjugate gradient method.

The number of iterations needed to obtain a converged solution is a function of the condition number of the matrix. With this in mind, the need for preconditioning becomes apparent. In this work the Jacobi or diagonal preconditioning is used. This preconditioning, in effect, places unity on the diagonal and less than unity on the off-diagonal terms. This type of preconditioner is computationally very fast and efficient.

6.3 P Adaptive Methods

The use of p adaptive element formulations is inspired by the need to obtain an accurate solution to a field problem. The hierarchical p formulation is inspired by both the need to obtain an accurate solution to a field problem, and the need to obtain a reasonable estimate of the error. The hierarchical p formulation is developed by a telescoping series of higher order polynomials, while retaining lower order functions.

The hierarchical p formulation is used in both two- and three-dimensions. The formulation is implemented such that each element is allowed to reach its own polynomial level independent of the other elements. The interelement continuity is satisfied by increasing common element sides to the highest order possible. The use of this adaptive methodology is shown to converge to the exact solution in the two-dimensional backstep example. As the solution approaches converegnce, the error indicators decrease with each increase in polynomial order.

In three-dimensions, the problem of obtaining a converged solution is very difficult. As the polynomial order increases, the number of required functional evaluations for integration makes convergence through p adaptive methods restrictive. In the case shown, constructing the mass matrix for 621 elements takes on the order of 1864 hours for a workstation type computer. Even for a supercomputer, which is 100 times faster, the computer time is in the neighborhood of 20 hours. Adding to this the time required to construct the other matrices and solve these matrices for each time step, the required computer time for a complete steady state solution becomes restrictive.

6.4 Recommendations

It is obvious throughout this work that accuracy implies increased cost. This fact is especially true for three-dimensional problems. For two-dimensional problems, the order of the element shape functions does not produce an impenetrable barrier to a solution. However, three-dimensional problems do possess an impenetrable barrier to a solution. In fact, the amount of computer time required to complete a solution with a small number of very high order elements may be much larger than the time required to solve the same problem with a very large number of linear elements. This comment is made based simply on the amount of time required to integrate the higher order shape functions and does not include the increased computational effort required to solve the equations.

As with any new research area, there are many large stumbling blocks which must be removed to advance the state of the art. Several extensions to this work could be made in order to remove some of these stumbling blocks. One very important research area is implementation of faster integration rules. Accomplishing this goal would have a large impact on the computational cost and make higher order approximations possible. Another important concern is the iteration performance for these higher order elements. As the polynomial order increases, so too does the bandwidth of the system of equations. Also, increasing the polynomial order increases the bandwidth of the system matrix, which in turn slows the iterative convergence. The condition of the matrix is also a concern. The ability to chose the higher order element shape functions from a large space of functions gives the freedom to chose the best functions based on the condition number of the system matrices. The solution of the system matrices can also be improved by use of an effective preconditioner. Using an effective preconditioner may decrease the matrix solution time significantly.

Bibliography

[1] I. Babuska. “The Finite Element Method with Penalty”. *Math. Comput.*, 27:221–228, 1973.

[2] I. Babuska and Zlamal M. “Nonconforming Elements In The Finite Element Method with Penalty”. *SIAM J. Numer. Anal.*, 10:863–875, 1973.

[3] J. N. Reddy. “On Penalty Function Methods in The Finite Elements Analysis of Flow Problems”. *Int. J. Numer. Methods Fluids*, 2:151–171, 1982.

[4] J. N. Reddy. “On The Finite Elements Method With Penalty for Incompressible Fluid Flow Problems”. In J. R. Whitman, editor, *The Mathematics of Finite Elements and Applications III*, pages 227–235. Academic Press, London, UK, 1979.

[5] T. J. R. Hughes, W. K. Liu, and A. Brooks. “Finite Element Analysis of Incompressible Viscous Flow by The Penalty Function Formulation”. *J. Comput. Phy.*, 30:1–60, 1979.

[6] R. S. Marshall, J. C. Heinrich, and O. C. Zienkiewicz. "Natural Convection in a Square Enclosure by a Penalty Function Method Using Primitive Variables". *Numer. Heat Trans.*, 1:315–330, 1978.

[7] F. H. Harlow and J. E. Welch. "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface". *Phys. Fluids*, 8:2182–2189, 1965.

[8] A. J. Chorin. "Numerical Solution of the Navier-Stokes Equations". *Math. Comput.*, 22:745–762, 1968.

[9] S. V. Patankar. *"Numerical Heat Transfer and Fluid Flow"*. Hemisphere, New York, 1980.

[10] S. V. Patankar, A. Pollard, A. K. Singhal, and S. P. Vanka. *"Numerical Prediction of Flow, Heat Transfer, Turbulence, and Combustion (Selected Works of Professor D. Brian Spalding)"*. Pergamon Press, New York, 1983.

[11] G. Comini and S. Del Giudice. "Finite-Element Solution of the Incompressible Navier-Stokes Equations". *Numerical Heat Transfer*, 5:463–478, 1982.

[12] G. Comini and S. Del Giudice. "A $(k - \epsilon)$ Model of Turbulent Flow". *Numerical Heat Transfer*, 8:133–147, 1985.

[13] Y. M. Kim and T. J. Chung. "Finite-Element Analysis of Turbulent Diffusion Flames". *AIAA Journal*, 27(3):330–339, March 1989.

[14] J. L. Sohn, Y. M. Kim, and T. J. Chung. "Finite Element Solvers for Incompressible Fluid Flows and Heat Transfer". In T. J. Chung and Gerald R. Karr, editors, *Finite Element Analysis in Fluids*, pages 880–885. University of Alabama Press, Huntsville Alabama, 1989.

[15] J. N. Reddy. *"An Introduction to the Finite Element Method"*. Mc Graw Hill, New York, 1984.

[16] J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewert. *"LINPACK Users Guide"*. SIAM Publications, Philadelphia, 1979.

[17] G. F. Carey and J. T. Oden. *"Finite Elements Computational Aspects"*. Prentice Hall, New Jersey, 1984.

[18] G. H. Golub and C. F. Van Loan. *"Matrix Computations"*. Johns Hopkins University Press, Baltimore, 1989.

[19] J. Dongarra. "Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment". Technical Report 23, Argonne National Laboratory, January 1986.

[20] Absoft Corporation. "F77 version 3.1", 1991. Personal Comunication with Technical Support.

[21] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. "Basic Linear Algebra Subprograms for Fortran Usage". *ACM Trans. Math. Software*, 5(3):308–371, 1979.

[22] K. J. Bathe. *"Finite Elements Procedures in Engineering Analysis"*. Prentice Hall, New Jersey, 1982.

[23] B. Irons. "A Frontal Program for Finite Element Analysis". *International Journal for Numerical Methods in Engineering*, 2:5–32, 1970.

[24] P. Hood. "Frontal Solution Program for Unsymmetric Matrcies". *International Journal for Numerical Methods in Engineering*, 10:379–399, 1976.

[25] C. Taylor and T. G. Hughes. *"Finite Element Programming of the Navier–Stokes Equations"*. Pineridge Press, Swansea, UK, 1981.

[26] K. J. Berry. "An Efficient C-Based Wavefront Solver for PC Finite Element Applications". *Computers and Structures*, 39:303–315, 1991.

[27] Rosen. "Matrix Bandwidth Minimization". In *ACM 23rd National Conf.*, pages 585–595. Brandon Systems Press, 1968.

[28] E. Cuthill and J. McKee. "Reducing the Bandwidth of Sparse Symmetric Matrices". In *ACM National Conf.*, pages 157–172, New York, 1969.

[29] E. Cuthill. "Several Strategies for Reducing the Bandwidth of Matrcies". In D. J. Rose and R. A. Willoughby, editors, *Sparse Matrcies and Their Applications*, pages 157–166. Plenum Press, 1972.

[30] A. George. "Computer implementation of the Finite Element Method". Technical report, Computer Science Dept., Stanford University, 1971.

[31] M. Hoit and E. L. Wilson. "An Equation Numbering Algorithm Based on a Minimum Front Criteria". *Computers and Structures*, 16(1-4):225–239, 1983.

[32] S. W. Sloan and W. S. Ng. "A Direct Comparison of Three Algorithms for Reducing Profile and Wavefront". *Computers and Structures*, 33(2):411–419, 1989.

[33] S. W. Sloan. "A Fortran Program for Profile and Wavefront Reduction". *International Journal for Numerical Methods in Engineering*, 28:2651–2679, 1989.

[34] D. A. Pierre. *"Optimization Theory With Applications"*. Dover, New York, 1986.

[35] C. M. Bazaraa, M. S.and Shetty. *"Nonlinear Programming Theory and Algorithms"*. John Wiley, New York, 1979.

[36] A. Cauchy. "Méthode Générale pour la Résolution des Systémes d' Équations Simultanées". *Comptes Rendus Hebdomadaires des Sénances de l' Académie des Sciences*, 25:536–538, October 1847.

[37] A. Jennings. *"Matrix Computations for Engineers and Scientist"*. John Wiley, New York, 1977.

[38] M. Hestenes and E. Stiefel. "Methods of Conjugate Gradients for Solving Linear Systems". Technical Report Report 1659, National Bureau of Standards, 1952.

[39] W. C. Davidson. "Variable Metric Method for Minimization". Technical Report ANL-5990, AEC Research Development Report, 1959.

[40] R. Fletcher and C. Reeves. "Function Minimization by Conjugate Gradients". *Computer Journal*, 7:149–154, 1964.

[41] I. Fried. "A Gradient Computational Procedure for the Solution of Large Problems Arising from the Finite Element Discretization Method". *International Journal for Numerical Methods in Engineering*, 2:477–494, 1970.

[42] A. Jennings and G. M. Malik. "The Solution of Sparse Linear Equations by the Conjugate Gradient Method". *International Journal for Numerical Methods in Engineering*, 12:141–158, 1978.

[43] K. V. G. Prakhya. "Some Conjugate Gradient Methods for Symmetric and Nonsymmetric Systems". *Communications in Applied Numerical Methods*, 4:551–539, 1988.

[44] O. Axelsson. "Conjugate Gradient Type Methods for Unsymmetric and Inconsistent Systems of Linear Equations". *Linear Algebra and its Applications*, 29:1–16, 1980.

[45] K. C. Jea and D. M. Young. "On the Simplification of Generalized Conjugate-Gradient Methods for Nonsymmertrizable Linear Systems". *Linear Algebra and its Applications*, 52 / 53:399–417, 1983.

[46] L. J. Hayes. "Advances and Trends in Element-by-Element Techniques". In A. K. Noor and J. T. Oden, editors, *State of the Art Surveys on Computational Mechanics*, pages 219–236. AMSE, New York, 1989.

[47] L. Meirovitch. *"Computational Methods in Structural Dynamics"*. Sijthoff & Noordhoff, Rockville, Maryland, 1981.

[48] O. C. Zienkiewicz, B. M. Irons, F. C. Scott, and J. Cambell. "Three Dimensional Stress Analysis". In *IUTAM Symp. on High Speed Computing of Elastic Structures.* University of Liege Press, 1971.

[49] A. G. Peano. "Hierarchies of Conforming Finite Elements For Plane Elasticity and Plate Bending". *Comput. Meths. with Appl.*, 2, 1976.

[50] B. A. Szabo and A. V. Mehta. "P-convergent Finite Element Approximations in Fracture Mechanics". *Int. J. Num. Meth. Engng*, 12:551–560, 1978.

[51] I Babuska, B. A. Szabo, and I. N. Katz. "The P-version of the Finite Element Method". *SIAM J. Numer. Anal.*, 18:514–545, 1981.

[52] I. Babuska. "The P- and hp-versions of the Finite Element Method: The State of The Art". In Dwoyer, Hussaini, and Voigt, editors, *Finite Elements: Theory and Applications.* Springer-Verlag, New York, 1988.

[53] A. T. Patera. "A Spectral Element Method for Fluid Dynamics: Laminar Flow in a Channel Expansion". *Journal of Computational Physics*, 54:468–488, 1984.

[54] K.Z. Korczak and A. T. Patera. "An Isoparametric Spectral Element Method for Solution of the Navier-Stokes Equations in Complex Geometry". *Journal of Computational Physics*, 62:361–382, 1986.

[55] L. J. Hayes. "Spectral Element Methods for the Incompressible Navier-Stokes Equations". In Y. Maday and A. T. Patera, editors, *State of the*

Art Surveys on Computational Mechanics, pages 71–143. AMSE, New York, 1989.

[56] D. W. Kelly, De S. R. Gago, O. C. Zienkiewicz, and I. Babuska. "A Posteriori Error Analysis and Adaptive Processes in the Finite Element Method: Part 1-Error Analysis". *International Journal for Numerical Methods in Engineering*, 19:1593–1619, 1983.

[57] D. W. Kelly, De S. R. Gago, O. C. Zienkiewicz, and I. Babuska. "A Posteriori Error Analysis and Adaptive Processes in the Finite Element Method: Part 2-Adaptive Mesh Refinement". *International Journal for Numerical Methods in Engineering*, 19:1621–1256, 1983.

[58] O. C. Zienkiewicz, De S. R. Gago, and D. W. Kelly. "The Hierarchical Concept in Finite Element Analysis". In A. K. Noor and J. M. Housner, editors, *Advances and Trends in Structural and Solid Mechanics*, pages 53–65. Pergamon Press, New York, 1983.

[59] I. Babuska, M. Griebal, and J. Pitkaranta. "The Problem of Selection Shape Functions for a p-Type Finite Element". *International Journal for Numerical Methods in Engineering*, 28:1891–1908, 1989.

[60] O. R. Burggraf. "Analytical and Numerical Studies of the Structure of Steady Sererated Flows". *Journal of Fluid Mechanics*, 24:113–151, 1966.

[61] K. N. Ghia and U. Ghia. "ELLIPTIC SYSTEMS: Finite-Difference Mehtod III". In W. J. Minkowycz, E. M. Sparrow, G. E. Schneider, and

R. H. Pletcher, editors, *Handbook of Numerical Heat Transfer.* John Wiley, New York, 1988.

[62] B. F. Armaly, F. Durst, J. C. F. Pereira, and B. Schonung. "Experimental and Theoretical Investigation of Backward-Facing Step Flow". *Journal of Fluid Mechanics*, 127:473–496, 1983.

[63] K. Morgan, J. Periaux, and F. Thomasset, editors. *"Analysis of Laminar Flow Over a Backward Facing Step".* Friedr. Vieweg & Sohn, 1984.

[64] J. L. Kueny and G. Binder. "Viscous Flow over Backward Facing Steps– An Experimantal Investigation". In K. Morgan, J. Periaux, and F. Thomasset, editors, *Analysis of Laminar Flow Over a Backward Facing Step*, pages 32–47. Friedr. Vieweg & Sohn, 1984.

[65] G. Yagawa and Y. Eguchi. "Comparison Between the Traction and Pressure–Imposed Boundary Conditions in Finite Element Flow Analysis". *International Journal for Numerical Methods in Fluids*, 7:521–532, 1987.

[66] G. F. Carey, K. C. Wang, and W. D. Joubert. "Performance of Iterative Methods for Newtonian and Generalized Newtonian Flows". *International Journal for Numerical Methods in Fluids*, 9:127–150, 1989.

[67] M. D. Gunzburger. "Iterated Penalty Methods for the Stokes and Navier-Stokes Equations". In T. J. Chung and Gerald R. Karr, editors, *Finite Element Analysis in Fluids*, pages 1040–1045. University of Alabama Press, Huntsville Alabama, 1989.

[68] M. P. Reddy. *"Finite Element Simulation of Three-Dimensional Casting, Extrusion and Forming Processes"*. PhD thesis, Virginia Polytechnic Institute and State University, Dec. 1990.

[69] R. Fletcher. *"Conjugate Gradient Methods for Indefinite Systems"*. Springer, New York, 1976. Lecture Notes in Mathimatics 506.

[70] J. Devloo, J. T. Oden, and P Pattani. "An h–p Adaptive Finite Element Method for the Numerical Simulation of Compressible Flow". *Computer Methods in Applied Mechanics and Engineering*, 70:203–235, 1988.

[71] C. Canuto, M. Y. Hussaini, Quarteroni, and T. A. Zang. *"Spectral Methods in Fluid Dynamics"*. Springer, New York, 1987.

[72] L. Demokowicz and J. T. Oden. "A Review of Local Mesh Refinment Techniques and Corresponding Data Structures in h–Type Adaptive Finite Element Methods". Technical Report TICOM Report 88–02, Texas Institute for Computational Mechanics, 1988.

Vita

Robert M. Fithen was born on August 14, 1959 in Minden, Louisiana. He graduated from High School in 1978. He attended college Louisiana Tech University, where he graduated with honors in 1984 (B.S.M.E.). He attended Texas A&M He worked for three years in the Computational Fluid Dynamics group at the Fort Worth division of General Dynamics. He came to Virginia Polytechnic Institute and State University in 1989 to obtain a PhD in Engineering Mechanics.

R. M. Fithen