1995

Unstructured grid algorithms for two- and three-dimensional flows on parallel computers

Thomas Hans Ramin
Iowa State University

Unstructured grid algorithms for two- and three-dimensional flows on parallel computers

by

Thomas Hans Ramin

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Mechanical Engineering
Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1995

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Roman Symbols

$\vec{b}$	right hand side vector of matrix equation
E	internal energy per unit mass
f	flux in x-coordiante direction
g	flux in y-coordiante direction
h	flux in z-coordiante direction
F	flux
c	speed of sound
i	cell index
k	thermal conductivity
q	conserved variables
p	pressure
t	time
T	temperature
u	velocity component in the x direction
v	velocity component in the v direction
w	velocity component in the w direction
$\vec{x}$	vector on unknowns in the matrix equation

A,B,C	Jacobian matrices
M	Mach number
Pr	Prandtl number
Re	Reynolds number
S_j	area of cell face j
V_i	volume of control volume i

Greek Symbols

γ	ratio of specific heats
μ	dynamic viscosity
ρ	density
$\vec{\eta}$	unit normal vector of the control volume surface
ηx, ηy,ηz	x,y,z components of $\vec{\eta}$

ACKNOWLEDGEMENTS

I thank the Internal Fluid Mechanics Division at NASA Lewis Research Center for awarding me a Graduate Researcher's Fellowship (Grant NGT-50949). I also thank NASA Lewis Research Center for the use of their computing facilities.

I am also grateful to the Scalable Computing Laboratory at Ames Laboratory for the use of the Ncube computer.

1. INTRODUCTION

The main application for unstructured grids probably lies in flows associated with complex geometries. Structured grid generation over complex geometries is very difficult in two and three dimensions. The complexity encountered with structured grids is caused by the inherent regularity of the grid. In a structured grid, each point (vertex) is connected to an equal number of neighbors by gridlines. Thus, the grid can be mapped into a regular Cartesian grid with a fixed number of gridlines in each coordinate direction. An advantage with structured grids is the ease with which the data structure can be utilized. Each vertex can be addressed by its location relative to the indexes of its neighbors. It is this inherent efficiency of a structured grid which makes it difficult to obtain reasonable grids over complex geometries. Researchers using structured grids have resorted to using domain decomposition to partition the domain into separate regions. A grid must then be obtained in each region separate from the others. This multi-domain grid generation approach leads to several problems. Overlapping grids with interpolation at the interface can be used to connect the different domains. The flow solver becomes significantly more complex.

In a domain discretized by unstructured grids, each vertex is allowed to have a different number of neighboring vertices. The problems associated with structured

grids mentioned above do not exist with unstructured grids. In contrast to structured grids, addressing neighboring vertices or control volumes is not possible by using the indexes of the grid. The data structure is more complex and must be generated explicitly. The data structure must be provided to the computer program and requires extra storage.

Any polygon or combination of polygons can be used to discretize a domain in an unstructured manner. Triangles in two dimensions and tetrahedra in three dimensions are the simplest and perhaps most convenient geometries that are available to discretize a domain. Triangles and tetrahedra can be used to generate grids about arbitrary geometries. In this study, only triangles (2-D) and tetrahedra are used to discretize the domain. In the current research, a control volume method is used to solve the conservation equations in two and three dimensions. A cell centered approach is taken which means that the triangles (2-D) or tetrahedra (3-D) of the grid themselves are used as control volumes. These points will be discussed later in this chapter and in the following chapters.

Unstructured grids are a relatively new facet of computational fluid dynamics and research needs to be done to find ways of implementing the approach as well as to determine limitations. It is already clear that the computer time and storage requirements of unstructured grids can become easily excessive, and could conceivably offset the advantages of the unstructured grid technique [1] [2] [3] [4].

The current research is done to develop an unstructured grid technique in two and three dimensions which can be applied to two- and three-dimensional flows. Attention is given to the adaptivity of different numerical schemes to low Mach number flows. The low Mach number capability is introduced by the use of time-derivative

preconditioning. The use of time-derivative preconditioning has an advantage over simply using an incompressible numerical scheme in that one and the same approach can be used for a much wider variety of flows. The implementation of the approach to massively parallel computers and workstation clusters is researched in this project. At the current stage of computing technology, the parallel approach to computing seems destined to replace current high performance computers in the near future. The continued use and application of any computational technique is entirely dependent on its adaptability to modern computing machinery.

This chapter describes the finite volume approach and its relation to the finite element method. Different approaches to implement the conservation laws are discussed along with issues associated with the application of numerical techniques on parallel computers. An overview over the test cases analyzed is also given.

1.1 Forms of the Conservation Law

A rigid control volume approach is normally used when solving the fluid dynamics conservation equations on unstructured meshes. The fundamental underlying ideas of the control volume approach are demonstrated below. The general relation to finite-element approaches is also mentioned for completeness.

The general form of the governing equations is:

$$\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} = 0$$

or, written in divergence form

$$\frac{\partial q}{\partial t} + \vec{\nabla} \cdot \vec{F} = 0$$

where $\vec{F} = f\vec{i} + g\vec{j} + h\vec{k}$.

The so called weak form of the conservation law is obtained by multiplying the equation in divergence form by a weighting function W. Integration by parts then gives the following expression:

$$\frac{\partial}{\partial t}\int_{CV} W q dV - \int_{CV} \vec{\nabla} W \cdot \vec{F} dV + \int_{CS} W \vec{F} \cdot \vec{\eta} dS = 0 \qquad (1.1)$$

The term $\vec{\eta}$ stands for the unit vector perpendicular to the control volume surface S ($\vec{\eta} = \eta_x\vec{i} + \eta_y\vec{j} + \eta_z\vec{k}$). The weak form of the governing equations serves as the basis for many numerical solution methods. It is in order to mention that the weighting functions can be quite general, leading to different classes of numerical methods. For example, in the Galerkin finite element method, the weighting functions at a gridpoint are the shape functions. The shape function for a linear element or control volume has a value of 1.0 at the location of the unknown and a value of zero at all the surrounding locations of unknowns [5]. This makes the last term in Equation (1.1) zero. This is only true for a location of an unknown in the interior of the domain. The weighting function, and thus the surface integral in Equation (1.1) is generally not zero over the boundary of the computational domain [5] [6]. In the interior of the domain, the integral of the product of the gradient of the weighting function and the fluxes remains.

In the control volume approach, a different weighting function is used, leading to different expressions. In the control volume method, the weighting function is 1.0 inside the control volume and zero outside. Equation (1.1) reduces to Equation (1.2)

which is the basis for the control volume method.

$$\frac{\partial}{\partial t}\int_{CV} q dV + \int_{CS} \vec{F}\cdot\vec{\eta} dS = 0 \tag{1.2}$$

Under certain conditions, the Galerkin finite element method and the control volume method can be shown to be equivalent for linear elements except for the time derivative term [4]. A linear element assumes a linear variation of the unknown within each control volume. The argument made by Barth [4] is based on the fact that for piecewise linear functions, the geometric terms in the volume integral of Equation (1.1) can be expressed entirely by the control volume surface. The resulting expression is consistent with the integration path representing the median dual (see Figure 1.1). The control volume lies inside the dual and the median dual is an example of a possible dual (which is also the control volume surface). The median dual surrounding vertex A connects the midpoint of the edges of the mesh to the geometric center of the faces of the mesh (in this case triangles).

Two approaches to computing physical phenomena on unstructured grids are generally used, vertex based schemes and cell centered schemes. In a vertex based scheme, the unknowns are stored at the vertices of the grid. The control volume surface passes through all neighboring triangles or tetrahedra. The number of edges in a vertex based scheme can be rather large and is generally different for each vertex. As is illustrated in Figure 1.1, a vertex based scheme would store the unknowns at vertex A and the control volume would lie inside the median dual. In a cell centered scheme, the unknowns would be stored at the geometric center of the triangle of the mesh and the corresponding control volume would be the edges of the mesh (the triangles in Figure 1.1). Since the control volume is the triangle itself and the

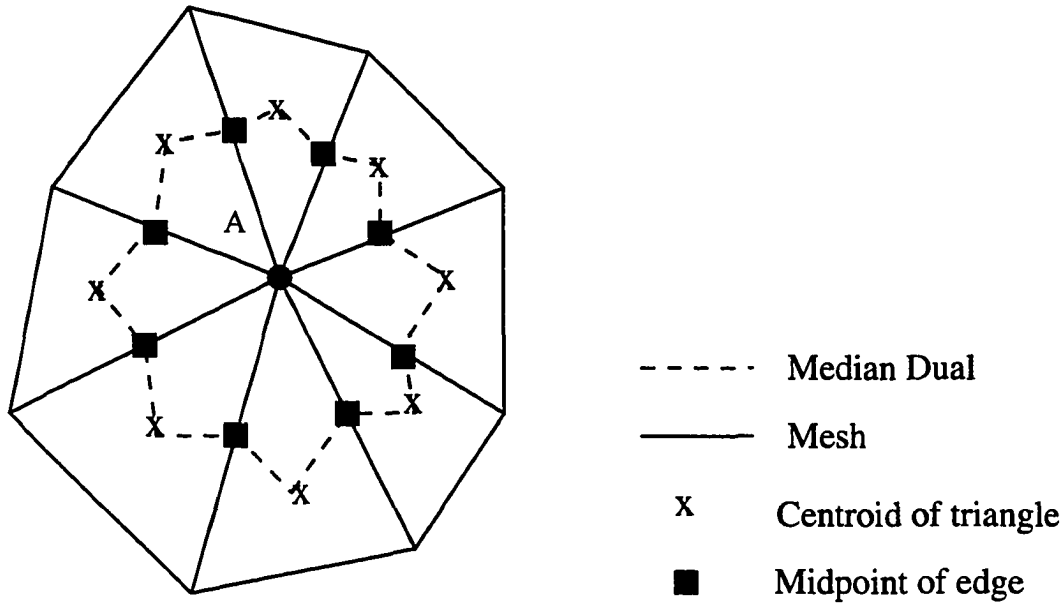

Figure 1.1: Illustration of mesh and dual

fluxes must be computed over the edges of the triangle. This approach can offer advantages in terms of efficiency and simplicity. On the other hand, difficulties as to the gradient reconstruction which result from the issues in finding an appropriate integration path have to be mastered. The calculation of the gradient in the control volume is necessary if an upwind scheme is to be of second order accuracy. In a second order scheme, the numerical solution method accounts for a linear variation of the unknowns in the control volume. In a first order scheme, the unknowns are taken as constants in the control volume. These issues are further discussed in the following chapter.

Unstructured grid techniques for the Navier-Stokes equations generally require more memory than is needed with conventional structured grids. With unstructured

grids, as with finite element methods, the grid connectivity needs to be stored. It is the intensive use of memory which is currently a hurdle for the increased popularity of unstructured grids. Using unstructured grids, researchers have the ability to model applied engineering problems in one piece. However, computational power limitations have largely prevented such calculations. Very large problems have been solved on current vector supercomputers, but more computer resources are needed. Intensive work on unstructured grid methods is proceeding in the expectation of simultaneous progress in massively parallel computing as a new and promising approach to high performance computing. In massively parallel computing the algorithm is divided among several, sometimes a large number of processors and executed simultaneously.

There are two approaches to parallel computing called synchronous and asynchronous computing. The first approach is to build synchronous or lockstep coordination into the hardware by forcing all operations to be performed at the same time in a manner that removes the dependency of one task on another. The second form, called asynchronous because there is no lockstep coordination mechanism, relies on coordination mechanisms called locks to coordinate processes.

One form of synchronous parallelism is called single instruction, multiple data (SIMD). In a SIMD computer, all processors do the same thing at the same time or else they remain idle. One instruction counter is used to sequence through a single copy of the program. The data that are processed by each processing element differ from processor-to-processor. Each processor in the array has a small amount of local memory where the distributed data reside while being processed in parallel. The processor array is connected to the memory bus of the front-end so the front-end can randomly access the local processor memories as if it were another memory. The

front-end is usually a workstation set up to provide user interface to the parallel machine.

Multiple instruction multiple data (MIMD) is the most general form of parallelism. In this form the processing units perform instructions independently on different data. Synchronization is achieved explicitly and locally rather than through a global synchronization mechanism. Computers of this type usually have distributed-memory architectures where the program and the data reside. Distributed-memory designs offer higher levels of parallelism through the interconnection of a large number of processors, but they require a message passing strategy. The design of a distributed-memory parallel processor places great demands on communication speed, routing, and data partitioning.

Since in both types of parallel computers each processor is allocated a part of the computational domain, a number of cells or any other part of the computational work required, a new aspect of modeling the physics of a problem on such a computer becomes the communication among processors. Naturally, a computational overhead is associated with the communication process which can become excessive for small grain sizes. The term grain size refers to how the workload of an algorithm is distributed among the processors. If the grain size is large, potentially concurrent tasks are executed sequentially by one processor. A small grain size implies more overhead.

The need for communication between processors often dictates how a problem is mapped onto a parallel computer topology. No set standard currently exists for the computer topology and there will most likely never be such a standard. It is advantageous for the user of a massively parallel computer to have a large number of communication paths available going from and leading to each processor. However,

with an increasing number of processors, such a network quickly becomes infeasible. Also, it is desirable that the computer topology be scalable to a large number of processors. Some parallel computers use a nearest neighbor network. A popular topology is the hypercube where the number of communication paths for a single processor is equal to the dimensions of the hypercube. The hypercube topology is a compromise between the desire to have a large number of processor communication paths and scalability.

1.2 Scope of the Current Research

The current research focuses on the development of a Navier-Stokes solver using unstructured grids in two and three dimensions and the implementation on parallel computers. Triangular and tetrahedral unstructured grids were used throughout in this study. In general, there is no requirement that only triangular or tetrahedral grids be used. Any polygon or combination of polygons could serve as a discretization of the domain. A cell-centered approach was used. In a cell-centered approach, the dependent variables are stored at the geometric centers of the polygons. In two dimensions, time-derivative preconditioning was applied to a central-difference scheme using conserved as well as primitive variables and to a flux-splitting type scheme using primitive variables only.

To date, not much work has been done toward applying cell-centered unstructured grids to three-dimensional viscous flows. The implementation of unstructured Navier-Stokes solvers on parallel computers is a new research area. Also, the use of time-derivative preconditioning using conserved variables and its application to flux-splitting schemes is relatively new.

Both two- and three-dimensional studies have been made. Upwind and central-difference schemes were investigated in the developmental stage of the research. It was found that the Roe upwind flux difference splitting was more robust than the central-difference scheme. This was particularly true for three-dimensional flow. The approach used in this study is based primarily on conserved variables, which, like approaches using primitive variables, exhibits poor convergence at low Mach number. Time-derivative preconditioning is employed for an explicit scheme in two dimensions. Since low Mach number preconditioning is based on rescaling the eigenvalues of the flux Jacobians, a new set of left and right eigenvectors results. For an upwind scheme, the appropriate left and right eigenvectors would have to be computed to apply preconditioning to the system of equations.

The test cases analyzed are tabulated in Table 1.1. The two-dimensional test cases were computed using a central-difference Runge-Kutta scheme. The first test case was that of a developing channel flow. The flow was chosen to demonstrate the correctness of the approach. The symmetric expansion case was analyzed to further demonstrate the ability and gain experience with the unstructured grid technique. A driven cavity flow with heat transfer was computed to demonstrate the ability of the technique to resolve wall parameters. The flow over an obstruction in a channel was used to demonstrate the benefits of the use of time-derivative preconditioning at low Mach numbers. All two-dimensional flows, except for the driven cavity flow were also analyzed on the Ncube, a massively parallel MIMD computer with 256 processors.

Three three-dimensional test cases were analyzed using an implicit upwind scheme. The first test case was a developing channel flow. As in two dimensions, the case was used to verify the correctness of the procedure. A three-dimensional driven cavity

was used to to further verify the algorithm. A curved channel case was used to apply the algorithm to a case more closely related to turbomachinery applications. All three-dimensional flows were calculated on a CRAY-YMP. The developing channel and the curved channel flow were also analyzed on parallel computers. The developing channel flow and the curved channel flow were analyzed on the LACE cluster, a workstation cluster consisting of 32 IBM RS-6000 workstations. The developing channel flow case was analyzed on the Ncube as well.

Table 1.1: Summary of test cases analyzed

2-dimensional flows	3-dimensional flows
developing channel	developing channel
driven cavity (isothermal)	curved channel
driven cavity (heat transfer)	driven cavity
symmetric expansion	
flow over an obstruction	

2. WORK BY OTHER RESEARCHERS

Unstructured grids have been given a lot of attention in the the field of computational fluid dynamics for quite some time. The increased popularity of unstructured grids increases the insatiable hunger for computational power in the computational fluid dynamics community. In fact, the popularity of unstructured grids is a result of the feasibility of obtaining significantly higher computational power due to massively parallel computer technology in the relatively near future.

A large number of researchers have solved the two-dimensional Euler equations using unstructured grids, and an increasing number of studies focusing on the two-dimensional Navier-Stokes equations is found in the literature. However, relatively few people have performed unstructured grid studies focusing on the three-dimensional Navier-Stokes equations. Some of these studies have been performed on massively parallel computers. This new approach to performing the calculations was brought about by the necessity to achieve a higher calculational speed, and the need for more memory to make unstructured grids a viable method for the future. At the present time, traditional vector supercomputers are not expected to be able to provide analysts with significantly higher computational speeds in the future. Advances in computational fluid dynamics are dependent on the further development of increasing computational power. Thus, research in the implementation of computational fluid

dynamics applications on massively parallel computers is of vital importance.

Researchers have used both the cell-centered and the vertex based approach in their unstructured grid studies. For example, Mavripils [1], Frink [3] and Batina [37] have used the cell-centered approach. The vertex based approach has been used by many including Mavriplis [1], Barth and Frederickson [8] and Holmes and Connel [9]. No particular approach has yet to be proven to be superior to another. Barth and Jespersen [10] point out that the real difference in the two approaches becomes apparent when the boundary conditions are implemented. The phantom cell boundary treatment seems to be used mostly by users of the cell-centered approach, whereas no clear preference for vertex based schemes has been identified as of yet. The dual is clearly identifiable in vertex based schemes, but the complexity of the implementation of the method is greatly increased. Often, vertex values are being computed or approximated in the cell-centered schemes. The use of vertex values is more convenient in the gradient computation since the cell itself can be used as the domain of interest. Approximate reconstruction in conjunction with cell-centered unstructured meshes are used by Frink [3], Knight [11] and Pan and Cheng [12]

It is because of the high memory requirements associated with implicit schemes that a large number of studies have used explicit schemes. Some of the researchers such as Anderson and Bonhaus [13], Pan and Cheng [12] and Smith and Spragle [14] successfully employed a turbulence model in their studies. For turbulent studies, explicit as well as implicit schemes have been used. For implicit turbulent schemes [12] [13], the author only knows of two-dimensional studies. A parameter that usually needs to be computed in most turbulence models is the distance from the wall. This computation is not straightforward in unstructured grids.

Ramamurti and Löhner [15] point out that the grid in the viscous region should adhere to the physics of the flow to be of respectable efficiency. It is pointed out by Ramamurti and Löhner [15] and Holmes and Connel [9] that a structured grid is actually more likely to be able to accommodate such flows. This is pointed out by several authors and structured grids of triangular [15] or rectangular shape [9] have been employed to resolve the viscous region.

Upwind as well as central-difference type methods have been used extensively when solving the Euler or Navier-Stokes equations on unstructured grids. Batina [7] and Whitaker [16] solved the three-dimensional Euler equations on unstructured grids implicitly using Roe's flux difference splitting. Both used an approximate Jacobian on the left hand side. The scheme was second order accurate upon convergence. The two-dimensional Euler equations were solved implicitly.

Frink [3] solved the three-dimensional Euler equations explicitly using Roe's upwind flux difference splitting. The three-dimensional turbulent Navier-Stokes equations were solved by Smith and Spragle [14] using an explicit central difference scheme with artificial dissipation. Anderson and Bonhaus [13], Knight [11] and Pan and Cheng [12] solved the two-dimensional Navier-Stokes equations implicitly using Roe's flux difference splitting. Holmes [9] solved the two-dimensional Navier-Stokes equations using a central difference scheme. The two-dimensional Euler equations were solved using a central difference scheme by Winterstein and Hafez [17] and by Barth and Jespersen [4] using a Roe's flux difference splitting.

Memory requirements for unstructured grid algorithms are high due to the nature of the data structure associated with them. Also, due to the inherent unstructuredness of the resulting coefficient matrix, many efficient matrix solution methods cannot

be applied. Since the further development and future popularity of the unstructured grid method is dependent on increased computing power, many researchers including Whitaker [16], and Ramamurti Löhner [15] used current vector supercomputers with specifically large memory capabilities to solve relatively realistic problems. At the present time, limitations in the physically achievable speedup of traditional supercomputers seem to have been reached. Massively parallel computers can offer much higher computational speed than traditional vector supercomputers. Both kinds of parallel computers, MIMD and SIMD have been used by researchers to solve the unstructured flow equations. The term MIMD stands for multiple instruction multiple data and SIMD stands for single instruction multiple data.

A detailed analysis and discussion of implementing a structured Navier-Stokes algorithm on an Ncube2e (a MIMD computer) is given by Kominsky [18]. The bulk of the study deals with implementing efficient solution methods designed specifically for structured grids on the hypercube topology of the Ncube computer. The author observed that different mappings of the grid onto the processors are appropriate for different parts of the overall algorithm. For example, Bruno and Capello [19] designed a technique to map the grid onto processors such that any plane cut perpendicular to a coordinate axis will intersect exactly one subdomain assigned to each node. Kominsky [18] was able to allocate a $5 \times 5 \times 5$ grid onto a processor of the Ncube, with 2 phantom gridpoints in each coordinate direction. The implementation by Kominsky [18] required message passing in stages to properly compute the artificial dissipation terms the numerical solution required. Message passing in stages was prefered over increasing the number of phantom cells. Kominsky [18] used the algorithm of Bruno and Capello [19] to efficiently apply the Thomas algorithm. Another technique men-

tioned by Kominsky [18] is to use a different mapping of the grid onto the processors at different stages of the overall algorithm. For example, different mappings would be applied for different directions in an ADI algorithm. Such an approach would be prohibitive in terms of computational overhead [18].

Das et al. [20] made qualitative studies solving the three-dimensional Euler equations using a MIMD machine. A two-dimensional implicit Navier-stokes solver was implemented on a MIMD machine by Hixon and Sankar [21]. Their work focused on the parallelization of the GMRES matrix solvers and not on the coarse grained parallelism associated with the domain decomposition procedure for solving a problem on a MIMD computer. Venkatakrishnan et al. [22] used a MIMD computer to solve the two-dimensional Euler equations. The authors used a vertex based unstructured grid scheme. The analysis by Venkatakrishnan et. al [22] describes several domain decomposition techniques. The goal of the investigators was to minimize the communication requirement among the processors while maintaining load balancing. The amount of communication necessary for a vertex based scheme is proportional to the number of vertices located on interprocessor boundaries. Approaches for the domain decomposition by the authors were geometric domain decomposition and sorting algorithms by Cuthill and McKee [23] and a spectral partitioning algorithm by Pothen et al. [24]. The authors of the study also investigated stripwise and domainwise partitioning of the grid. The effect of domainwise partitioning required more but shorter messages whereas stripwise decomposition was accompanied by fewer, but longer messages. The overall result of the study was that domainwise decomposition with spectral sorting resulted in the best performance.

A SIMD computer was used by Hammond and Barth [2] to solve the two-

dimensional Euler equations. Cui and Knight [25] used a SIMD machine to solve the two-dimensional Navier-Stokes equations. The techniques of implementing an algorithm on a MIMD and a SIMD computer differ greatly. As opposed to a MIMD computer, processors in a SIMD computer have very little memory and all processors in a SIMD computer perform the same instruction. The common approach of allocating the domain on the processors is to map the entire grid onto the processors so that each processor has only one control volume stored. If there are more control volumes or gridpoints than processors, several gridpoints are allocated to the same processor. As for a MIMD computer, communication between processors is more efficient with the nearest processor. In a SIMD computer, each allocated gridpoint or control volume generally must communicate with the neighboring gridpoint which is located on a different processor.

3. METHOD OF SOLUTION

This chapter discusses the conservation equations used for fluid flow and the approach taken to solve them. Some of the issues involved when solving the Navier-Stokes equations with the control volume method are also discussed. The two- and three-dimensional Navier-Stokes equations are solved in conservation law form. A central-difference scheme, the AUSM (Advection Upstream Splitting Method) and and Roe's flux difference splitting method were used to discretize the equations. The implicit versions were solved using a block Gauss-Seidel procedure. Preconditioning was applied for a two-dimensional explicit scheme using primitive as well as conserved variables.

3.1 Governing Equations

As was discussed in the introduction, control volume methods are based on a form of the governing equations where the expression for the time rate of change in the control volume is integrated over the control volume and the fluxes are integrated over the control volume surface. The control volume method is demonstrated for two-dimensional problems.

The conservation equations for the control volume method have the following form:

$$\frac{\partial}{\partial t}\int_{CV} q dV + \int_{CS} \vec{F}\cdot\vec{\eta} dS = 0 \tag{3.1}$$

where $\vec{F} = f\vec{i} + g\vec{j} + h\vec{k}$. The vector q is the vector of unknowns in terms of conservative variables:

$$q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}$$

where u, v, w are the velocity components, and E is the total internal energy per unit mass:

$$E = c_v T + \frac{u^2 + v^2 + w^2}{2}$$

which for an ideal gas is:

$$E = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{u^2 + v^2 + w^2}{2}$$

The total fluxes consist of a inviscid and viscous part:

$$f = f_i - f_v$$

$$g = g_i - g_v$$

$$h = h_i - h_v$$

where the viscous and inviscid flux vectors are given below.

$$f_i = \begin{pmatrix} \rho u \\ \frac{1}{2}\rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho Hu \end{pmatrix} \quad g_i = \begin{pmatrix} \rho v \\ \rho vu \\ \frac{1}{2}\rho v^2 + p \\ \rho vw \\ \rho Hv \end{pmatrix} \quad h_i = \begin{pmatrix} \rho w \\ \rho wu \\ \rho vw \\ \frac{1}{2}\rho w^2 + p \\ \rho Hw \end{pmatrix}$$

where $H = E + \frac{p}{\rho}$

$$f_v = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{xx}u + \tau_{xy}v + \tau_{xz}w + k\frac{\partial T}{\partial x} \end{pmatrix}$$

$$g_v = \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ \tau_{yx}u + \tau_{yy}v + \tau_{yz}w + k\frac{\partial T}{\partial y} \end{pmatrix}$$

$$h_v = \begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ \tau_{zx}u + \tau_{zy}v + \tau_{zz}w + k\frac{\partial T}{\partial z} \end{pmatrix}$$

The τ_{xx}, τ_{xy}, τ_{xz}, τ_{yy},τ_{yz}, and τ_{zz} are the shear stresses which are defined as:

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu(\vec{\nabla}\cdot\vec{v})$$

$$\tau_{xy} = \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})$$

$$\tau_{xz} = \mu(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x})$$

$$\tau_{yy} = 2\mu\frac{\partial v}{\partial y} - \frac{2}{3}\mu(\vec{\nabla}\cdot\vec{v})$$

$$\tau_{yz} = \mu(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial x})$$

$$\tau_{zz} = 2\mu\frac{\partial w}{\partial z} - \frac{2}{3}\mu(\vec{\nabla}\cdot\vec{v})$$

3.1.1 Nondimensionalization

It is a common practice to use nondimensionalized forms of the governing equations as the basis for numerical approximations. The main advantage to this approach is that if the proper reference quantities are used, all variables are scaled to have values within a certain range.

The following nondimensionalization was used in the calculations:

$$
\begin{aligned}
t^* &= \frac{t}{L_{ref}/u_{ref}} \qquad x^* = \frac{x}{L_{ref}} \qquad y = \frac{y}{y_{ref}} \\
z^* &= \frac{z}{L_{ref}} \qquad u^* = \frac{u}{u_{ref}} \qquad v^* = \frac{v}{u_{ref}} \\
w^* &= \frac{w}{u_{ref}} \qquad \rho^* = \frac{\rho}{\rho_{ref}} \qquad T^* = \frac{T}{T_{ref}} \\
\mu^* &= \frac{\mu}{\mu_{ref}} \qquad R^* = \frac{1}{\gamma M^2{}_\infty} \quad C_p{}^* = \frac{1}{(\gamma-1)M^2{}_\infty} \\
M_\infty &= \frac{u_{ref}}{\sqrt{\gamma R T_{ref}}}
\end{aligned}
\tag{3.2}
$$

From now on, the asterisks will be dropped for convenience as all variables are in their nondimensional form. The definition of each of these variables can be found in the nomenclature. The subscript ref refers to a reference quantity. For the above set of governing equations used in this study, the nondimensional reference quantities that appear in the equations are the Reynolds (Re) and Prandtl (Pr) number:

$$Re = \frac{\rho_{ref} L_{ref} u_{ref}}{\mu_{ref}} \quad Pr = \frac{C_p \mu_{ref}}{k_{ref}}$$

The thermal conductivity k is computed from the Prandtl number which is taken as constant in this study:

$$k = \frac{C_p \mu}{Pr}$$

The nondimensional numbers appear only the the viscous terms. The inviscid terms remain unchanged in this form. The nondimensional viscous terms are:

$$f_v = \begin{pmatrix} 0 \\ \frac{\tau_{xx}}{Re} \\ \frac{\tau_{xy}}{Re} \\ \frac{\tau_{xz}}{Re} \\ \frac{1}{Re}(\tau_{xx}u + \tau_{xy}v + \tau_{xz}w) + \frac{C_p\mu}{RePr}\frac{\partial T}{\partial x}) \end{pmatrix}$$

$$g_v = \begin{pmatrix} 0 \\ \frac{\tau_{yx}}{Re} \\ \frac{\tau_{yy}}{Re} \\ \frac{\tau_{yz}}{Re} \\ \frac{1}{Re}(\tau_{yx}u + \tau_{yy}v + \tau_{yz}w) + \frac{C_p\mu}{RePr}\frac{\partial T}{\partial y} \end{pmatrix}$$

$$h_v = \begin{pmatrix} 0 \\ \frac{\tau_{zx}}{Re} \\ \frac{\tau_{zy}}{Re} \\ \frac{\tau_{zz}}{Re} \\ \frac{1}{Re}(\tau_{zx}u + \tau_{zy}v + \tau_{zz}w) + \frac{C_p\mu}{RePr}\frac{\partial T}{\partial z} \end{pmatrix}$$

The inviscid fluxes in the surface integral in Equation (3.1), can be written as:

$$\vec{F} \cdot \vec{\eta} = (\vec{V} \cdot \vec{\eta}) \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho H \end{pmatrix} + \begin{pmatrix} 0 \\ p\eta_x \\ p\eta_y \\ p\eta_z \\ 0 \end{pmatrix}$$

where $\vec{V} = u\vec{i} + v\vec{j} + w\vec{k}$ and $\vec{\eta} = \eta_x\vec{i} + \eta_y\vec{j} + \eta_z\vec{k}$ which is the normal to the control volume surface, pointing out of the control volume.

3.2 Control Volume Method

In the control volume method the governing equations are integrated over the control volume. The following expression results:

$$\int_{CV} \frac{\partial q}{\partial t} dV + \int_{CV} \vec{\nabla} \cdot \vec{F} dV = 0$$

where V is the volume (3D) or area (2D) of the control volume. The key step in applying the control volume method is to apply the divergence theorem to obtain:

$$\int_{CV} \frac{\partial q}{\partial t} dV + \int_{CS} \vec{F} \cdot \vec{\eta} dS = 0$$

where $\vec{\eta} = \eta_x\vec{i} + \eta_y\vec{j} + \vec{\eta}_z\vec{k}$ is the unit vector normal to the control volume surface pointing out of the control volume. The application of the Gauss divergence theorem makes the potentially tedious integration of the derivative of the flux vector over the control volume unnecessary.

For an upwind type scheme, the upwinded flux values on both sides of the surface need to be computed. The flux values on each side of the control volume are based on flux or variable values associated with the corresponding control volumes. For a central-difference type scheme, the value at the control surface is computed based on the variable or flux values associated with the cell at each side of the surface. To maintain stability in central-difference schemes, dissipation is added which gives an effect similar to that of an upwind scheme. These points are discussed further later

in this chapter.

In two dimensions the integral over the control volume becomes :

$$\frac{\partial}{\partial t}\int_{CV} q dV + \int_{CS}(f dy - g dx) = 0 \tag{3.3}$$

Note that the integral over the control volume surface is taken counterclockwise. This integral can also be written as:

$$\frac{\partial}{\partial t}\int_{CV} q dV + \int_{CS}(f\eta_x + g\eta_y) dS = 0 \tag{3.4}$$

This form of the conservation law is easier to implement numerically in three dimensions than the form in Equation (3.3). The author is not aware of the existence of an equivalence for counterclockwise in three dimensions. In three dimensions, the form of the integral becomes:

$$\int_{CV} \frac{\partial q}{\partial t} dV + \int_{CS}(f\eta_x + g\eta_y + h\eta_z) dS = 0 \tag{3.5}$$

3.3 Numerical Solution of the Governing Equations

As was explained earlier, a numerical scheme can be cell-centered or vertex based. When the governing equations are solved numerically using a discretized domain, the values of the dependent variables are stored at the vertices in a vertex based scheme, and the variables are stored at the geometric centers of the polygons of the mesh in a cell-centered scheme. There seems to be a tendency in the unstructured grid research community towards nodal schemes. Barth [4] argues that the integration path for a nodal scheme is more isotropic with respect to wave orientation. In other words, in

a nodal scheme, the surface integral involves a larger number of sides, resulting in a contribution of fluxes from more directions than in a cell-centered scheme. Since the flux calculation involves a locally one-dimensional approximation of the waves of which the flux is composed, less isotropicity can be viewed as constituting a higher dependency of the solution on the grid [35]. If the cell-centered approach and the node based approach are compared, it must be noted that the cell-centered approach represents a finer grid. This reasoning is based on the fact that in a triangular mesh the number of triangles is always larger than the number of vertices. In two dimensions, the grid for a cell-centered scheme is approximately twice as fine. This estimate is based on twice as many mesh triangles as vertices in a triangular mesh. The finer grid lessens the effect of the grid dependency.

Roe [36] points out the larger integration error for the integration around a triangle compared to a polygon with more than three sides. The integration error associated with integration around a triangle is of high enough order as to not be worrisome in a second order scheme.

The cell-centered scheme is potentially computationally more efficient if it is taken into account that it represents a finer grid. The gradient calculations have to be performed for each control volume. The potential savings are a result of the median dual being one to one. This means that the same number of faces is in the median duals of a particular mesh as there are triangle faces. The median dual is used as the control volume surface in vertex based schemes. Thus, the same number of control volume faces have to be dealt with in the flux calculation, regardless of the approach used. In the current research a cell-centered scheme is used and no further consideration is given to the concept of a dual.

Previous investigators reported successful attempts at using cell-centered schemes [37] [3] [11]. If a high enough degree of accuracy can be obtained using a cell-centered scheme, then cell-centered schemes offer the advantage of a potentially greater computational efficiency. The higher computational efficiency results from the equal number of sides associated with cells of the same kind and the cell-centered scheme representing a calculation on a finer grid. Previous research studies have not shown that results obtained using a cell-centered approach are inferior to results from nodal schemes. For the current research, a cell-centered approach was used with the intention to point out advantages and further capabilities of the cell-centered approach.

A difficulty with the use of a cell-centered scheme is the lack of a conveniently defined control volume surface for each grid. The control volume surfaces in a cell-centered scheme are formed from the edges of the mesh. For a triangular mesh, a control volume is a triangle itself and the control volume is formed by edges of that triangle. As is shown in Equations (3.4) and (3.5), the fluxes in the conservation equations need to be integrated over the control volume surface. There is no simple geometric relationship between cell centers neighboring the face and the geometric center of the face itself which is applicable in even degenerate grids. These points will be addressed further below.

The calculation of the surface fluxes for a second or higher order cell-centered scheme can be done in numerous ways as long as the gradients meet certain requirements. Thus far, no particular way of performing the flux calculations has been shown to be superior to another in terms of accuracy or efficiency. The gradient calculation is discussed in more detail in the following chapter.

It is in order to mention that almost any type of polygon can be used as a

control volume and that different types of polygons can occur in the same grid [29] [4][31]. The present study uses triangles for two-dimensional problems and tetrahedra in three dimensions. Just as for structured meshes, decoupling of the solution must be prevented by the addition of artificial dissipation [6][32] or upwinding. In the present research, both approaches have been implemented explicitly and implicitly. It is generally desirable for a scheme to be of second order spatial accuracy; thus, the values in the cell cannot be taken as constant [6].

If an upwind scheme is used, the gradient in the cell must be obtained to calculate the values of the flow variables at the cell faces [4] [8] to achieve second order accuracy. Barth [4] points out the difficulties in finding a consistent integration path associated with a cell-centered scheme. The gradients have to be reconstructed from the solution since they are not calculated as part of the unknowns. The calculation of the gradients is sometimes referred to as reconstruction. The gradient in a control volume can be calculated based on Equation (3.10). The integration path is over the control volume surface. It is desirable that the integration in the gradient calculation yields the correct gradient for a linear variation of the dependent variable and at the same time also holds for degenerate grids. A triangular grid is termed degenerate if the centroids of the cells surrounding a cell become collinear with the centroid of that particular cell.

A consistent integration path is shown in Figure 3.1. It represents overlapping control volumes, which does not constitute a problem. However, the approach would increase the complexity of the implementation of the algorithm substantially. In addition, the surface integral shown in Figure 3.1 is not implementable in a three-dimensional application in a straightforward manner.

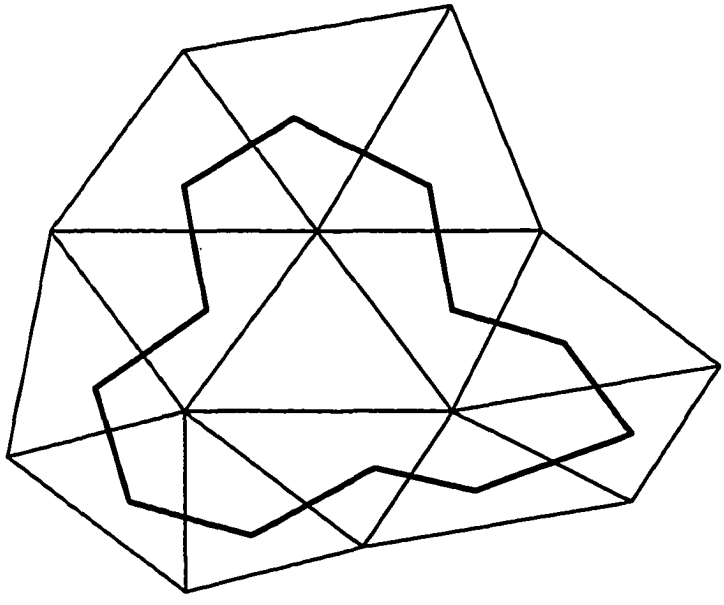

Figure 3.1: Integration path for first order reconstruction

A different approach to the gradient calculation was taken by Frink [3]. The basic idea of the method is to take advantage of the cell as the control volume. This requires the knowledge of the flow variables at the nodes. The nodal values must be obtained from the cell center values surrounding the nodes. This approach applied to the gradient calculation does not yield the exact gradient for a linear variation of the dependent variable. This is further discussed in section 3.3.2. However, good results and second order behavior have been reported for this method [3].

3.3.1 Discretization

The finite volume formulation given in Equation (1.2) lends itself for application to an unstructured discretization of the flow domain. Only grids consisting of

triangles (2-D) and tetrahedra (3-D) were considered in the current study. For each three-dimensional control volume, Equation (1.2) is evaluated as

$$\frac{\partial}{\partial t}V_i q_i + \sum_{j=1}^{n}((f-f_v)\eta_x + (g-g_v)\eta_y + (h-h_v)\eta_z)_j S_j \tag{3.6}$$

where q_i is the vector of unknowns in cell i, and f, g, h are the inviscid fluxes across face j. The subscript v denotes viscous terms, S_j is the area of face j and V_i is the volume of control volume i. The summation is performed over the n faces of the control volume.

3.3.1.1 Temporal discretization The temporal discretization of the time derivative term in Equation (3.6) is a first order forward difference:

$$\frac{\partial}{\partial t}V_i q_i = V_i \frac{q_i^{n+1} - q_i^n}{\Delta t} \tag{3.7}$$

3.3.1.2 Spatial discretization The normals, face areas and volume and normals of each control volume are calculated as shown below. Figure 3.2 serves as an illustration. The following expression applies to face area S indicated in Figure 3.2.

face area (face 123):

$$S = \frac{1}{2}|\vec{r}_{21} \times \vec{r}_{31}|$$

where $\vec{r}_{21}$ is the vector pointing from vertex 2 to vertex 1 and $\vec{r}_{31}$ is the vector pointing from vertex 3 to vertex 1. For example, the vector $\vec{r}_{21}$ is computed as $\vec{r}_{21} = (x_2 - x_1)\vec{i} + (y_2 - y_1)\vec{j}$. The expression for the other faces on the tetrahedron

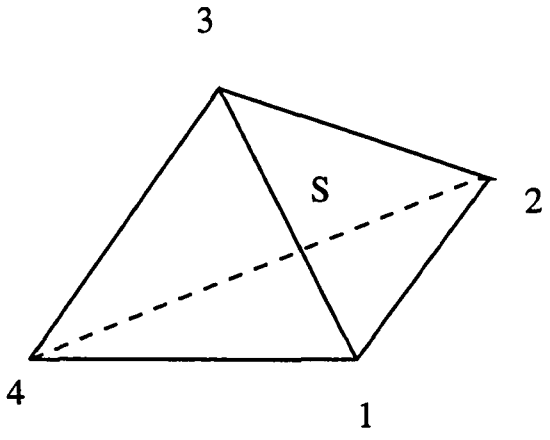

Figure 3.2: Illustration of 3-D control volume

is done accordingly.

volume:

$$V = \frac{1}{6}|\vec{r}_{41} \cdot (\vec{r}_{21} \times \vec{r}_{31})|$$

normals:

to insure the normals point out of the control volume, the sign of the following dot product is used:

$$sign = \frac{\vec{r}_{41} \cdot (\vec{r}_{21} \times \vec{r}_{31})}{|\vec{r}_{41} \cdot (\vec{r}_{21} \times \vec{r}_{31})|}$$

In the above equation, the term $(\vec{r}_{21} \times \vec{r}_{31})$ is twice the directed surface area. The

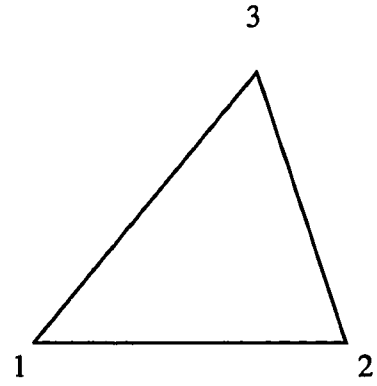

Figure 3.3: Illustration of 2-D control volume

vector $\vec{r}_{41}$ denotes the vector from the vertex opposite to face 123 (vertex 4) to a vertex on face 123 (vertex 1). In this case the vector is $\vec{r}_{41}$ which is the vector pointing from vertex 4 to vertex 1 is used in the calculation of the term *sign*. Vectors $\vec{r}_{42}$ or $\vec{r}_{43}$ could have also been selected. If the dot product $\vec{r}_{41} \cdot (\vec{r}_{21} \times \vec{r}_{31})$ is negative, the vector associated with the directed surface area points into the control volume. In the x-direction, the normal component is computed as:

$$\eta_x = sign \frac{(\vec{r}_{21} \times \vec{r}_{31}) \cdot \vec{i}}{S}$$

where, for example, $\vec{r}_{21}$ is the vector pointing from point 1 to point 2 in Figure 3.2.

The calculation of the geometry related terms is also presented for two-dimensional geometries for completeness. Figure 3.3 serves as an illustration.

In two dimensions the control volume surface consists of the three sides of the triangle and the control volume is the triangle itself. In two dimensions, the S and V in Equation (1.2) correspond to the length of the control volume faces and surface area, respectively. The expressions for the control volume surface and volume are given below.

area:

Control volume surface area S (unit height in the z-direction) of face 12 is:

$$S_{12} = \sqrt{\vec{r}_{12} \cdot \vec{r}_{12}}$$

volume:

control volume V (unit height in the z-direction) is computed as:

$$V = \frac{1}{2}|\vec{r}_{21} \times \vec{r}_{31}|$$

normals:

the components of a vector normal to face 31 are:

$$\vec{n} = (\vec{r}_{31} \cdot \vec{j})\vec{i} - (\vec{r}_{31} \cdot \vec{i})\vec{j}$$

which is the directed surface area of face 31. In order to assure that the normal (the vector $\vec{n}$ associated with the directed surface area) points of the control volume, the

sign of the following dot product is used:

$$sign = \frac{\vec{n} \cdot \vec{r}_{21}}{|\vec{n} \cdot \vec{r}_{21}|} \tag{3.8}$$

The vector $\vec{r}_{21}$ points from the vertex opposite to face 31 (vertex 2) to a vertex on face 31 (vertex 1). Instead of vector $\vec{r}_{21}$ (the vector pointing from vertex 2 to vertex 1), vector $\vec{r}_{23}$ could have been used in Equation (3.8).

The x-component of the normal pointing out of the control volume is

$$\eta_x = sign\frac{\vec{n}}{S} \cdot i$$

The calculation for the the $\vec{\eta}_y$ component of the normal is done similarly.

3.3.2 Gradient Calculation

For an upwind scheme of second order accuracy the flow variables must be linearly extrapolated to the control volume faces. The gradient in each control volume must be computed (reconstructed) from the flow variables. The gradient at the cell centers can be calculated by applying the Gauss divergence theorem:

$$\int_{CV} \vec{\nabla} q dV = \int_{CS} q\vec{\eta} dS \tag{3.9}$$

from which the gradient at the cell center can be calculated as is shown in Equation (3.10). The gradient is constant in each control volume.

$$\vec{\nabla} q_c \simeq \frac{1}{V_i} \int_{CS} q\vec{\eta} dS \tag{3.10}$$

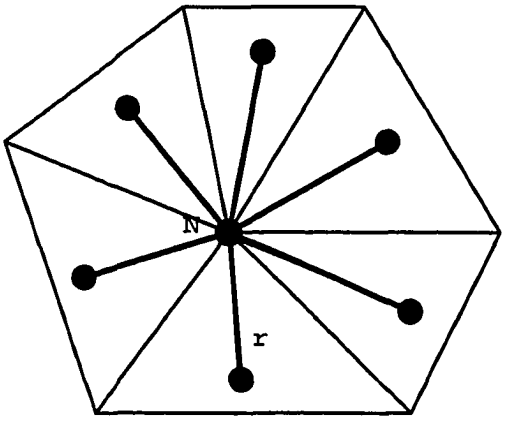

Figure 3.4: Distance averaged nodal values

This expression approximates the gradient at the cell center. Once the gradient is known at the cell center, the dependent variables at the cell faces are computed as:

$$q_f = q_c + \vec{\nabla} q_c \cdot \Delta\vec{r} + O(\Delta r^2)$$

The gradient could be computed using the Gauss divergence theorem applied to the integration path in Figure 3.1. Since this approach cannot be readily implemented in three dimensions due to its enormous complexity [8], another approach is taken in the current work.

In the current approach, the gradient calculation requires the nodal values of the flow variables. They are obtained by an inverse distance weighted average of the surrounding cell-centered values [3] as shown below.

$$qn = \frac{\sum_{i=1}^{n} \frac{q_i}{r_i}}{\sum_{i=1}^{n} \frac{1}{r_i}}$$

where n is the number of cells surrounding vertex N, and r_i is the distance from the cell center of cell i adjacent to vertex N and q_i is the volume averaged variable in cell i. This is also shown in Figure 3.4 for further reference. Based on the vertex values of the flow variables a gradient is constructed in the cell which is then used to compute the flow variables at the cell faces. For example, the gradient in a triangle is constructed as is shown in Equation (3.11) (also see Figure 3.5).

$$\vec{\nabla} q_c = \frac{\frac{1}{2}(q_1 + q_2) - q_3}{\frac{1}{2}(x_1 + x_2) - x_3}\vec{i} + \frac{\frac{1}{2}(q_1 + q_2) - q_3}{\frac{1}{2}(y_1 + y_2) - y_3}\vec{j} \tag{3.11}$$

The component of the gradient in the direction from the cell center to the centroid of the surface is

$$\nabla q_{cf} = \vec{\nabla} q_c \frac{\cdot \Delta\vec{r}}{\Delta r} = \frac{\frac{1}{2}(q_1 + q_2) - q_3}{3\Delta r}$$

where Δr is the distance from the cell center to the centroid of the cell face , q_3 refers to the variable q at vertex 3 in Figure 3.5, and $\Delta\vec{r}$ denotes the vector pointing from the cell center to the centroid of the cell face. This is illustrated in Figure 3.5 for the two-dimensional case.

In three dimensions, the corresponding expression is:

$$\nabla q_{cf} = \frac{\frac{1}{3}(q_1 + q_2 + q_3) - q_4}{4\Delta r}$$

In two dimensions the value of q computed at face f becomes:

$$q_f = q_c + \frac{1}{6}(q_1 + q_2) - \frac{1}{3}q_3$$

In three dimensions the value of q computed at face f becomes:

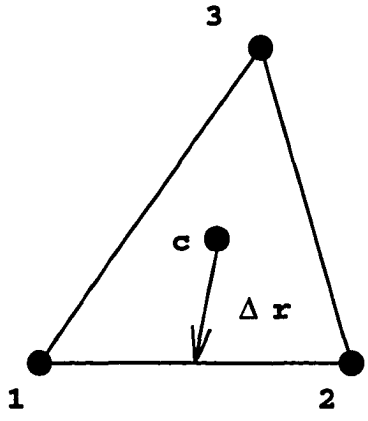

Figure 3.5: Illustration of gradient calculation

$$q_f = q_c + \frac{1}{12}(q_1 + q_2 + q_3) - \frac{1}{4}q_4$$

Note that in three dimensions the value of the variables at the node opposite to the face are denoted by q_4.

In the present research, the primitive variables (ρ, u, v, p) were calculated at the vertices and extrapolated from the cell center to the cell faces. Extrapolation of the primitive variables is reported to give better results than extrapolation of conserved variables or the fluxes [6]. The fluxes were then computed using the extrapolated primitive variables.

3.4 Calculation of the Time Step

Generally, local time stepping was employed to accelerate convergence. The time step in each cell was limited by the maximum allowable CFL number for the

particular scheme used. The following expression for a local time step for a cell centered scheme was derived by Frink [3].

$$\Delta t_i \leq \nu \frac{V_i}{A_i + B_i + C_i} \tag{3.12}$$

where

$$\begin{aligned} A_i &= (|u_i| + c_i)V_i^x \\ B_i &= (|v_i| + c_i)V_i^y \\ C_i &= (|w_i| + c_i)V_i^z \end{aligned} \tag{3.13}$$

V_i^x, V_i^y and V_i^z are the projected volumes of cell i in the x,y and z directions. The term c_i denotes the speed of sound in cell i. If time-derivative preconditioning is used, the expression in Equation (3.40) replaces the speed of sound.

The projected cell areas in the x direction are computed as

$$V_i^x = \sum_{j=1}^{n} \frac{1}{2}(\eta_x(j) + |\eta_x(j)|)S_j \tag{3.14}$$

where j is summed over all the faces of cell i and S(j) represents the face area. The projection of cell i in the y and z directions is done similarly.

3.5 Numerical Schemes

Both explicit and implicit schemes were used to solve to Navier-Stokes equations. The explicit scheme used a multistage Runge-Kutta time stepping procedure. The algebraic equations arising in the implicit scheme were solved using a block Gauss-Seidel solver. A central-difference scheme using artificial dissipation and the AUSM

scheme were implemented explicitly, whereas an implicit upwind scheme was applied in three dimensions. Time-derivative preconditioning was applied to the central-difference and the AUSM schemes.

For any grid, an explicit approach requires less computational effort per time step than an implicit approach. In the case of an explicit solver the flux calculation and flux accumulation can be performed in separate steps. The fluxes can be stored at the faces of the grid. The flux calculation and accumulation was done in loops through the faces of the grids.

For the implicit approach, the flux computation and matrix assembly were performed in one step. Loops needed to be performed through the array of unknowns. For a cell-centered scheme, these were the cells. Since each cell face in the interior of the domain was shared by two cells, all flux calculations were performed twice for each face if an implicit scheme is used. The redundancy of doing the flux calculations for each face twice can only be avoided by greatly increasing complexity of the algorithm and memory requirements. This of course, does not mean that explicit schemes offer more overall efficiency.

3.5.1 Central-Difference Scheme

A central-difference scheme was implemented explicitly. The central-difference approach was implemented by calculating the face values of the variables as the arithmetic average of the two adjacent centroidal values. Artificial dissipation needed to be used in conjunction with a central-difference scheme.

The inviscid terms required no gradient calculation in a central-difference scheme

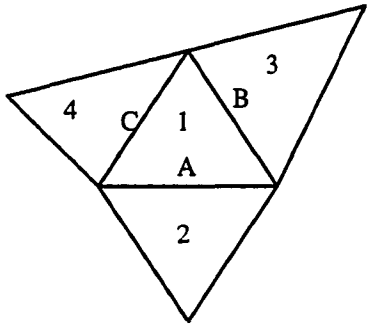

Figure 3.6: Stencil for central-differenced inviscid terms

since no extrapolation from the cell center to the faces of the control volume was performed. The approach was previously successfully implemented by Jorgenson [33]. A central-difference scheme can exhibit erroneous results due to decoupling of the dependent variables in a control volume from its neighbors. Remedies for the decoupling phenomena associated with central-difference schemes are the use of a staggered grid [30], or the use of artificial dissipation [32]. Since it was deemed to be desirable to avoid the complexity and memory intensity of a staggered grid, the use of artificial dissipation was preferred.

As an illustration, in a central-difference scheme, the control volume formulation of the governing equation is expressed in Equation (1.2):

$$\frac{\partial}{\partial t}\int_{CV} q dV + \int_{CS} \vec{F}\cdot\vec{\eta} dS = 0 \qquad (1.2)$$

The integration of the fluxes over the faces of the control volume is shown below

and is illustrated in Figure 3.6.

$$\int_{CS} \vec{F} \cdot \vec{\eta} dS \cong \frac{(F_1 + F_3)}{2} \cdot S_{1-3} \cdot \vec{\eta}_{1-3} + \frac{(F_1 + F_2)}{2} \cdot S_{1-2} \cdot \vec{\eta}_{1-2} + \frac{(F_1 + F_4)}{2} \cdot S_{1-4} \cdot \vec{\eta}_{1-4} \tag{3.15}$$

The time derivative term in Equation (1.2) is expressed as in Equation (3.7) . The required addition of artificial dissipation is described below.

3.5.1.1 Artificial Dissipation As mentioned above, artificial dissipation must be added to a central-difference scheme to avoid odd-even decoupling. The type of artificial dissipation used here is patterned after that used by Mavriplis [1]. Good results were obtained using this type of artificial dissipation in combination with the unstructured grid approach [1] [33].

With the addition of artificial dissipation, the conservation equation can be written as:

$$\int_{CV} \frac{\partial q}{\partial t} dV + \int_{CS} (f\eta_x + g\eta_y + h\eta_z) dS - D = 0 \tag{3.16}$$

where D represents the artificial dissipation terms. One inexpensive form of the artificial dissipation consists of the use of a combination of fourth and second order differences in the flow variables. This has been previously demonstrated to give good results by several authors [32] [1]. The sum of the differences across the control volume faces represents a second undivided difference. Likewise, the sum of the differences of the second undivided differences represents a fourth undivided difference.

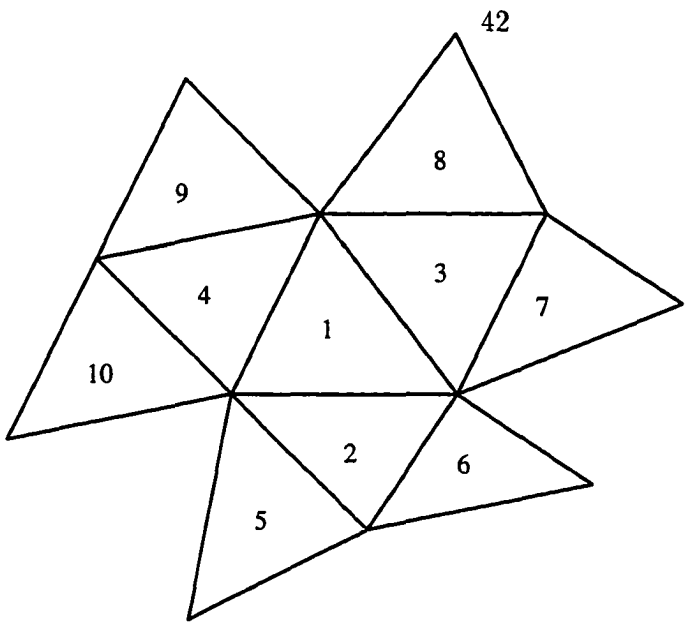

Figure 3.7: Stencil for central-differenced artificial dissipation

The same approach was implemented for triangular meshes. The undivided second derivative of the dependent variable in cell i is calculated as:

$$\nabla^2 q_i = \sum_{j=1}^{n} (q_j - q_i) \tag{3.17}$$

where the summation is over the adjacent cells (denoted by j's) of the control volume. Accordingly, the fourth undivided difference is computed as:

$$\nabla^4 q_i = \sum_{j=1}^{n} (\nabla^2 q_j - \nabla^2 q_i) \tag{3.18}$$

It is desirable to have the dissipation terms consistent with the control volume formulation used. The dissipation terms in Equation (3.16) are represented as a sum over the the faces of the control volume. For example, the dissipation term D_i for control volume i would be represented as a summation of the terms d_{ij} over the

faces (j) of the control volume.

$$D_i = \sum_{j=1}^{n} d_{ij} \tag{3.19}$$

Scaling of the artificial dissipation terms is necessary to obtain the desired behavior near discontinuities and allow convergence to steady state. The second undivided derivative is scaled by

$$\epsilon_{ij} = \epsilon^1 \frac{\sum_{j=1}^{n}(p_j - p_i)}{\sum_{j=1}^{n}(|p_j| + |p_i|)} \tag{3.20}$$

where ϵ^1 is set to zero for subsonic flow [1] and to one otherwise.

The fourth undivided difference is scaled by the factor

$$\epsilon_{ij}^2 = max(0, \epsilon^2 - \epsilon_{ij}) \tag{3.21}$$

where ϵ^2 is an empirical factor which was set to 1/32 [1]. The second undivided difference becomes dominant near shocks, where the fourth order difference becomes insignificant, or is even turned off. The fourth order difference is needed to allow for convergence in the region of significant gradients.

In order to properly scale the dissipation terms for any time step used, the artificial dissipation terms need to be premultiplied by the factor proportional to the time step. The calculation of the time step considers the absolute value of the largest eigenvalue as is shown in Equation (3.12). Hence it is legitimate to scale the dissipation terms with the largest eigenvalue or the time step itself.

Based on the preceding discussion about the exact form and scaling of the dissipation term, the contribution at each face from each of the neighboring cell j can

be represented as:

$$d_{ij} = \epsilon_{ij} \frac{S_{ij}}{\Delta t_{ij}} [q_j - q_i] + \epsilon_{ij}^2 \frac{S_{ij}}{\Delta t_{ij}} [\nabla^2 q_i - \nabla^2 q_j] \tag{3.22}$$

where the expression for S_{ij} and Δt_{ij} represent arithmetic averages of the control volumes and times steps, respectively of cell i and j. All quantities in the above equation are nondimensional.

3.5.2 Advection Upstream Splitting Method (AUSM)

An efficient flux splitting scheme was proposed by Liou and Steffen [47]. According to Liou and Steffen, the scheme attempts to combine the efficiency of flux-vector splitting schemes such as Van-Leer flux vector splitting and the accuracy of flux difference schemes such as Roe's scheme [6]. The formulation of the numerical flux is neither a flux vector splitting nor a flux difference splitting technique. In AUSM, the convective and pressure fluxes are treated separately. The scheme is based on the notion that the convective fluxes are convected across a face by an appropriately defined interface velocity $V_{1/2}$. The interface flux for supersonic flow is treated by taking the state on one side of the face, depending on the sign of the contravariant velocity. The following discussion applies only to subsonic flows.

The expression for the flux is then expressed as a function of the interface velocity $V_{1/2}$ and the states on either side of the face of the control volume. In a first order scheme, the flow variables are taken as constants in a control volume, whereas in a second order scheme the flow variables must be extrapolated from the cell centers to the cell faces. The pressure flux is treated separately since it is governed by the acoustic wave speed. This corresponding expression for the inviscid flux is shown in

Equation (3.23):

$$F_i = V_{1/2} \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho H \end{pmatrix}_{L/R} + \begin{pmatrix} 0 \\ \eta_x p \\ \eta_y p \\ 0 \end{pmatrix} \tag{3.23}$$

where L and R refer to either side of the control volume face. Liou and Steffen [47] chose to represent the interface velocity $V_{1/2}$ as a function of the wave speed $V \pm c$ traveling toward the cell face from the adjacent cells L and R. The interface velocity can then be written as:

$$V_{1/2} = c_{L/R} M_{1/2}$$

where the interface Mach number is expressed as:

$$M_{1/2} = M_L^+ + M_R^-$$

For a subsonic flow, the flux component to the left (L) and right (R) of the face are selected according to the direction of the convective interface velocity $V_{1/2}$. If the flux is convected from cell L to cell R, the expression in Equation (3.23) used the values of the variables from cell L.

Liou and Steffen [47] express the split Mach number $M^{\pm}$ as:

$$M^{\pm} = \pm \frac{1}{4}(M \pm 1)^2$$

The interface pressure flux is similarly expressed as a function of the states L and R on either side of the face:

$$p_{1/2} = p_L^+ + p_R^-$$

As with the convective term, a splitting based on the wave speeds $V \pm c$ is applied:

$$P^{\pm} = \frac{p}{2}(1 \pm M)$$

The AUSM scheme can be summarized as is shown in Equation (3.24):

$$F_{i,1/2} = \frac{1}{2}M_{1/2}[\begin{pmatrix} \rho c \\ \rho uc \\ \rho vc \\ \rho Hc \end{pmatrix}_L + \begin{pmatrix} \rho c \\ \rho uc \\ \rho vc \\ \rho Hc \end{pmatrix}_R] - \frac{1}{2}|M_{1/2}|[\begin{pmatrix} \rho c \\ \rho uc \\ \rho vc \\ \rho Hc \end{pmatrix}_R - \begin{pmatrix} \rho c \\ \rho uc \\ \rho vc \\ \rho Hc \end{pmatrix}_L] + \begin{pmatrix} 0 \\ \eta_x(p_L^+ + p_R^-) \\ \eta_y(p_L^+ + p_R^-) \\ 0 \end{pmatrix} \tag{3.24}$$

In the above equations, the local Mach number is based on the contravariant velocities associated with each face of the control volume.

$$M = \frac{\eta_x u + \eta_y v}{c}$$

The scheme has been applied to the Euler and Navier-Stokes equations by Liou and Steffen [47] and was reported to give results as good as Roe's scheme.

3.5.3 Implicit Upwind Scheme

An implicit Roe flux difference splitting scheme was implemented in addition to the central-difference scheme and AUSM described earlier in this chapter. The interpolation for the second order scheme is described below. The numerical flux of any upwind scheme can be written as the contribution of the flux vectors on both sides of the face plus an upwind term. For Roe's scheme in particular, this representation of the inviscid flux at a cell face is written as:

$$F = \frac{1}{2}[F(q^R) + F(q^L) - |\tilde{A}|(q^R - q^L)] \tag{3.25}$$

The variables q^R and q^L correspond to the state on either side of the face. In Equation (3.25) $|\tilde{A}|$ is the dissipation matrix and will be described below. The viscous fluxes will also be subjected to further discussion.

For an implicit time-integration, the discretized form of the equations is written as:

$$V_i \frac{\Delta q_i}{\Delta t_i} - \sum_{i=1}^{4} F_{ij}^{n+1} s_{ij} = 0 \tag{3.26}$$

where

$$\Delta q = q^{n+1} - q^n \tag{3.27}$$

A direct calculation of the flux at the new time level F^{n+1} in Equation (3.26) based on the expression in Equation (3.25) would be very complex and computer-time intensive for a second order scheme. It was deemed necessary to take a simplified approach as is described below. The expression can be linearized in time as shown

in Equation (3.28).

$$F^{n+1} = F^n + (\frac{\partial F}{\partial q})^n \Delta q \tag{3.28}$$

The calculation of the flux Jacobian $(\frac{\partial F}{\partial q})$ is required. For a second order scheme, a complete computation of the flux Jacobian in Equation (3.28) is very complex and computer time intensive since the flux calculation at the control volume faces must take into account the extrapolation of the flow variables from the cell center to the control volume faces. An approximation to the flux Jacobian was used instead. The approximation of the flux Jacobian in Equation (3.28) was taken to be equivalent to the Steger-Warming flux splitting which is described by Anderson et al. [38] and Hirsch [6]. The explicit part (F^n) of Equation (3.28) was computed using Roe's flux difference splitting. The scheme is first order accurate if the solution changes with time. Upon convergence to steady state, the scheme is second order accurate. This convergence behavior is a result of the terms on the left hand side in Equation (3.29) not being computed at the cell faces, but rather taken as constant in cells.

The discretized scheme is summarized in Equation (3.29) for cell i. The subscript m refers to a face of cell i and the corresponding neighboring cell.

$$[\frac{V}{\Delta t} I + \sum_{m=1}^{4} A^+(q_i) S_i] \Delta q_i + \sum_{m=1}^{4} A^-(q_m) S_m \Delta q_m = \frac{1}{2} \sum_{m=1}^{4} [F(q^R) + F(q^L) - |\tilde{A}|(q^R - q^L)]^n \tag{3.29}$$

A^+ and A^- are the forward and backward flux Jacobians. They are calculated from the expression in Equation (3.30).

$$A^+ = R\Lambda^+ R^{-1} \qquad A^- = R\Lambda^- R^{-1} \tag{3.30}$$

where Λ^+ and Λ^- are the diagonal matrices whose entries are the positive and the negative eigenvalues (λ), respectively, and R is the matrix whose columns are the right eigenvectors of the flux Jacobian.

The entries of Λ^+ and Λ^- are respectively:

$$\lambda^+ = \frac{1}{2}(\lambda + |\lambda|) \quad \lambda^- = \frac{1}{2}(\lambda - |\lambda|) \tag{3.31}$$

where λ denotes the eigenvalues of the Jacobian A. The right eigenvectors and corresponding eigenvalues are listed in appendix A.

For the inviscid terms shown in Equation (3.29), the implicit part of the equation was represented by the left hand side. The equations were solved for the deltas ($\Delta q = q^{n+1} - q^n$) of the unknowns in each cell. The resulting system of equations can be written as a matrix equation:

$$G\vec{x} = \vec{b} \tag{3.32}$$

3.6 Matrix Solution

In three dimensions, the matrix G is a block matrix containing 5×5 blocks since there are five simultaneous equations to be solved in each cell. The blocks contain the coefficients of the unknown quantities in the vector $\vec{x}$. The vector $\vec{x}$ contains the delta quantities of the unknowns and consists of 5×1 blocks. In general, as is shown in Equation (3.29), for each cell the left hand side involves terms from the four cells

surrounding the tetrahedron and from the tetrahedron itself. Thus, the general row of matrix G contains five entries (blocks). The number of blocks is different near the boundary. The vector b contains the right hand side of Equation (3.29). The vector b is also referred to as the residual or the explicit part.

If the viscous terms are treated implicitly, the number of nonzero blocks in matrix G increases to 17. The viscous terms are discussed below. In this study, the viscous terms were treated explicitly to reduce matrix storage requirements associated with implicit schemes.

The system of equations corresponding to the implicit upwind formulation given by Equation (3.29) is represented by Equation (3.32). As mentioned above, the matrix is sparse with nonzero blocks on each regular row. If a cell has a face on the boundary, there will be fewer entries on the row because the flux across the boundary face contains information from inside the computational domain. There is no regular pattern of the entries in the matrix A. A regular matrix pattern generally allows for well established and efficient methods [38], but there is also a large number of very efficient sparse matrix solvers available. In the present research, a block Gauss-Seidel matrix solver was applied to the system of equations represented by Equation (3.32). The reason why a Gauss-Seidel matrix solver was used was that it is independent of any matrix structure and allowed for a memory efficient solution method. Memory efficiency was also the reason why the viscous terms were computed explicitly. The block Gauss-Seidel solver allows the solving of a matrix equation with very small overhead. A block Gauss-Seidel method was also used by Jorgenson [33] for two-dimensional unstructured grids. In the Gauss-Seidel method for three-dimensional flows, the diagonal 5×5 block of matrix A is retained. The remaining matrix is

solved by a LU decomposition method.

3.6.1 Viscous Terms

In a second order scheme the gradients of the inviscid terms must be computed at the cell centers. In the viscous terms the gradients must be computed at the cell faces. The approach by Jorgenson [33] was used in the current study. The gradient at a particular cell face was computed by applying the expression in Equation (3.9) to the control volume established by the two cells adjacent to the cell of interest. The values at the cell faces of the control volume were obtained by averaging cell center values from adjacent cells. This is equivalent to a central-difference approach. It should be noted that the viscous gradient calculation is only second order accurate if the centroids of the control volume faces are located midway between two adjacent cell centroids and if the centroid of the cell face is the centroid of volume established by the two cells adjacent to it . For example, in Figure 3.8, the gradient at cell face C is calculated by integrating around the indicated path. Based on the reasoning above, the expression for the gradient at face c is given by Equation (3.33):

$$\begin{aligned}\nabla\vec{q_c} &= \frac{1}{V_1+V_4}((q_1+q_3)\cdot s_{1-3}\cdot\vec{\eta}_{1-3}+(q_1+q_2)\cdot s_{1_2}\cdot\vec{\eta}_{1-2} \\ &\quad +(q_4+q_9)\cdot s_{4-9}\cdot\vec{\eta}_{4-9}+(q_4+q_{10})\cdot s_{4-10}\cdot\vec{\eta}_{1-10}) \qquad (3.33)\end{aligned}$$

In Equation (3.33), $\vec{\eta}_{a-b}$ refers to the normal to side $a-b$ pointing out of the volume spanned by cells 1 and 4. Side $a-b$ is shared by cells a and b.

As with the inviscid fluxes, the viscous fluxes have to be integrated over the control volume surface. The discretized form of the integral is given in Equation

(3.34):

$$\int_{CS}(f_v\eta_x + g_v\eta_y)dS = \sum_{i=1}^{3}(f_{v_{i-j}}\eta x_{i-j} + g_{v,i-j}\eta y_{i-j})S_{i-j} \qquad (3.34)$$

where the subscript $i-j$ refers to the face shared by cells i and j. The derivatives in the viscous terms are calculated corresponding to Equation (3.33). For example, the term τ_{xy} contained in the viscous fluxes is:

$$\tau_{xy} = \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})$$

The term $\frac{\partial u}{\partial y}$ in $\tau_x y$ was computed as shown:

$$\begin{aligned}\frac{\partial u}{\partial y}_{1-4} &= \frac{1}{V_1+V_4}((u_1+u_3)\cdot s_{1-3}\cdot\eta y_{1-3} + (u_1+u_2)\cdot s_{1-2}\cdot\eta y_{1-2} \\ &\quad +(u_4+u_9)\cdot s_{4-9}\cdot\eta y_{4-9} + (u_4+u_{10})\cdot s_{4-10}\cdot\eta y_{4-10}) \qquad (3.35)\end{aligned}$$

Other terms in the viscous fluxes involving derivatives are calculated accordingly.

3.6.2 Runge-Kutta Scheme

The system of equations may be solved explicitly using a multistage Runge-Kutta time-stepping scheme. After n time steps the value of q_i is q_i^n. One can write the discretized set of governing equations (Equation (3.6) and (3.7)) as:

$$\Delta q_i = -R_i$$

where R_i and Δq_i are:

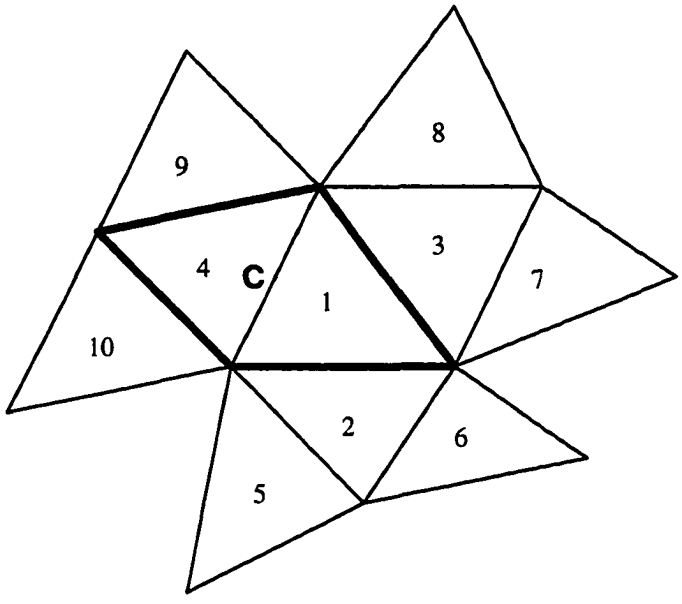

Figure 3.8: Stencil for viscous terms

$$R_i = \frac{\Delta t_i}{V_i} \int_{CS} \vec{F} \cdot \vec{\eta} dS$$

$$\Delta q_i = q_i^{n+1} - q_i^n$$

where V_i is the volume of cell i.

A m-stage Runge-Kutta scheme is implemented as follows:

$$\alpha_k = \frac{1}{m-k+1}, \quad k = 1, ..., m$$

where k is the k^{th} stage

After n time steps the value of q_i is q_i^n. Then, a new value of q_i is computed in m iterations as is illustrated below. The superscript in parentheses below denotes

the stage.

$$
\begin{aligned}
q_i^{(0)} &= q_i^n \\
q_i^{(1)} &= q_i^{(0)} - \alpha_1 R_i^0 \\
&\cdots \\
q_i^{(m-1)} &= q_i^{(0)} - \alpha_{m-1} R_i^{m-2} \\
q_i^{(m)} &= q_i^{(0)} - \alpha_m R_i^{m-1}
\end{aligned}
$$

After the calculation of m stages, the value of q_i at the new time level $n+1$ was updated to be

$$q_i^{n+1} = q_i^{(m)}$$

A three stage scheme was used for most calculations. If time accuracy was not required, local time stepping was used. In this study, only local time stepping was used. The calculation of the local time step was explained earlier in this chapter.

3.7 Low Mach Number Preconditioning

In two dimensions time-derivative preconditioning had to be used to achieve convergence for Mach numbers below 0.05. The flow in a channel with an obstruction was analyzed at a Mach number as low as 0.0005. Time-derivative preconditioning has been applied by several authors using a central-difference scheme with added artificial dissipation [34] [51]. In this study, time-derivative preconditioning was applied to a central-difference scheme using primitive and conserved variables. The preconditioning was also successfully applied to the AUSM scheme [47] using primitive variables.

The preconditioning matrix given by Choi and Merkle [34] was used in this study. If only the steady state solution was of interest, the time-derivative preconditioned system of equations was written as:

$$\Gamma \frac{\partial}{\partial t} \int_{CV} q_v dV + \int_{CS} \vec{F} \cdot \vec{\eta} dS = 0 \tag{3.36}$$

where q_v is the viscous set of unknowns:

$$q = \begin{pmatrix} p \\ u \\ v \\ T \end{pmatrix}$$

and Γ is the preconditioning matrix, which is given by Choi and Merkle [34] :

$$\Gamma = \begin{pmatrix} \frac{1}{\beta M^2} & 0 & 0 & 0 \\ \frac{u}{\beta M^2} & \rho & 0 & 0 \\ \frac{v}{\beta M^2} & 0 & \rho & 0 \\ \frac{(\rho E + p)}{\rho \beta M^2} - 1 & \rho u & \rho v & \frac{\gamma \rho R}{\gamma - 1} \end{pmatrix} \tag{3.37}$$

where $\beta = \gamma RT$ and M is the reference Mach number.

The preconditioned system of equations enhances the low Mach number performance of numerical solution procedures for the Navier-Stokes equations. For a time-derivative preconditioned approach, a new set of eigenvalues all having the same order of magnitude is obtained. It is believed that the new eigenvalues are the cause for the greatly improved convergence for low Mach number flows. For the unpreconditioned Navier-Stokes equations the eigenvalues of the Jacobians differ widely at low Mach

numbers (in Equation (3.41) becomes c very large compared to V). The eigenvalues corresponding to the Equations (3.36) are the eigenvalues of the linear combination of Jacobian matrices:

$$C = \eta_x A + \eta_y B \tag{3.38}$$

where $A = \partial f/\partial q$ and $B = \partial g/\partial q$. The eigenvalues of the preconditioned system of equations (Equation(3.36)) are the eigenvalues of the matrix $\Gamma^{-1}C$:

$$\lambda(\Gamma^{-1}C) = (V, V, \frac{V(1+\frac{\beta M^2}{\gamma RT}) \pm \bar{c}}{2}) \tag{3.39}$$

where $V = \eta_x u + \eta_y v$ and $\bar{c}$ is given by:

$$\bar{c}^2 = (u^2+v^2)(1-\frac{\beta M^2}{\gamma RT})^2 + 4\beta M^2 \tag{3.40}$$

The eigenvalues for the unpreconditioned flux Jacobian are:

$$\lambda(C) = (V, V, V \pm c) \tag{3.41}$$

The time step for the preconditioned system of equations was based on the largest of the new set of eigenvalues. The preconditioning with the matrix in Equation (3.37) assures the requirement that three eigenvalues are positive and one is negative for subsonic flows.

The implementation of a preconditioned upwind type of scheme is not straightforward since the left and right eigenvectors need to be evaluated in the course of the flux calculation. The new set of eigenvalues in the preconditioned approach is

accompanied by new sets of eigenvectors. For a central-difference scheme, the preconditioned approach does not require any special treatment in the flux calculation or the upwind terms. The scaling of the dissipation terms (see section 3.5.1.1) must be based on the preconditioned eigenvalue. The dissipation term is scaled by the largest eigenvalue which in turn is related to the time step used.

The time-derivative preconditioned system of equations was solved explicitly by multiplying Equation (3.36) by the inverse of the preconditioning matrix Γ^{-1}.

$$\frac{\partial}{\partial t}\int_{CV} q_v dV + \Gamma^{-1}(\int_{CS} \vec{F}\cdot\vec{\eta} dS) = 0 \tag{3.42}$$

where Γ^{-1} is:

$$\Gamma^{-1} = \begin{pmatrix} \beta M^2 & 0 & 0 & 0 \\ \frac{-u}{\rho} & \frac{1}{\rho} & 0 & 0 \\ \frac{-v}{\rho} & 0 & \frac{1}{\rho} & 0 \\ \frac{\frac{-\gamma}{\gamma-1}p+\frac{1}{2}\rho(u^2+v^2)+\rho\beta M^2}{\rho^2\gamma R}(\gamma-1) & \frac{-u(\gamma-1)}{\rho\gamma R} & \frac{-v(\gamma-1)}{\rho\gamma R} & \frac{\gamma-1}{\rho\gamma R} \end{pmatrix}$$

The preconditioning matrix in Equation (3.37) is for the primitive vector of unknowns. Preconditioning can also be applied using conserved variables by changing the variables by multiplying the preconditioning matrix by the Jacobian $\frac{\partial q}{\partial q_v}$. The resulting preconditioning matrix is:

$$\bar{\Gamma} = \begin{pmatrix} \frac{(\gamma-1)(u^2+v^2)}{\Phi} & \frac{(1-\gamma)u}{\Phi} \\ \frac{u}{\Phi}(\frac{(\gamma-1)(u^2+v^2)}{\Phi}-1) & 1+\frac{(1-\gamma)u^2}{\Phi} \\ \frac{v}{\Phi}(\gamma-1)(u^2+v^2))-v & 1+\frac{(1-\gamma)uv}{\Phi} \\ \frac{1}{2}(\gamma-1)(u^2+v^2)(\theta+2)-\frac{\gamma e}{\rho} & u(1-\gamma)(\theta+1) \end{pmatrix}$$

$$\begin{pmatrix} \frac{(1-\gamma)v}{\Phi} & \frac{(\gamma-1)}{\Phi} \\ \frac{(1-\gamma)uv}{\Phi} & \frac{(\gamma-1)u}{\Phi} \\ 1+\frac{(1-\gamma)v^2}{\Phi} & \frac{(\gamma-1)v}{\Phi} \\ v(1-\gamma)(\theta+1) & \theta(\gamma-1)+\gamma \end{pmatrix} \tag{3.43}$$

where

$$\Phi = 2\beta M^2$$

and

$$\theta = \frac{(e+p)}{\rho\beta M^2} - 1$$

An explicit scheme was used in this part of the study. The preconditioning was introduced into the solution by multiplying the flux vectors by the inverse of the preconditioning matrix. The same procedure was applied when preconditioning was used in conjunction with primitive variables.

The inverse of the conservative preconditioning matrix is given is Appendix B. Time-derivative preconditioning was also applied to the AUSM scheme [47] described earlier in this chapter. It was found that if preconditioning was applied to the AUSM scheme at lower Mach numbers, the pressure terms in the momentum equations had to be central-differenced to achieve convergence. The upwinding had to be based on the eigenvalues of the unpreconditioned Jacobian.

4. PARALLEL COMPUTING

In this chapter issues in parallel processing are discussed. The discussion begins with a presentation of different types of parallel machines and their processors. The key issues in the computational overhead due to message passing are addressed. Then, the domain decomposition and the actual parallel implementation and the associated data structure are reviewed.

Two- and three-dimensional versions of the unstructured solution algorithm described in the previous chapters have been implemented on a massively parallel computer and a workstation cluster. The massively parallel computer was a MIMD (multiple instructions, multiple data) parallel computer. On a MIMD parallel computer, several or even a large number of processors execute the program simultaneously and independently from one another. Each processor has its own memory, which is not accessible by the other processors. Communication between the different processors is necessary to properly model the solution domain. The architecture and operating system of a massively parallel computer is specifically designed for effective and high speed communication among the processors which is essential for gains in computational speed over traditional computers.

A different approach to having one computer designated for parallel applications is to combine traditional workstations into a network-based workstation cluster. A

workstation cluster is easily expandable and the workstations do not need to be exclusively used for parallel computing. When a parallel computing approach is used on a workstation cluster to perform calculations, several workstations execute the same algorithm simultaneously and independently. A workstation cluster uses the MIMD approach to parallel computing. Communication between the workstations takes place through a communications network.

The two-dimensional flow solver was implemented on the Ncube2s computer, and the three-dimensional flow solver was also implemented on the Ncube2s as well as on a workstation cluster consisting of RS6000 workstations. Both the two-dimensional and three-dimensional flow solvers are described in earlier chapters. The unstructured grid was partitioned according to coordinate location and required user input. Good communication efficiency had to be obtained iteratively. In order to implement the unstructured grid solver on the Ncube, the entire data structure had to be converted to a local data structure plus the communication data. The communication data are algorithm dependent and in some cases contain both cell and vertex data that need to be communicated between neighboring processors. For a central-difference and a first order upwind scheme, only cell data would need to be communicated. No global data reference needs to be retained. With this approach, the problem size that can be analyzed is only limited by the number of processors available for the calculation.

Parallel virtual machine (PVM) is the name of the communication software used on the workstation cluster. It is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. Thus, large computational problems can be solved by using the aggregate power of many computers acting as a distributed memory MIMD computer. The

PVM software supplies the functions to automatically start up tasks on the virtual machine and allows the tasks to communicate and synchronize with each other. A task is defined as a unit of computation in PVM analogous to a UNIX process. By sending and receiving messages, multiple tasks of an application can cooperate to solve a problem in parallel. PVM supports heterogeneity at the application, machine and network levels. In other words, PVM allows application tasks to exploit the architecture best suited to their solution. The PVM software handles all data conversions that may be required if different computers in the cluster use different integer or floating point operations. PVM also allows the virtual machine to be interconnected by a variety of different networks. The necessary software available to implement the message passing on the Ncube is available in a special library for parallel applications.

4.1 Parallel Processors

A MIMD parallel computer consists of real concurrent processors which are connected through communication channels. It has distributed memory, which means that each processor has its own memory which is not shared with other processors. There is no global memory and hence no global data space. A MIMD parallel computer can be categorized by the interconnection network topology. The communication network in a MIMD machine is static. Distributed-memory machines offer high levels of parallelism, but a message passing paradigm is required as input by the user.

One of the attractive features of a parallel approach to computing is that a parallel computer is scalable, at least in theory. This means that if the number of

processors is held proportional to the amount of computational work, the problem size can be increased without increasing the computational time.

Only a limited number of processors can be interconnected if the computer is to be scalable. Otherwise, the communication network will grow to unmanageable dimensions as the number of processors increases. A hypercube interconnection network is used in many commercial MIMD computers (iPSC,Ncube) because the number of links grows slowly as the number of processors grows. An n-dimensional hypercube contains 2^n processors connected by a link in each dimension. There are $n = log_2 2^n$ links at each processor. The hypercube topology is illustrated in Figure 4.1 for a three-dimensional hypercube (8 processors) and in Figure 4.2 for a four-dimensional hypercube (16 processors). As can be visualized, the connectivity in the relatively scalable hypercube topology becomes rather complex for even a moderate number of processors. If a message is passed from a processor to a processor other than a neighbor, the message must traverse the point-to-point links in the communication network to reach its destination. The number of point-to-point links a message must traverse is called the number of hops. The maximum number of hops a message must do in a $n-$dimensional hypercube typology is n.

On the Ncube, as with all MIMD machines, a program is executed in parallel by sending a copy of the executable to all the processors in the hypercube selected by the user. The data must also be distributed to the different processors by message passing or input/output operations. Since each processor should operate on different data, the data associated with the physical problem must be partitioned by some criteria.

The procedures involved in executing a program on a workstation cluster is

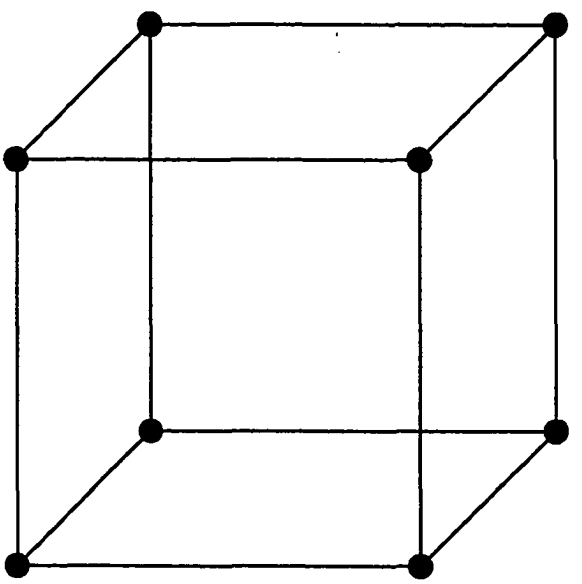

Figure 4.1: Three-dimensional hypercube topology

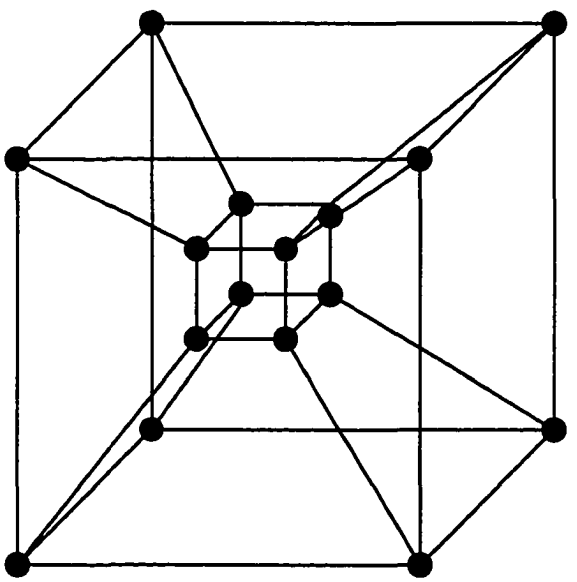

Figure 4.2: Four-dimensional hypercube topology

similar to those followed in using a MIMD parallel computer. However, multiple users are usually allowed to access a workstation in a workstation cluster whereas the processors on a massively parallel computer are usually reserved for single users. Since parallel computers tend to attempt to achieve their computational speed by utilizing a very large number of processors, the processors tend to have limited computational and memory capacity. On a workstation cluster, fewer processors (workstations) are used, but each has more computing capacity. Since computational fluid dynamics applications tend to have very large memory requirements, the computational work done on a processor of a massively parallel computer is smaller compared to that of a workstation. For smaller amounts of computational work, the efficiency is lowered by higher communication requirements. This observation is associated with the grain size of the problem and is described in the following sections. Grain size is the distribution of the parallelizable workload onto the processors of the parallel machine. If the grain size is large, potentially parallelizable tasks are performed sequentially by a processor. A finer grain size implies a larger overhead and the use of a larger number of processors. For example, if a processor is simply allocated a number of cells, loops over these cells are likely to be parallelizable in itself. If a larger grain parallelism is used, not every detail of the code is parallelized.

4.1.1 Communication among Processes

As is discussed above, communication among the processors is necessary in a distributed-memory computer. It should be considered a penalty to parallel computing because a serial computation does not need it. A distributed-memory computer is one in which each processor can directly address only a portion of the total mem-

ory of the system. Distributed-memory machines are also called message-passing systems. The concept of sharing variables among processors does not exist in this type of architecture. Therefore, when data must be shared in a distributed-memory environment, processes access the shared data by sending messages.

The basic operations needed for message-passing are send and receive. Sending and receiving messages are used to synchronize the different tasks on the different processors. As an illustration, consider processors P1 and P2 which share data. Communication is controlled by sending and receiving commands on processors P1 and P2, respectively.

P1; x: input request for the values of x from processor P2

P2; y: output of the value of y to processor P1

One way of assuring synchronization of the different processes is to make the receive statement blocking, which means that the receiving processor halts the execution of the program until the requested message has arrived.

Knowledge of the way the message passing works is essential in the implementation of the message passing procedure. If synchronizing message passing between processes is used, the problem that arises is often termed the producer-consumer problem [27]. The problem arises because the sender of data must store the data somewhere until the receiver requests the data. The problem is handled by using a message buffer. For the illustration above, process P1 would send data in variable x to a specifically reserved message buffer. The message buffer B receives the data in variable b, and processor P2 receives data from B in variable y. Processors P1 and P2 operate completely independently from one another until send or receive instructions are issued. Processor P1 would send a stream of values into the buffer as fast as it

can. Waiting is only necessary if the buffer cannot receive data fast enough. Processor P2 operates on its own until it requests to receive data for variable y from the buffer B. Waiting is only required if the buffer B cannot supply P2 with the requested data.

If all the memory available in the message buffer on a parallel computer is occupied with data, no data can be added to the buffer until some data are cleared by a read instruction. If the algorithm is such that more send commands are issued, the execution of the program will stall. If the size of the message buffer is not sufficient to accommodate all the data that need to be communicated between the processors, the algorithm must be adjusted by the user so that some data are cleared (read) from the buffer before it is filled. This usually results in an expense in algorithm efficiency because the initiation of a send command requires considerable CPU time. This is also referred to as message latency. It is advantageous to send fewer but longer messages. One way to achieve this is to pack the data before sending and then send it in a long message to the other processor. The data need to be unpacked at the receiving processor. The packing and unpacking of data also exacts an efficiency penalty, but the price to pay is small compared to sending every piece of data in a separate message. It is good parallel computing practice to send few, but long messages containing large amounts of data. However, tradeoffs might be necessary due to buffer space limitations.

4.2 Domain Decomposition

A copy of the program is executed asynchronously and simultaneously on all processors. Thus, distribution of the data should be in a manner that all processors

ideally have the same amount of calculations to do. This major issue in parallel computing is termed load balancing [27]. The time a calculation takes is dictated by the processor that finishes last. Idle time in some of the processors results if the computation times of different processors differ.

As mentioned earlier, the communication between the processor in parallel computing represents a penalty compared to traditional computers. The passing of data among processors takes several clock cycles and can potentially cause the performance of the calculation to deteriorate. In the Ncube, limited, but very effective processor interconnectivity exists. Neighbor to neighbor communication is more efficient than communication involving a large number of hops. Therefore, neighboring domains should also be on neighboring processors in the hypercube if possible. The amount of communication a processor has to perform should be kept small compared to the calculations it performs [27] [2] [20]. This consideration is termed the granularity of an algorithm. The larger the granularity, the smaller the penalty incurred by the necessary communication, but the parallelism is generally reduced as well.

There are probably an infinite number of ways to allocate data to different processors. In the calculations performed for this study, the domain was divided into connected sections with an approximately even number of cells. This should be efficient in terms of balancing the work load since the amount of calculations in a cell centered scheme is the same for each cell (e.g. this is not the case for a vertex based scheme). It is possible to copy the entire data structure of the domain onto all processors. Each processor would only use a fraction of the memory allocated. However, this approach would be very inefficient in terms of memory use. This is a severe limitation with the use of current MIMD computers since the amount of memory

associated with one processor is much smaller than on a workstation. The approach taken was to divide the entire domain into subdomains and to locally restructure the data structure. This complicates the use of the parallel computer tremendously, but improves the efficiency of the parallel computation.

If the domain is divided into connected parts, cells near the partition edges need information from cells located on another processor for the calculations. The geometry located on another processor is duplicated as phantom cells. Updated data are received from the neighboring processors and stored in the phantom cells before it is used in the next step of the calculation. In the procedure described, a message is the vector of flow variables.

In the approach used, numerical values at the cell faces were computed using information from the three vertices of a cell in two dimensions (four vertices in three dimensions) as is described in Chapter 3. The vertex values in turn were computed by inverse distance averaging the cell center values of all cells containing the particular vertex. The viscous terms do not increase the number of phantom cells (message passing). The message passing is explained in detail below.

4.3 Parallel Implementation

In the current method, the domain was partitioned into simply connected subdomains (see section 4.2). One of the partitioning criteria was that an equal number of cells was to be allocated to each processor. The amount of computational work was considered to be the same for all cells. The basis for this argument was that three flux calculations must be performed for each cell. Allocating an equal number of cells to each processor resulted in very good load balancing. Good performance on a

MIMD machine also requires a small communication overhead. The communication between the processors required in the current method can be explained on the basis of the adjacent cell information needed. As was described earlier, the gradients in the cell were obtained from the nodal values. The nodal values in turn were obtained by inverse distance averaging values in the cells sharing a particular vertex. Thus, if a cell on a particular processor had a vertex that was shared by a cell allocated to a different processor, the vector of unknowns for that particular cell was sent to the other processor.

In the case of the central-difference scheme, only cell information needed to be communicated between neighboring domains. In the case of a second order upwind scheme the gradient in a cell was calculated which changed the information that needed to be communicated between the processors. If the gradient in a cell needed to be calculated, cell information from cells on other processors that shared a vertex of a cell on another processor is not the only information that needed to be communicated. For a flux calculation, the flux also needed to be calculated on the opposite side of the face. The cell on the the opposite side may lie on another processor. The only information needed in addition to the communication described above is from the vertex opposite to the face with the cell allocated to another processor. If that particular vertex is not shared by a cell on the current processor, that information had to be communicated as well. The first communication step gave enough data to each processor so that the solution values of such a vertex could be calculated. With the goal of reducing the amount of communication, such vertex values were passed in a second step. This was more efficient than communicating all the information for another vertex calculation.

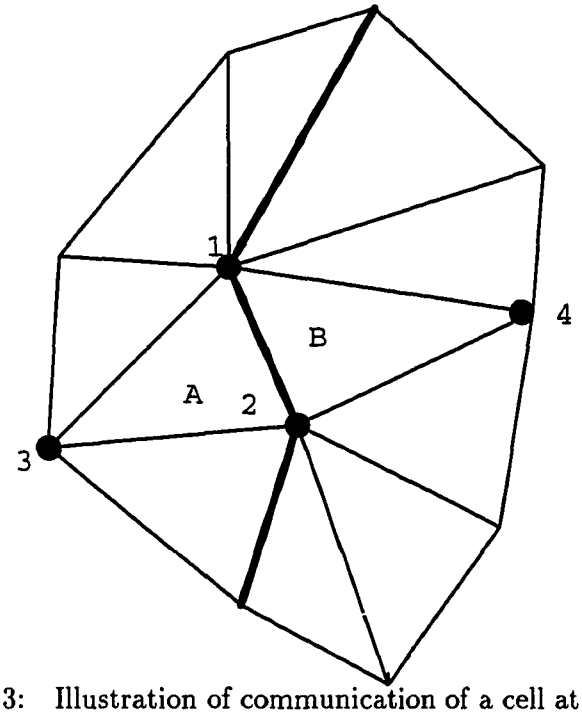

Figure 4.3: Illustration of communication of a cell at a subdomain boundary

This strategy is illustrated in Figure 4.3 . The thick line represents the processor domain boundary (cells A and B are allocated to different processors). The calculation of the nodal values for vertices 1 and 2 was done on both processors. All cells shown were needed for that calculation. The flux calculation for face A-B was also done on both processors. The flux calculation also involved information at vertices 3 and 4 which needed to be communicated to the other processor (e.g. information from vertex 3 to processor B)

The amount of communication increases as the size of the boundary between processors increases in terms of the number of boundary faces. Boundary minimization methods such as the reverse Cuthill-McKee algorithm [23] and the spectral partitioning algorithm of Pothen et al. [24] have been used successfully by several authors

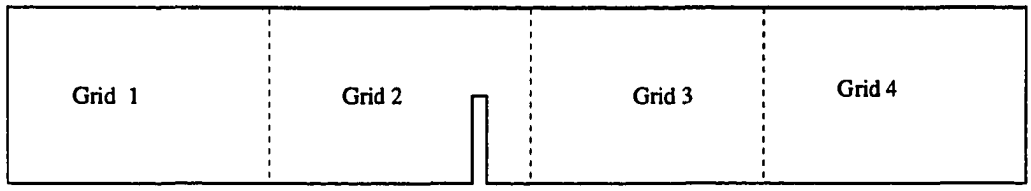

Figure 4.4: Illustration of domain subdivision for 4 processors

[22] [42].

In the approach used in the current study, the domain partitioning was done interactively. This approach is applicable when the entire grid is known prior to the partitioning step. In order to achieve good load balancing, several partitioning attempts were often made and compared. The load balancing criteria used in the current study was an allocation of an equal number cells to each processor. At least in two dimensions, the approach did not greatly increase the complexity of the calculation.

The domain partitioning procedure applied in this study is outlined below and is illustrated in Figures 4.4 and Figure 4.5 for four and eight processors, respectively. The flow over an obstruction in a channel is used as an example. This flow is further described in the next chapter. First, the flow domain or grid is divided into simple, approximate subdomains. For example, the flow domain shown in Figures 4.4 and

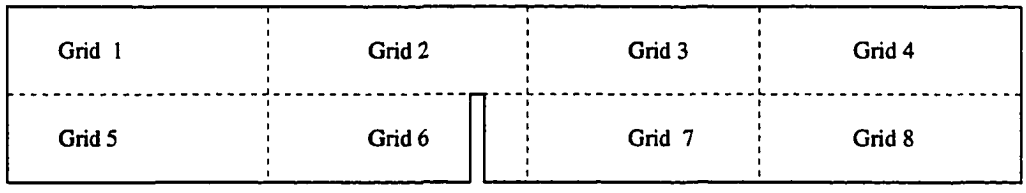

Figure 4.5: Illustration of domain subdivision for 8 processors

4.5 is divided into a one- and a two-dimensional Cartesian grid, respectively. The subdividing grid is indicated by the dotted lines in Figures 4.4 and 4.5. The goal of the subdivision is that each subdomian contains $NCELL/NSUB$ control volumes, where $NCELL$ stands for the total number of cells contained in the grid and $NSUB$ stands for the number of processors (partitions) used. If the cells are not evenly distributed among the processors, the subdivision of the grid is adjusted so that each subgrid contains approximately the desired number of cells. The adjustment of the the subdivision is done by altering the location the subdividing grid (the dotted lines in Figures 4.4 and 4.5).

4.3.1 Message Passing

This section describes some of the details of the implementation of the communication among processors. The message passing described below is applicable to PVM

as well as to the parallel library on the Ncube2s. A distributed-memory machine such as the Ncube or a workstation cluster can directly access only a portion of the total system memory. The concept of sharing variables does not exist in this type of architecture.

As is mentioned above, the basic operations needed for message passing are *send* and *receive*. The message passing can be used to control the parallel execution of an algorithm. The receive routines can be set to accept any message, any message from a specified source, any message with a specified message tag, or only messages with a specified message tag from a specified source. The message passing used in this research was blocking. The blocking message passing can be illustrated by revisiting the example given above. For example, processor P1 requests a value for variable x from processor P2 and processor P2 outputs the value of y to processor P1:

P1; x: input request for the values of x from processor P2

P2; y: output of the value of y to processor P1

In a blocking message passing environment, processor P1 will only execute the *receive* operation if it is requested and the *send* operation from processor P2 has already been executed. In other words, the value of y from processor P2 has to be in the message buffer. If the *receive* operation is executed on processor P1 before the *send* on processor P2, all execution of processes on processor P1 will wait for the arrival of the message from P2 in the message buffer.

When the message passing is blocking, the user can make sure that the data from a processor is sent to the correct data location on another processor by ordering the messages that are sent so that they are received in the same order or by using message tags. If the messages are sent and received in the same order, less residence time in

the message buffer is required which is good practice. If buffer space does not become a limitation, communication speed can be increased by *packing* the communication data. *Packing* the data to be sent means that the data are stored in an array which is then sent to the receiving processor. Naturally, the data need to be *unpacked* after having been received on the other processor. This increases memory requirements on both processors and the message buffer, but it has the advantage that longer and fewer messages can be sent. For each message that is sent several clock cycles of overhead is incurred.

5. RESULTS

Two- and three-dimensional testcases were computed. Two-dimensional analyses were performed using an explicit central-difference scheme and AUSM in conjunction with Runge-Kutta time stepping. Two-dimensional testcases computed on DEC workstations include a developing channel flow, a driven cavity with and without heat transfer, a symmetric expansion flow and a flow over an obstruction. The developing channel flow, the symmetric expansion flow and the flow over an obstruction were also solved using the Ncube2s parallel computer.

The three-dimensional testcases were a developing channel flow, a developing curved channel flow and a driven cavity flow. The implicit upwind scheme described in Chapter 3 was used in the three-dimensional calculations. The cases were computed on a CRAY-YMP. All cases except for the driven cavity were also computed on a workstation cluster. A summary of the cases analyzed and the approach taken in the analysis is shown in Table 5.1.

5.1 Two-dimensional Results

A two-dimensional explicit Navier-Stokes solver using central differences has been implemented on a DEC-5000 workstation and the Ncube2s massively parallel computer. The performance of the two machines was compared by solving the equivalent

Table 5.1: Detailed summary of cases analyzed (WS=workstation)

Two Dimensions			
Flow	Grid	Scheme	Computer
developing channel	7100	central difference	Ncube, WS
driven cavity	5200	central difference	WS
expansion flow	7600	central difference	Ncube, WS
flow over an obstruction	8600	central difference,AUSM	Ncube, WS
Three Dimensions			
developing channel	29260	upwind	LACE, CRAY- YMP
	14604	upwind	CRAY- YMP
driven cavity	64500	upwind	CRAY-YMP
curved channel	118000	upwind	LACE, CRAY-YMP

problem on the DEC 5000 and on the Ncube using four and eight processors. The Ncube gave the better performance even when only four processors were used. The number of cells used for the cases analyzed was relatively small so not much speedup could be achieved by using a cube with dimensions higher than 3 (eight processors).

Four subsonic flows were computed using the central-difference scheme with Runge-Kutta time stepping. This approach is described in Chapter 3. The first testcase was a developing channel flow at a Reynolds number of 10 based on the half width of the channel and the inlet conditions which were uniform. The flow geometry is shown in Figure 5.1. The flow was analyzed to gain experience with the unstructured grid approach and to establish confidence in the accuracy of the algorithm.

The flow analyzed had a uniform inlet velocity. A characteristic feature frequently used to assure the correctness of the solution to this problem is the velocity at the centerline of the channel. Good agreement with the data provided by Chen [26] is shown in Figure 5.2 . The channel geometry analyzed for this testcase had a

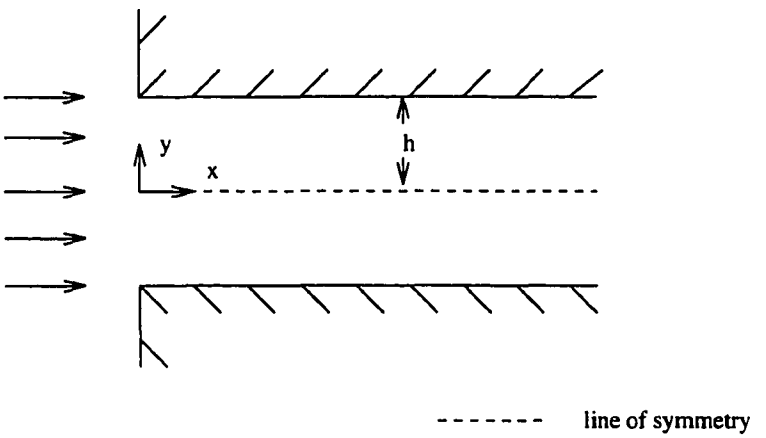

Figure 5.1: Geometry of a two-dimensional channel inlet

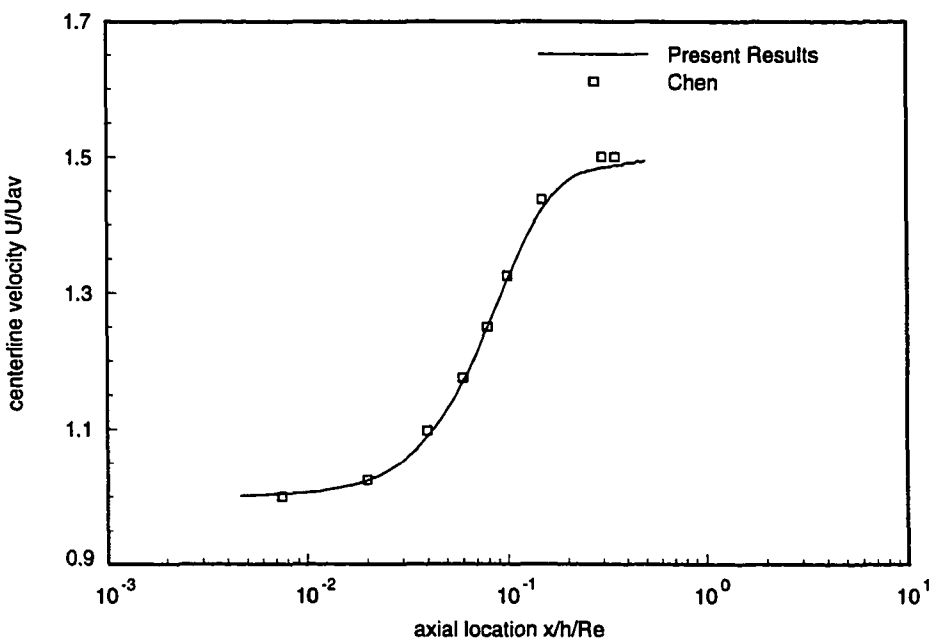

Figure 5.2: Centerline velocity for developing channel flow Re=10

length to height ratio of five and the grid contained 7100 cells (approximately 7100/4 cells per processor for four processors). Since the formulation used is for compressible flow, the velocities had to be corrected for the reduction of density along the channel in order to compare with incompressible flow results. The case converged in 2500 iterations and used 142 CPU minutes on the DEC. On the Ncube the CPU requirements were 128 minutes using four processors and 68 minutes using eight processors. For this case 420 messages between cell centers were exchanged at each iteration. Each message consisted of a 4x1 vector of double precision variables. When the problem was solved with eight processors, the number of messages that had to be passed approximately doubled. The profile of the computation on the Ncube is shown in Figures 5.3 and 5.4. As can be seen from the computer run profile, the time the processors spent on communication approximately doubled when the number of processors used for the calculation was increased from four to eight. This behavior of the computer run profiles was expected since the number of messages that had to be passed approximately doubled when the number of processors was doubled.

The second test case was a driven cavity flow at a Reynolds number of 100. The configuration for this flow is shown in Figure 5.5. The fluid is initially at rest in the square cavity. The top lid is suddenly moved at a constant velocity and the fluid inside begins to move due to viscous effects. The flow pattern becomes very complex and can only be analyzed using the full Navier-Stokes equations. The flow was analyzed to simulate an isothermal and incompressible flow. Time-derivative preconditioning was used to simulate near incompressible conditions. The flow was analyzed at a Mach number of 0.01. A grid using 5600 cells with some clustering near the walls was used to compute the flow. The isothermal two-dimensional driven

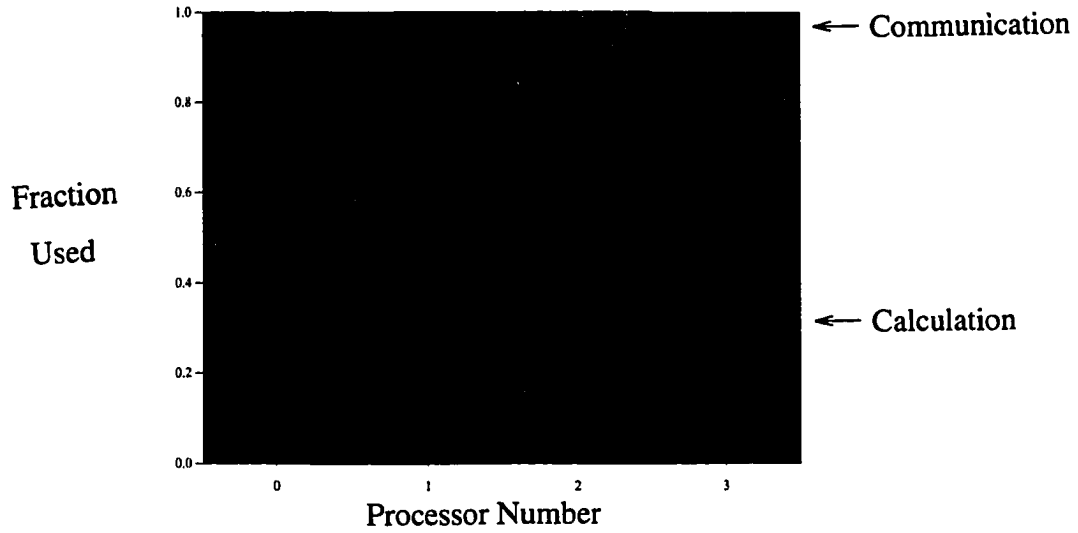

Figure 5.3: Profile for channel flow on four processors

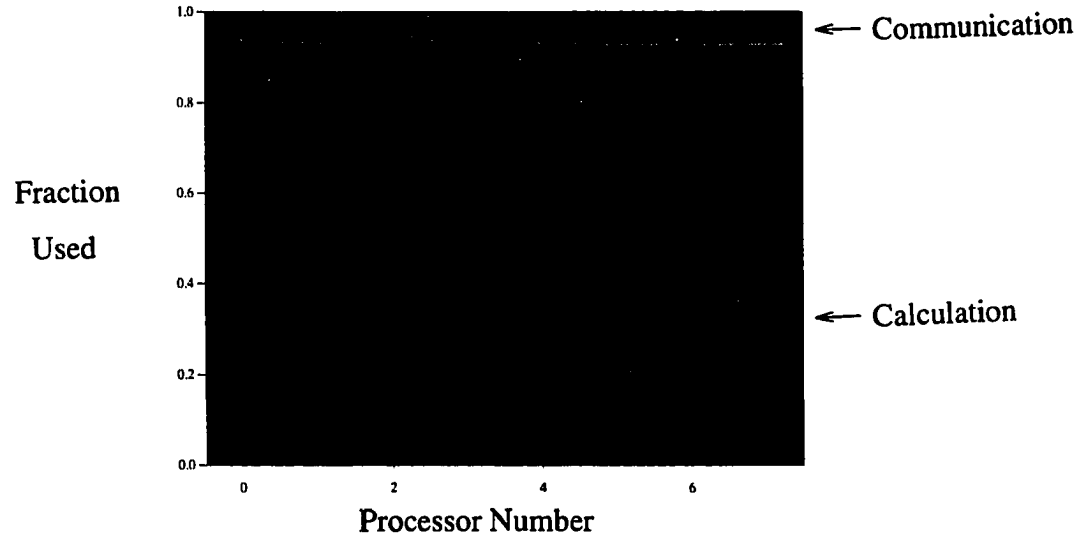

Figure 5.4: Profile for channel flow on eight processors

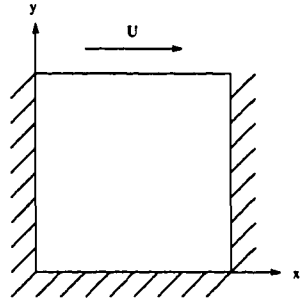

Figure 5.5: Geometry of two-dimensional driven cavity flow

cavity flow was previously analyzed by many researchers including Chen [26] and Ghia et al. [50]. The velocity along the vertical centerline of the driven cavity serves as a basis for comparison of the results. As can be seen in Figure 5.6, the present results agree well with the results obtained by Chen [26] and Ghia et al. [50]. Chen [26] used a 39 × 39 grid for the analysis and Ghia et al. used a 129 × 129 grid. The velocity vectors for the flow are shown in Figure 5.7.

Heat transfer phenomena were also studied for the driven cavity problem described above at a Reynolds number of 100. The same grid as for the isothermal flow was used in this test case. The reference Mach number used was also 0.01 to simulate incompressible conditions. The stationary wall temperature (T_w) is held at a temperature cooler than that of the top moving wall (T_t). The top moving wall temperature is held at a constant value. The temperature of the stationary walls

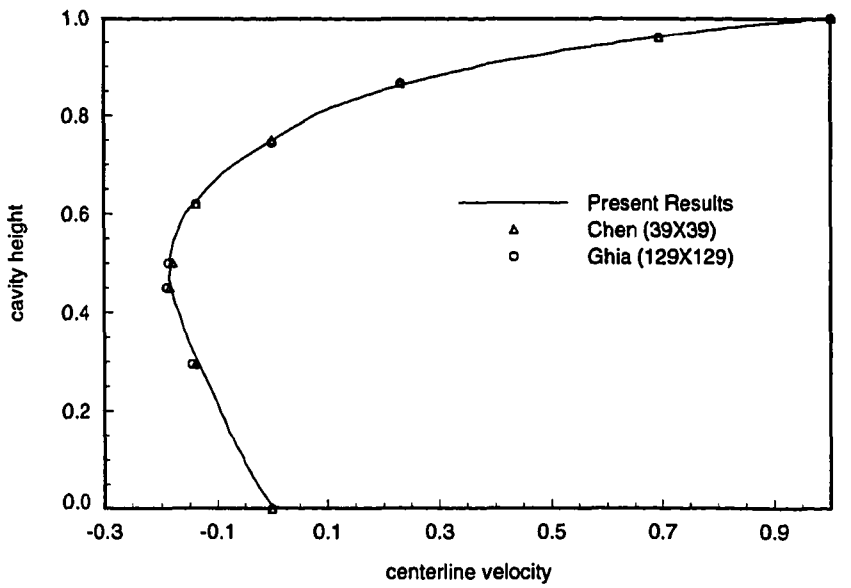

Figure 5.6: Calculated velocity profile along the vertical centerline of the two-dimensional driven cavity for Re=100

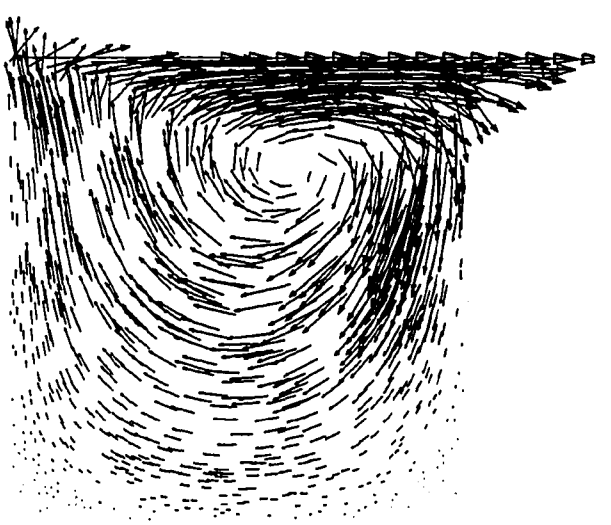

Figure 5.7: Calculated velocity vectors for a two-dimensional driven cavity for Re=100

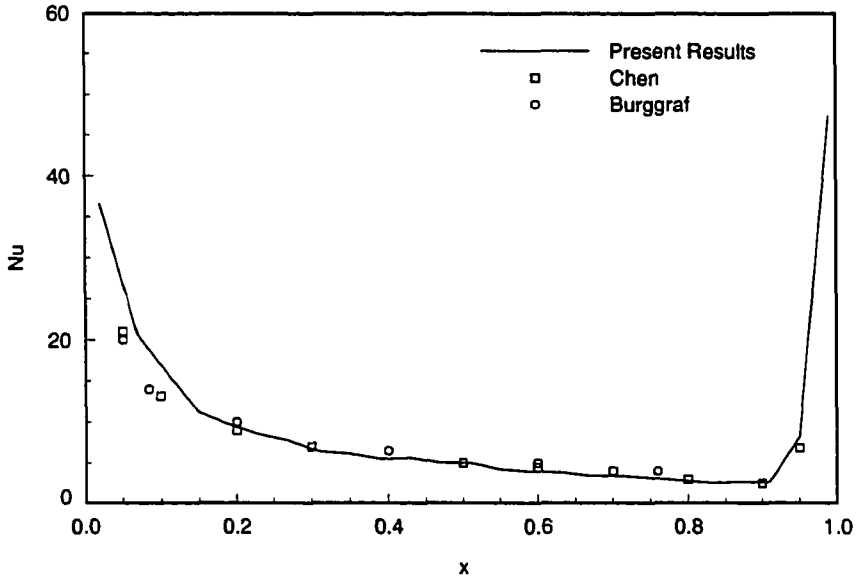

Figure 5.8: Local Nusselt number at the top wall of the two-dimensional driven cavity

was $T_w = 0.9 \times T_t$. The local Nusselt number along the moving wall is used for comparison to other researchers' results.

The local Nusselt number calculated in the present analysis is shown in Figure 5.8 together with the results obtained by Chen [26] and Burggraf [48]. Chen [26] used a 21 × 21 grid the calculation and Burggraf [48] used a 41 × 41 grid. The present results compare well with the results of Chen [26] and Burggraf [48].

The local Nusselt number shown in Figure 5.8 is defined to be:

$$Nu = \frac{hL}{k}$$

where L is the height of the cavity, k is the thermal conductivity and h is the heat transfer coefficient based on the temperature difference between the stationary wall

and the moving wall ($\Delta T = T_w - T_t$).

The third test case was a sudden expansion flow. This flow was computed on a DEC 5000 workstation and on the Ncube2s massively parallel computer. The channel cross sectional area of the incoming flow suddenly expands by a factor of three. The flow was computed for a Reynolds number of 56 based on the upstream channel height and the centerline velocity. A fully developed velocity profile was used as the upstream boundary condition. The inlet for this flow was located one step height upstream of the expansion.

The domain analyzed extended three full channel heights (based on geometry after the expansion) beyond the expansion. The grid contained 7500 cells. A symmetry boundary condition along the channel centerline was used for the calculation. A velocity vector plot of the solution is shown in Figure 5.9 . Velocity profiles at different locations behind the expansion are shown in Figures 5.10 and 5.11 along with the results of Jorgenson [33] and Durst et al. [49]. Good agreement with the numerical results provided by Jorgenson [33] and the experimental results provided by Durst et al. [49] was found. The velocity vector plot for the region of the flowfield right behind the step gives a good qualitative representation of the correctness of the solution. The reattatchment length of the separation bubble is four times the step height which is within one percent of the value found by TenPas [39].

The converged solution of the sudden expansion flow was obtained after 17000 time steps. On the DEC 5000 this represents 960 CPU minutes and on the Ncube the times were 890 min and 481 min for four and eight processors respectively. When this case was solved on the Ncube using four processors, 400 messages were exchanged at each iteration.

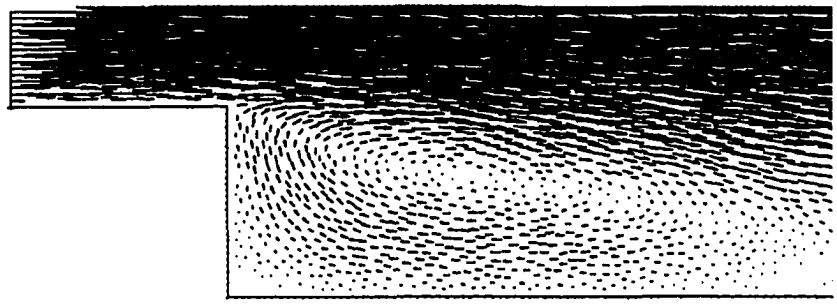

Figure 5.9: Calculated velocity vectors for expansion flow

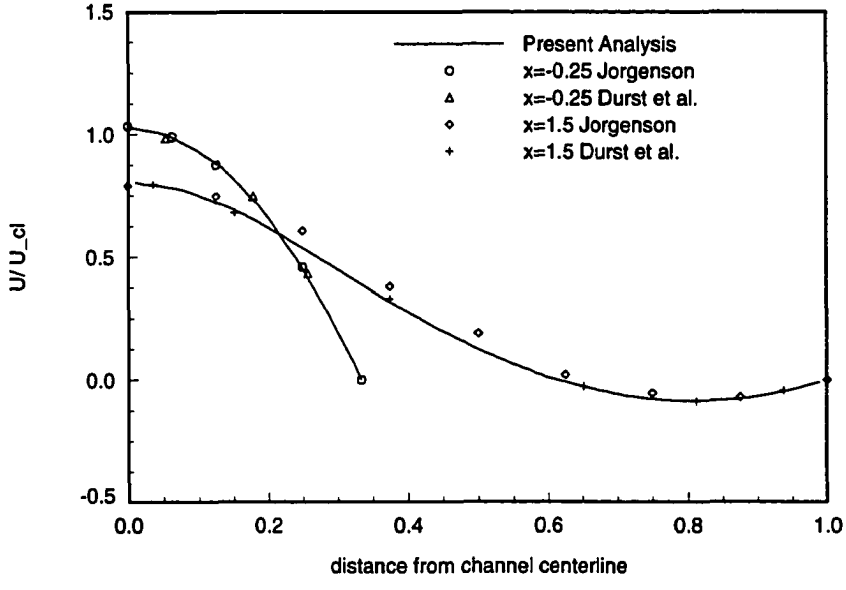

Figure 5.10: Velocity profiles for a laminar flow with a 3:1 symmetric expansion, $Re = 56$

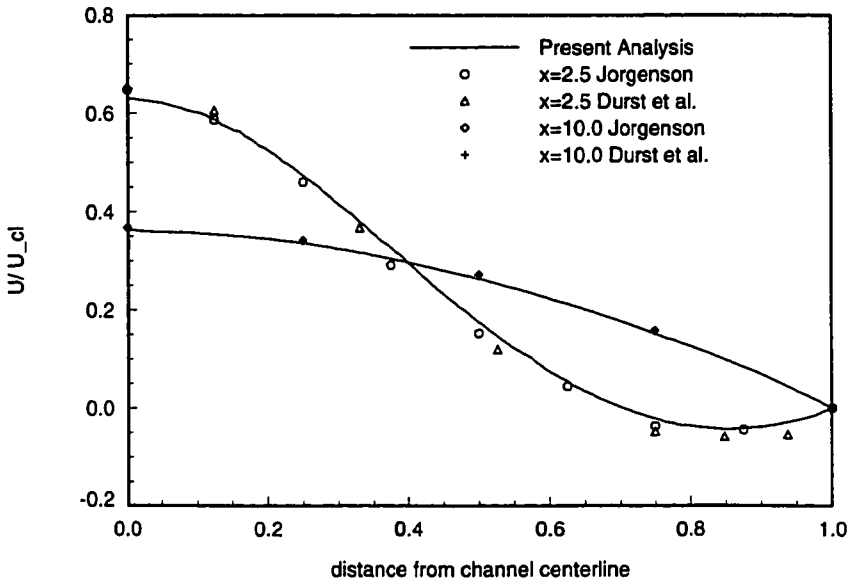

Figure 5.11: Velocity profiles for a laminar flow with a 3:1 symmetric expansion, $Re = 56$

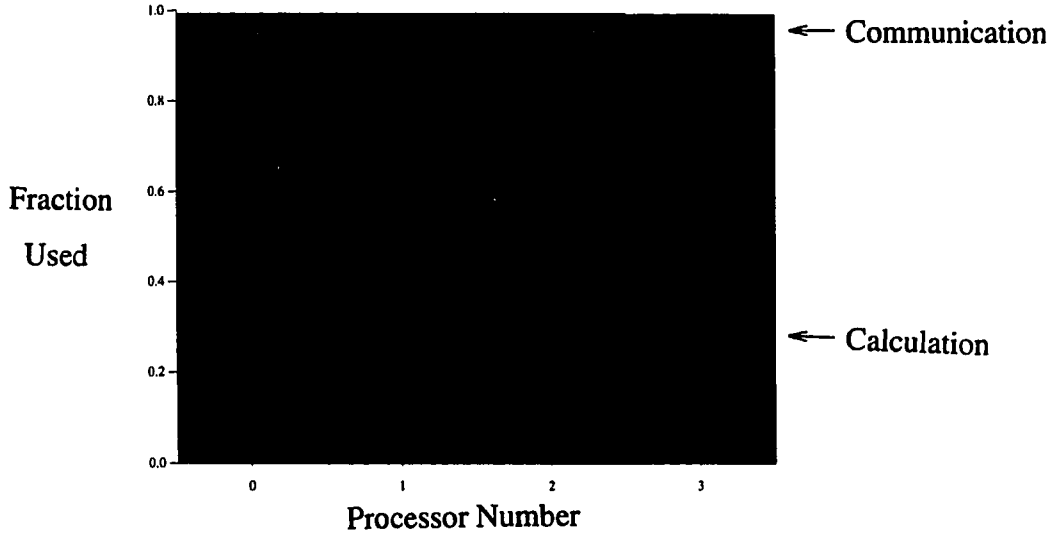

Figure 5.12: Profile for expansion flow on four processors

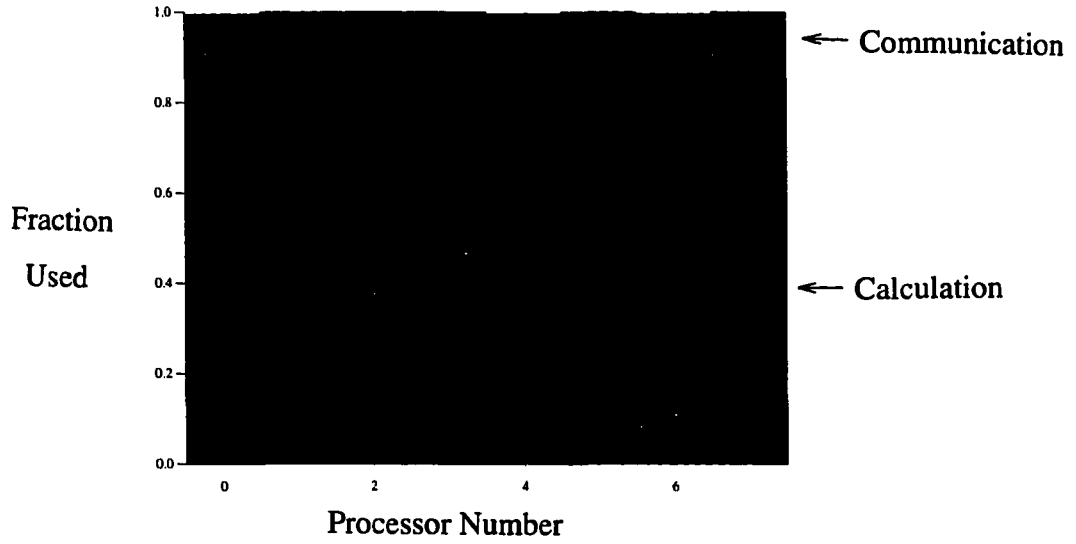

Figure 5.13: Profile for expansion flow on eight processors

The profile of the computer runs on the Ncube are shown in Figures 5.12 and 5.13 for the symmetric expansion flow for four and eight processors, respectively. As was expected, the time spent on communication was higher for the computer run using eight processors preventing a linear speedup with the number of processors. The number of processors used was small enough so that very good speedup could be achieved by doubling the number of processors.

The fourth two-dimensional testcase analyzed was a channel with an obstruction. This problem was used to compare the performance of several numerical schemes over a range of Mach numbers. The flow was computed using the explicit central-difference scheme and the AUSM scheme described in Chapter 3. Time-derivative preconditioning was applied to the central-difference formulation using primitive as well as conserved variables and to the AUSM scheme using primitive variables.

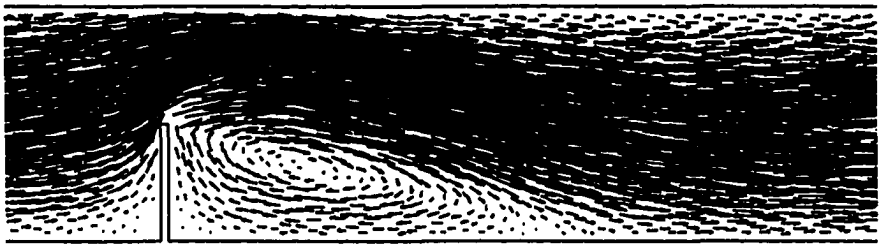

Figure 5.14: Calculated velocity vectors for channel flow with obstruction, Re=100

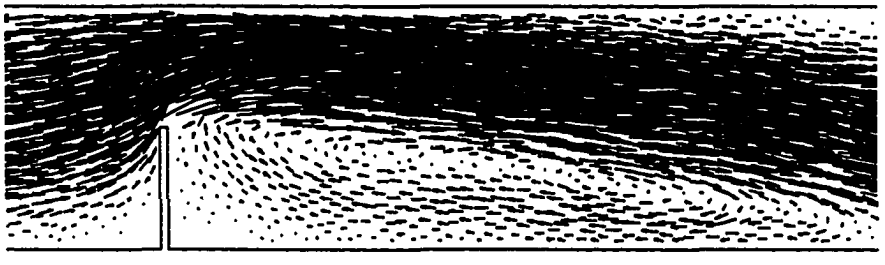

Figure 5.15: Calculated velocity vectors for channel flow with obstruction, Re=500

The nondimensional channel height was 1.0 and the nondimensional height of the obstruction was 0.5. The length of the obstruction was 0.03. A uniform grid with 7688 cells was used. The case was analyzed for Reynolds numbers of 100 and 500 based on inlet conditions and the channel height. The reattatchment length after the obstruction was used as a basis for comparison with data found in the literature. The flow was calculated for several Mach numbers ranging from 0.1205 to 0.0005. The system of governing equations had to be preconditioned to achieve convergence at such a low Mach number. The time-derivative preconditioning developed by Choi and Merkle [34] described earlier was used. A central-difference scheme with added fourth order artificial dissipation as well as AUSM were employed to simulate the flow.

The flow was computed for Reynolds numbers of 100 and 500 based on the full channel height and uniform inlet conditions. A basis for comparisons to data found in the literature is the reattachment length of the flow after the obstruction. The reattachment lengths were calculated to be 1.7 and 3.6 times the full channel height for Reynolds numbers of 100 and 500, respectively. The results obtained agreed well with those obtained by Nallasamy [43] who computed the reattatchment lengths of 1.65 and 3.5 times the full channel height for Reynolds numbers of 100 and 500, respectively. The velocity vector field for this flow is shown for Reynolds numbers of 100 and 500 in Figures 5.14 and 5.15, respectively.

The convergence histories are shown in Figure 5.16 for the central-difference scheme. They are essentially identical for the two Reynolds numbers (Re=100 and Re=500 based on the channel height and inlet conditions). Convergence was achieved in 3000 iterations for both Reynolds numbers. The unpreconditioned scheme did not

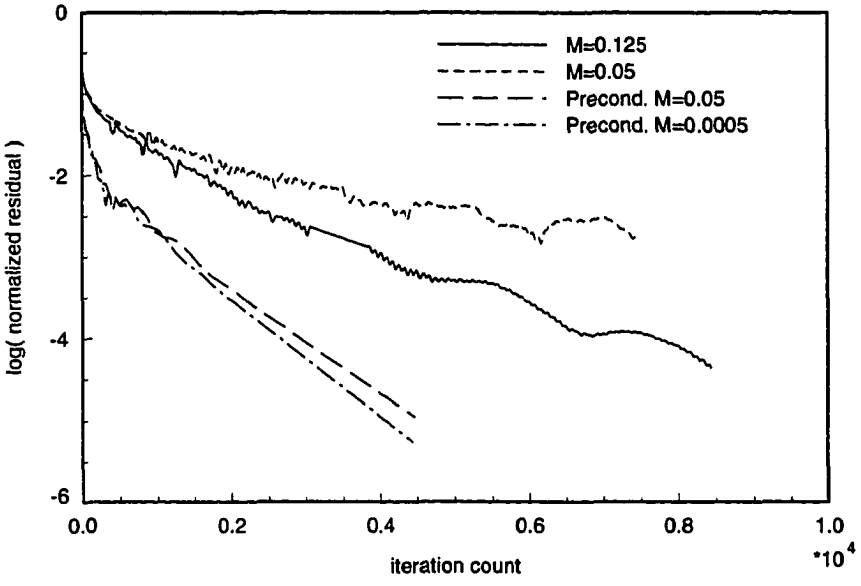

Figure 5.16: Convergence for the channel flow with obstruction with and without preconditioning (central differences), Re=500

converge at Mach numbers below 0.05. As can be seen, the convergence for a preconditioned scheme is enhanced for low Mach numbers and is nearly Mach number independent. The convergence histories for the preconditioning central-difference scheme using conserved and primitive variables are shown in Figure 5.17. The corresponding preconditioning matrices are given in Equation (3.37) and Equation (3.7), respectively. The time-derivative preconditioning also enhanced convergence and caused the convergence to be Mach number independent when conserved variables were used (see Figure 5.17).

The same results were obtained for the calculation of the flow over an obstruction in a channel, regardless if primitive or conserved variables were used. Both approaches were equally robust. The preconditioning matrix corresponding to the conserved variables (Equation 3.37) is not as sparse as for the primitive variables (Equation

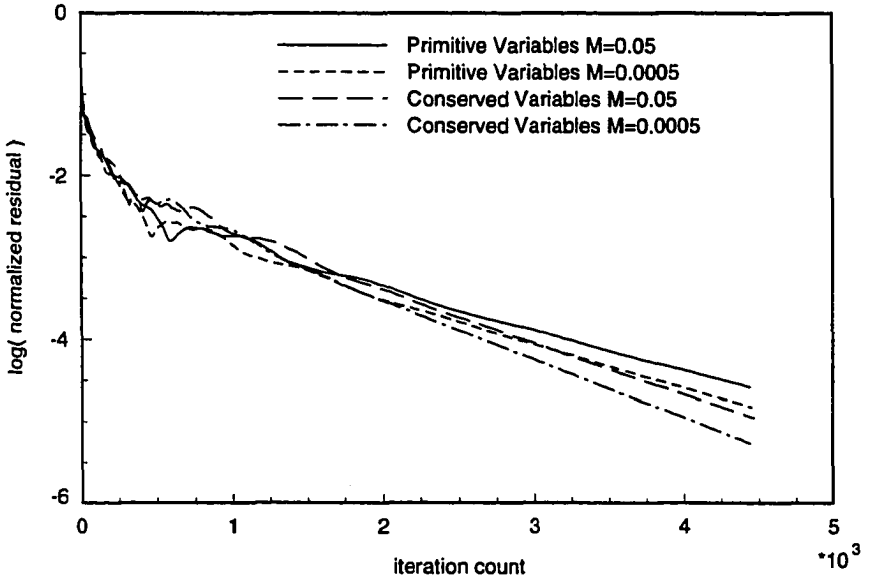

Figure 5.17: Convergence for the channel flow with obstruction for different variables (central differences), Re=500

3.7). This resulted in slightly higher computing time requirements.

The flow was also analyzed using the preconditioned AUSM scheme. The results obtained were identical to those obtained when the central-difference scheme was used. The convergence histories are shown in Figure 5.18 and demonstrate the merits of using time-derivative preconditioning at low Mach numbers such as Mach number independence and enhanced convergence.

The flow calculation was timed on a DEC 5000 workstation and on the Ncube computer using 4 and 8 processors. The timed results were obtained for the preconditioned explicit central-difference scheme at a reference Mach number of 0.0005 using primitive variables. The case required 200 CPU minutes on the DEC 5000 workstation and 190 and 100 CPU minutes on the Ncube using 4 and 8 processors, respectively.

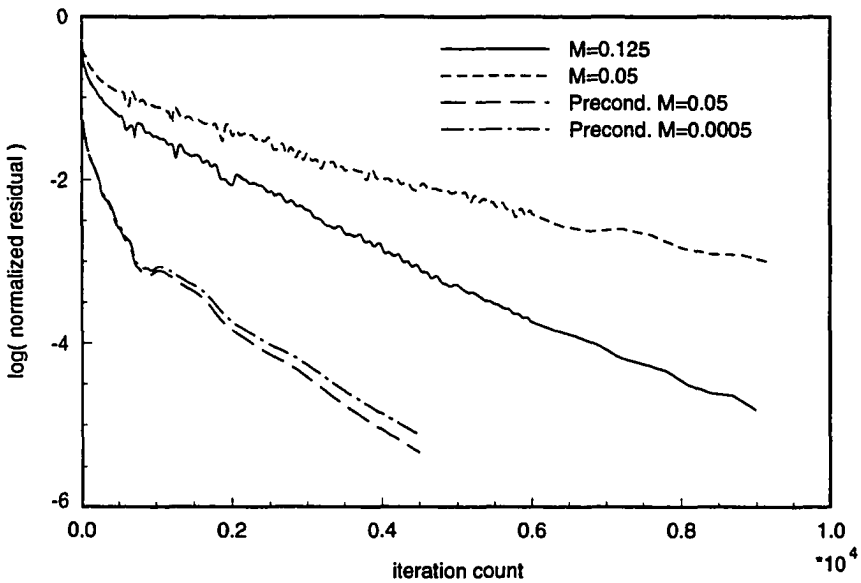

Figure 5.18: Convergence for the channel flow with obstruction with and without preconditioning (AUSM), Re=500

5.2 Three-dimensional Results

The developing flow in a rectangular channel was calculated to demonstrate the accuracy of the method in three dimensions. The developing channel case has been studied by several investigators including Gegg [40] and TenPas [39]. An analytical expression for the fully developed velocity profile for an incompressible flow has also been developed [41]. This type of flow is rather well suited for a structured grid but was thought to prove as a case well suited to point out differences between the structured and unstructured approach.

All three-dimensional unstructured grids used in the this research were generated using the software package Vgrid3d. The version of Vgrid3d used (version 1.0) did not allow for a stretching of the tetrahedra. Only the triangle size could be varied. As

indicated in the introduction, control volumes with high aspect ratios lend themselves for the analysis of viscous flows since the flow gradients are generally predominant in one direction.

A suitable structured grid for this testcase is Cartesian with gridpoints clustered near the inlet and at the walls. The grid is stretched in the streamwise direction as the streamwise gradients diminish.

The program Vgrid3d [52] was used to generate the unstructured grid. The version of Vgrid3d (version 1.0) used in this research did not allow for control over the aspect ratio of the control volumes. Since no stretching of the individual tetrahedra was possible, the cell size was dictated by the need for the resolution of the gradients in the crosstream directions. Thus, the grids used had to be nearly uniform, which is rather inefficient in comparison to a stretched structured grid. For example TenPas [39] used a 11 $\times$ 11 $\times$ 41 grid for one quarter of the cross section of a channel. For a full channel height to length ratio of 5, this grid would correspond roughly to a 11 $\times$ 11 $\times$ 110 grid if a uniform grid was assumed. It is clear that the number of cells in an unstructured grid could easily become excessive if a uniform or nearly uniform unstructured grid was used for this type of flow.

The developing flow in a channel was calculated using a channel height to length ratio of 5. The calculations were performed for a 14604 cell and a 29260 cell grid. The flow was computed for the entire channel. The flow geometry is shown in Figure 5.19.

One of the characteristics of the flow is the centerline velocity along the channel axis. The computed centerline velocity is shown in Figure 5.20 along with the results of TenPas [39]. The developing channel flow was analyzed at Reynolds numbers of 5

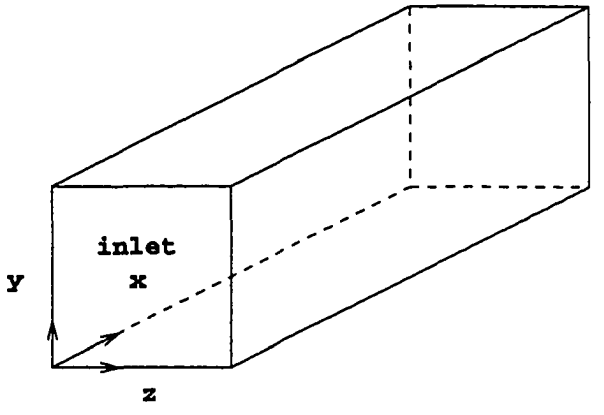

Figure 5.19: Geometry for developing channel flow

and 25. The Reynolds number was based on half of the channel height and the inlet conditions. A uniform inlet velocity profile was used in the current analysis as well as by TenPas [39].

The results shown are for the fine grid (29260 cells) for a Reynolds number of 5 and 25. For the Reynolds number of 5 case, the results are also shown for the coarse grid (14604 cells). The results agree well with the results obtained by TenPas [39] except that the final centerline velocity tends to be somewhat higher compared to the results obtained by TenPas [39] (2 % higher). TenPas [39] used a higher inlet velocity to achieve the final centerline velocity.

Since the results in the present calculation were obtained using a compressible formulation the centerline velocity had to be corrected for the change in density. The

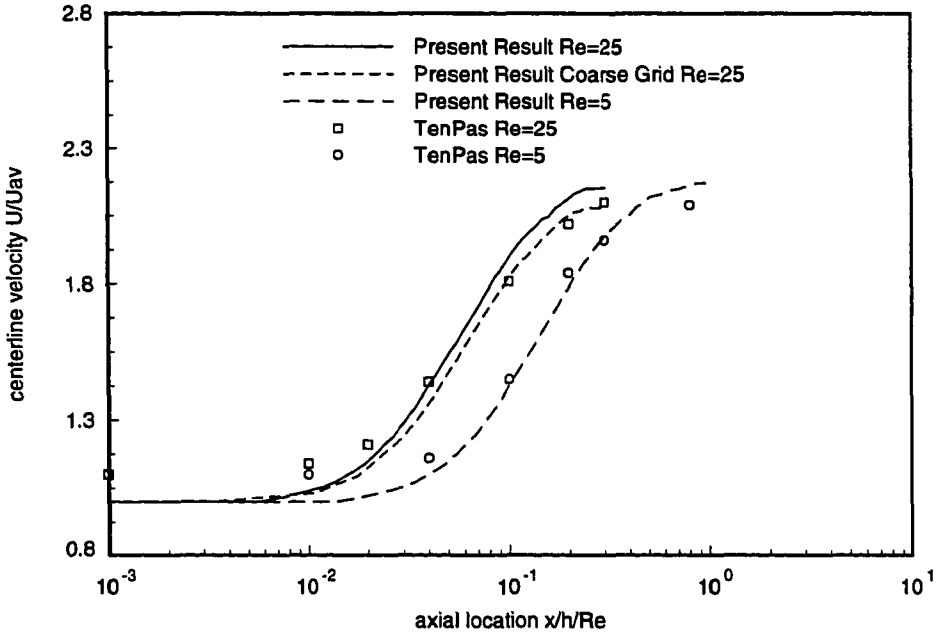

Figure 5.20: Centerline velocity for developing channel flow

correction was based on the fact that a constant mass flow rate is maintained in the duct.

Another three-dimensional testcase was the three-dimensional flow in a curved duct. The channel had a quadratic cross section and inner to outer radius ratio of 1.8:2.8. The duct curvature ratio was $R/a = 1.8$ (see Figure 5.21). The study of viscous flows in curved ducts poses a problem of fundamental interest in the field of internal fluid mechanics. It is a model for understanding some of the important phenomena occuring occuring in flows in turbomachinery applications. Examples of such flows are the flow through blade passages, engine inlets, diffusers and so on. Many of these problems have strong curvature in their geometry. The presence of longitudinal curvature generates centrifugal forces which act at right angles to the

main flow direction and distort the flow. Characteristic parameters of the flow are the Dean number (K) and the curvature ratio R/a :

$$DeanNumber = K = Re(\frac{a}{R})^{0.5}$$

where Re is is the Reynolds number based on inlet conditions and the length a.

The number of cells used to simulate the flow was 118000. The inlet velocity profile was uniform. Only the upper half of the curved duct was computed with symmetry boundary conditions imposed at he midplane. The results of the calculations are shown in Figures 5.28 through 5.30. The computed velocity vector profile in the plane of symmetry is shown in Figure 5.24. As can be seen from the velocity vectors, the velocity profile entering the channel is uniform and the flow becomes increasingly asymmetric as it progresses downstream. The secondary velocity vector fields at different locations along the curved channel are shown in Figures 5.25, 5.26 and 5.27. The secondary flow will never die out in a curved channel since the flow is continuously accelerated around the bend. Figures 5.28 and 5.29 show the fully developed velocity profile in the horizontal and vertical midsurfaces, respectively.

The fully developed velocity profile is compared to the results obtained by Ghia and Sokhey [44]. The developing flow in the present analysis was calculated at a Dean number of 100. Ghia and Sokhey [44] also calculated the fully developed velocity profile in both midplanes of the channel at a Dean number of 100. They found the fully developed velocity profiles to be relatively independent of the curvature ratio. The lowest curvature ratio of the flow geometry for which Ghia and Sokhey present results is 3. The basis for using the results of Ghia and Sokhey [44] for comparison is that their results also show that for a fixed Dean number, the curvature ratio has

very little effect on the fully developed velocity profile. The velocity profiles calculated in the present study differ somewhat from the results of Ghia and Sokhey [44]. Presumably the differences in the calculated results are caused by the differences in the governing equations used in the present study and by Ghia and Sokhey. Ghia and Sokhey used the parabolized Navier-Stokes equations in their calculations. The effect of streamwise diffusion is not considered in the parabolized Navier-Stokes equations. As a consequence of the parabolic flow approximation, the streamwise and transverse pressure gradients were decoupled. The method used by Ghia and Sokhey [44] was set up to advance the solution to a downstream location. The calculation was performed in a once-through manner in the streamwise direction. The streamwise pressure gradient appearing in the governing equations was determined so as to satisfy the conservation of mass-flow rate across the duct cross section. The cross stream pressure variation at the new downstream location is resolved separately.

Besides using a different method to model the flow, the flow geometry in the present study has a curvature ratio of 1.8. The development of the axial velocity profile along the curved channel midplanes is shown in Figure 5.30. No data exists in the literature, to the knowledge of the author, with which the results in Figure 5.30 could be compared. The results show that even near the channel inlet, centrifugal forces are of significant strength compared to the viscous forces so that asymmetry is introduced into the flow. As the flow progresses downstream, the centrifugal forces become more significant and the peak values of the axial velocity shift towards the outer wall of the curved channel.

The three-dimensional laminar driven cavity flow at a Reynolds number of 100 was also computed. The flow geometry is sufficiently simple so that an unstructured

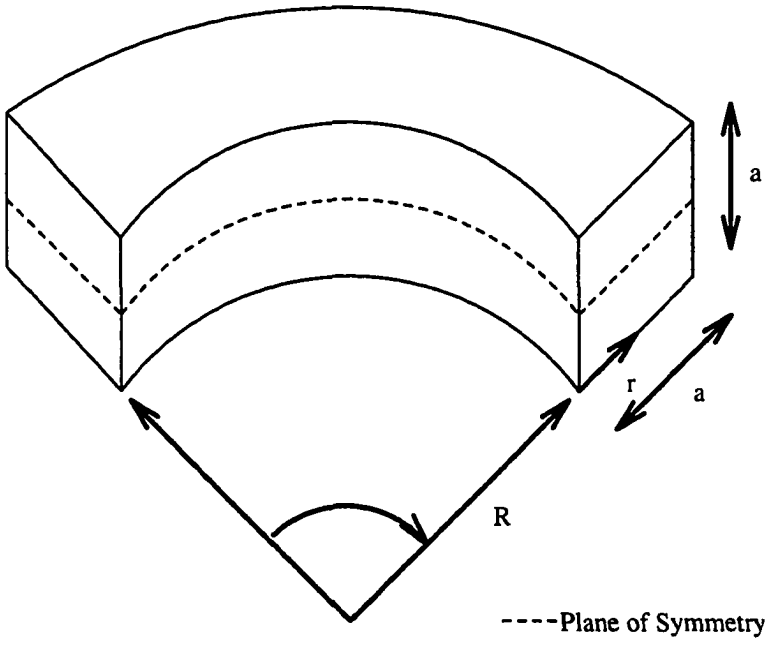

Figure 5.21: Geometry for developing curved channel flow

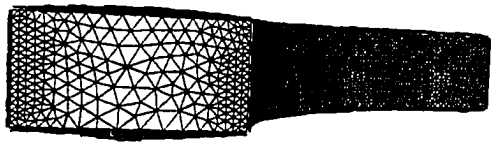

Figure 5.22: View 1: Surface grid for curved channel flow

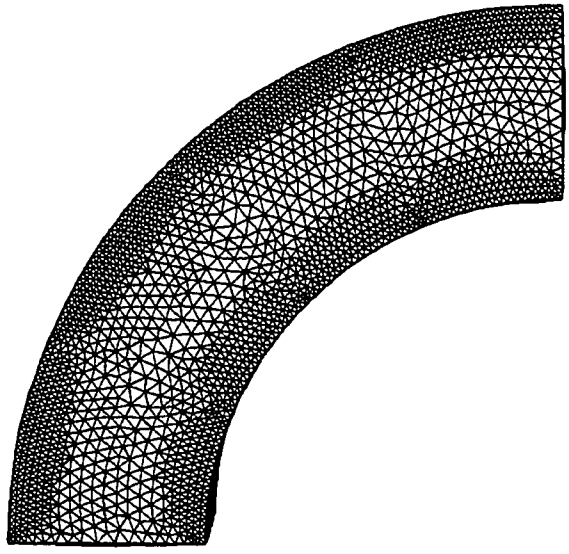

Figure 5.23: View 2: Surface grid for curved channel flow

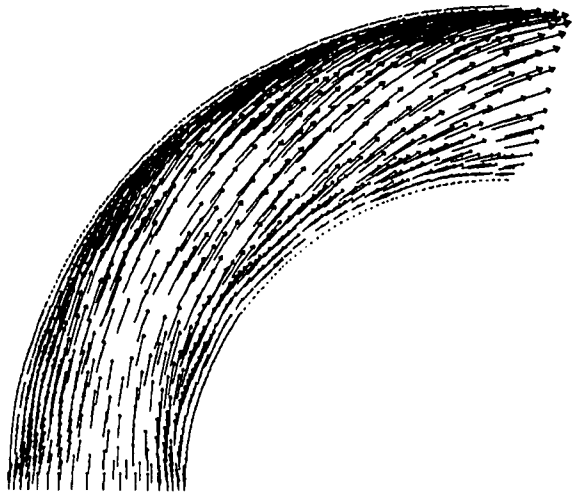

Figure 5.24: Velocity vector field at midplane of curved channel

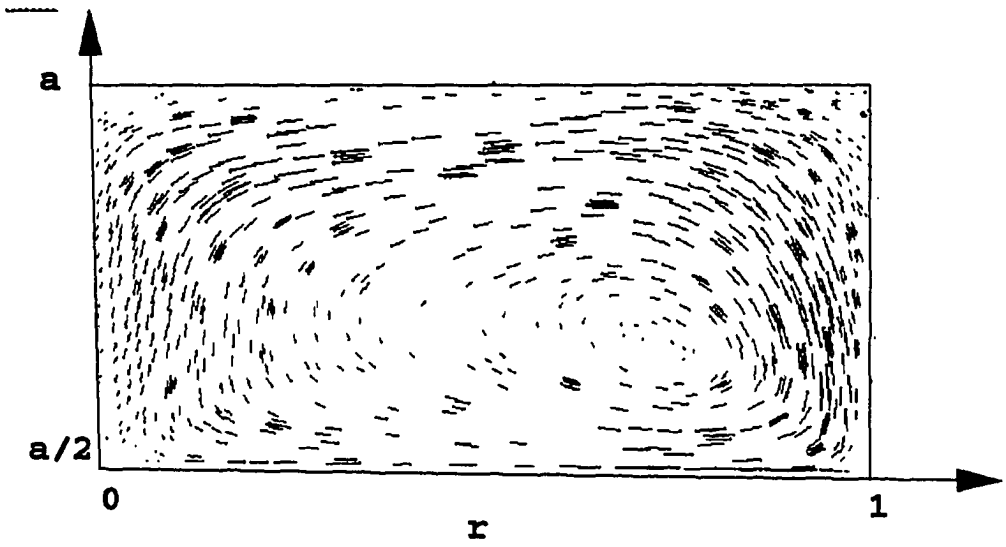

Figure 5.25: Secondary flow velocity vector field at 45 deg.

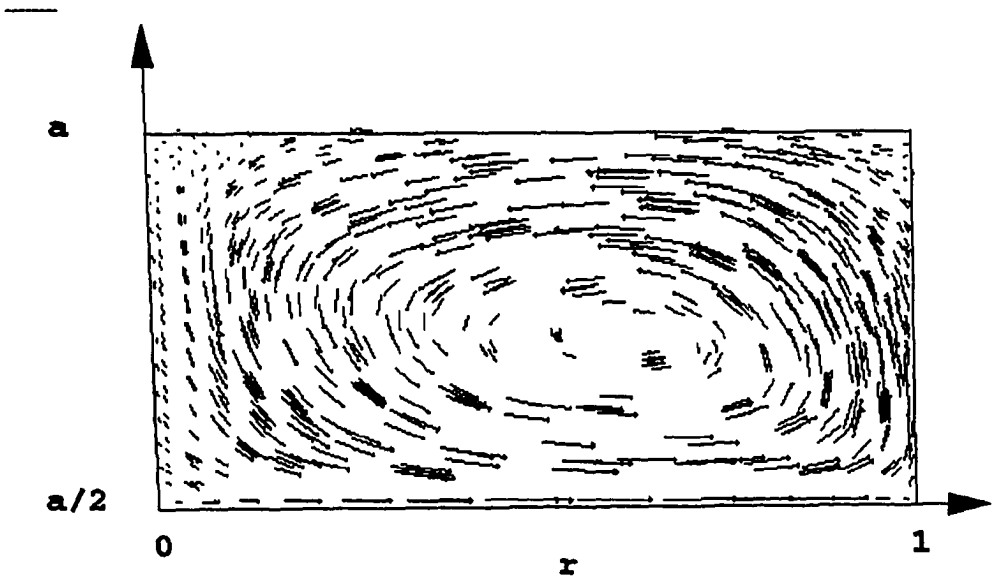

Figure 5.26: Secondary flow velocity vector field near at 67.5 deg

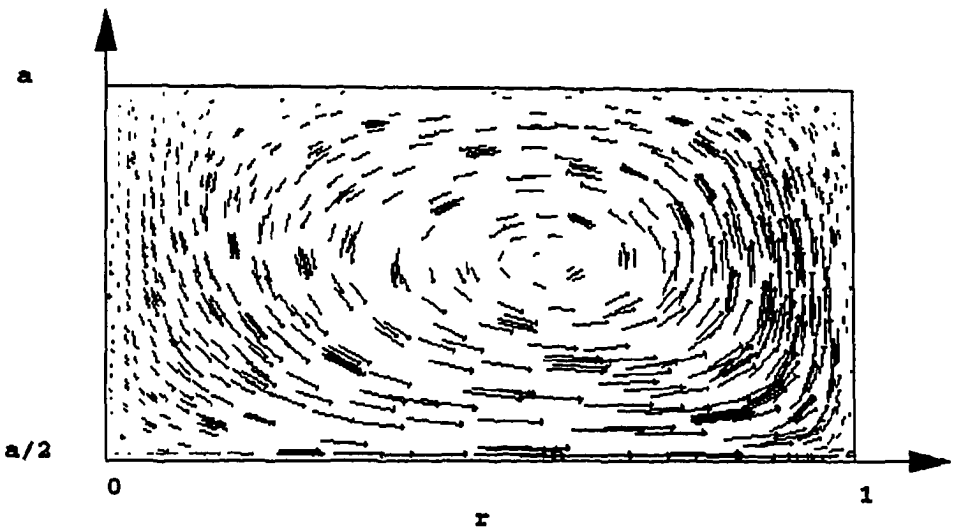

Figure 5.27: Secondary flow velocity vector field near near the exit

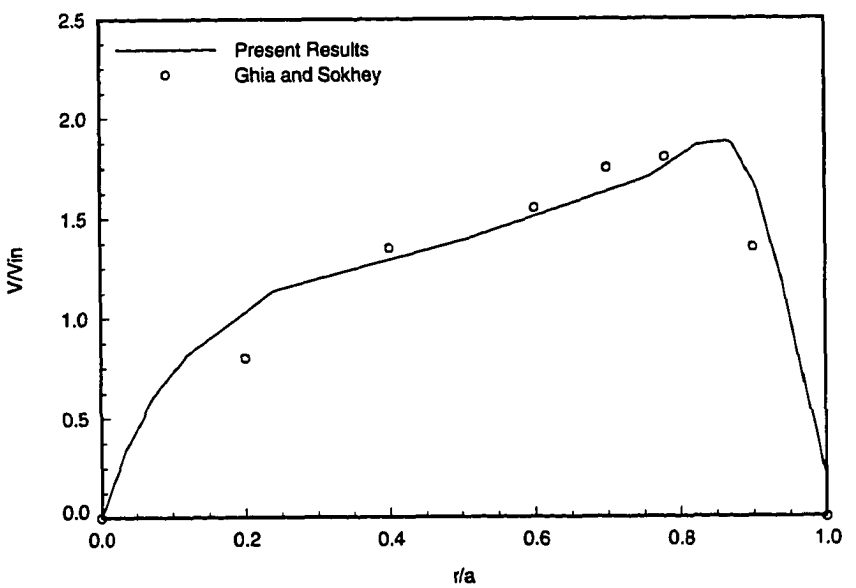

Figure 5.28: Velocity profile at the plane of symmetry of curved channel, K=100

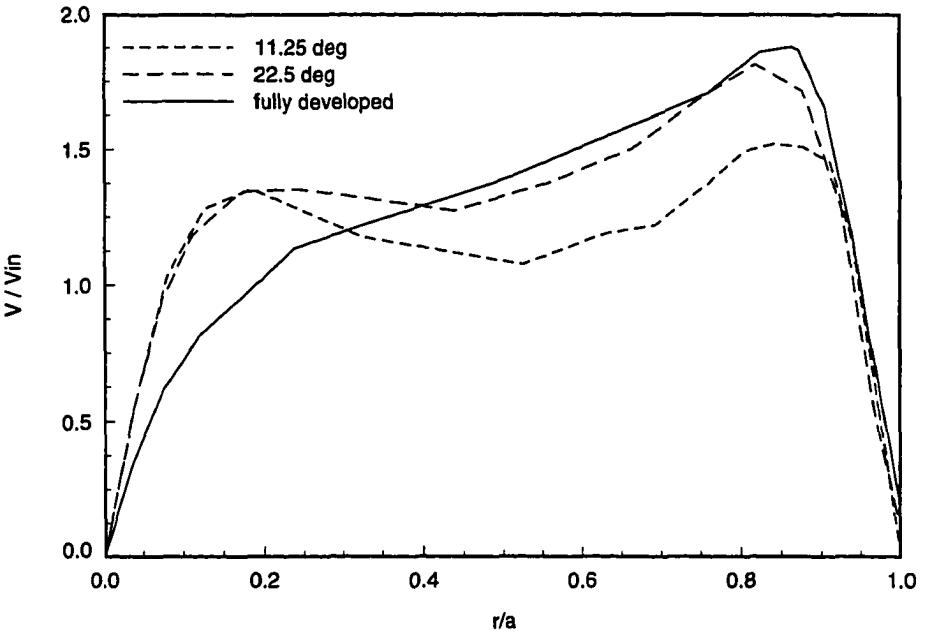

Figure 5.29: A: velocity profiles at the mid-radius plane of curved channel, K=100

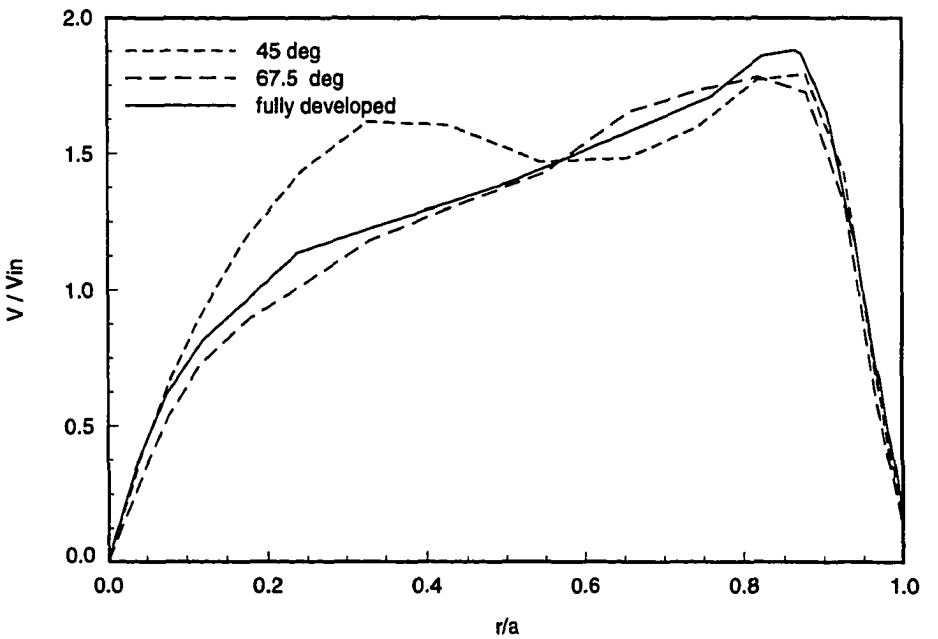

Figure 5.30: B: velocity profiles at the midradius plane of curved channel, K=100

grid was not expected to show significant advantages over a structured grid. The flow was analyzed using unstructured grids since the results could be compared to the work of other investigators, further demonstrating the versatility of the approach used. The flow was analyzed at a Mach number of 0.05, which seems very low for a compressible formulation. Such a low Mach number was chosen to simulate a nearly incompressible flow. Nevertheless, convergence was obtained in 5500 time steps. No time-derivative preconditioning was applied to three-dimensional flows. The surface grid used for the calculation is shown in Figure 5.31. The flow was analyzed using 64500 cells. Some clustering of tetrahedra towards the walls and corners was used. The data representing the centerline velocity of the driven cavity served as a basis for comparison to the work of other researchers. The computed centerline velocity profile is shown in Figure 5.33 along with the results of Chen and Shuen [45] and Rosenfeld et al. [46]. The results obtained in the present study are in good agreement with the work of the other researchers.

Three-dimensional flows were also computed using parallel computers. The limit of today's conventional and and vector computers in terms of memory and reasonable problem sizes is quickly reached even by moderate problem sizes. The computational work required to solve a problem of a certain size is larger in three dimensions than in two dimensions. Thus, as discussed above, the grain size of a parallelized three-dimensional algorithm is inherently larger than in two-dimensions. A larger grain size is expected to allow for larger parallel speedups.

The three-dimensional version of the implicit upwind scheme described in the previous chapters was implemented on the Ncube and the LACE workstation cluster at NASA Lewis Research Center in Cleveland. The LACE cluster consists of 32 IBM

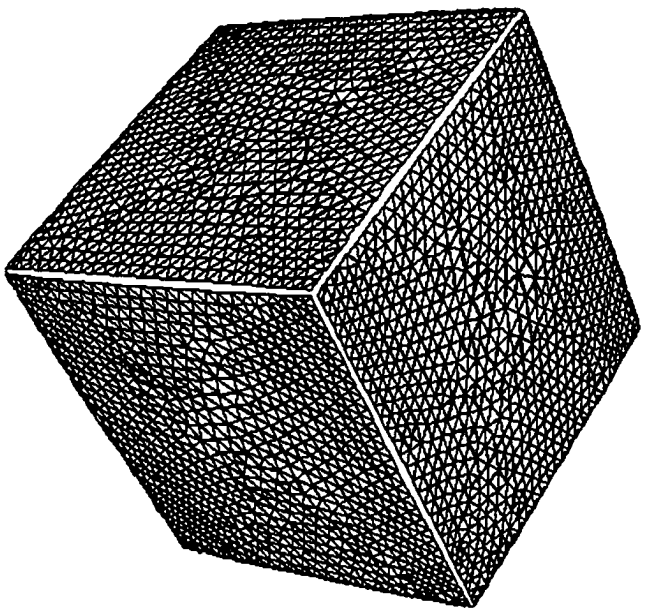

Figure 5.31: Driven cavity flow: surface grid

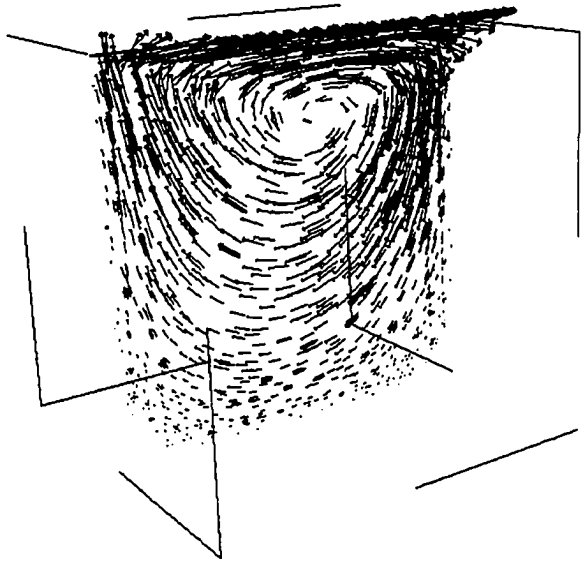

Figure 5.32: Driven cavity flow: velocity vectors in midplane

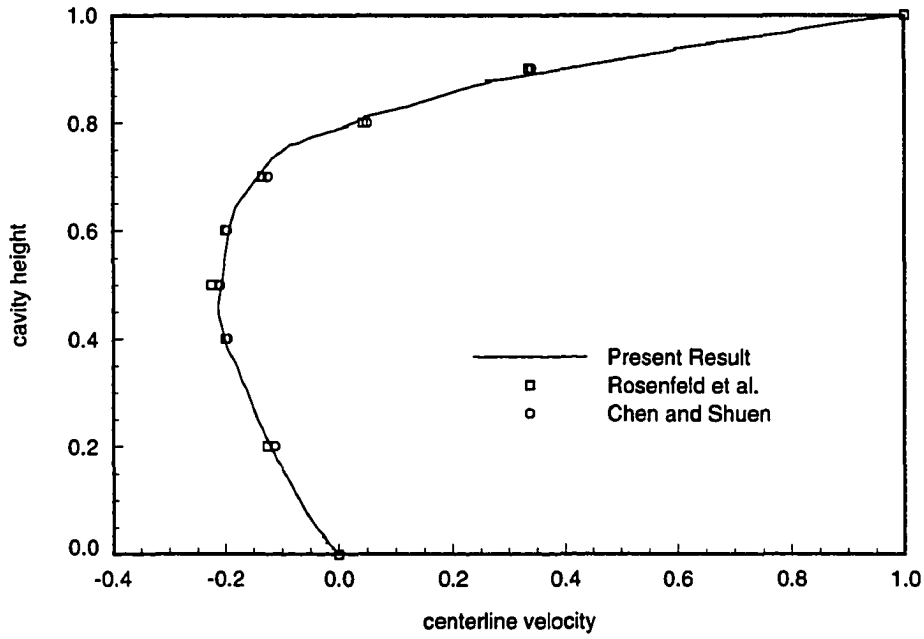

Figure 5.33: Driven cavity flow: centerline velocity profile

RS6000 workstations.

The processors on the Ncube have very limited memory and this resulted in problems in allocating each processor a three-dimensional grid which would correspond to a reasonable grain size. In some cases, instances occurred where the phantom cell of a grid allocated to a particular processor was not allocated to its directly neighboring domain. This of course complicates the message passing logic and the message passing efficiency. On the Ncube size problems were also encountered with the number of messages that needed to be passed among the processors. The design of the message passing logic had to be such that the messages were read by the receiving processor as soon as possible after being sent. This resulted in more and shorter messages than necessary. As is discussed in the previous chapter, each message that is sent

represents some overhead to the overall algorithm.

As opposed to the Ncube, all the memory in the workstations on the LACE cluster is available for parallel computing applications such as message passing. The type of limitations due to available processor memory were not encountered on the LACE cluster. The messages could be packed and all the message passing could be performed in a single message exchange step in the algorithm. Also, the parallelism of the algorithm could be coarsened such that the channel case with 29260 cells could be analyzed on 1 processor and the curved channel case using 118000 cells could be computed using 2 processors.

The result of the timings of the calculations are shown in Figure 5.34 for the smaller grid (29260 cells). The result of the Ncube timing is also shown in the figure for completeness. For all cases analyzed, the workstation cluster gave considerable speedups with an increasing number of processors. For comparison, the trivially parallel case is also shown. Trivial parallelism implies complete independence of the processors with no communication overhead. This is the theoretical ideal case. The figure shows that with an increasing number of processors the performance of the algorithm diverted slightly from the ideal case scenario. Naturally, this is due to the increased overhead incurred from distributing smaller parts of the computational domain to more processors, while the communication overhead per processor is essentially constant. This effect is also illustrated in Table 5.2 where an increase in efficiency is observed with an increasing number of processors while the problem size was kept constant. Also shown in Table 5.2 are the timing results for the larger problem analyzed (118000 cells). As can be seen, the efficiency was generally higher for the larger problem size with the number of processors held constant. This is an

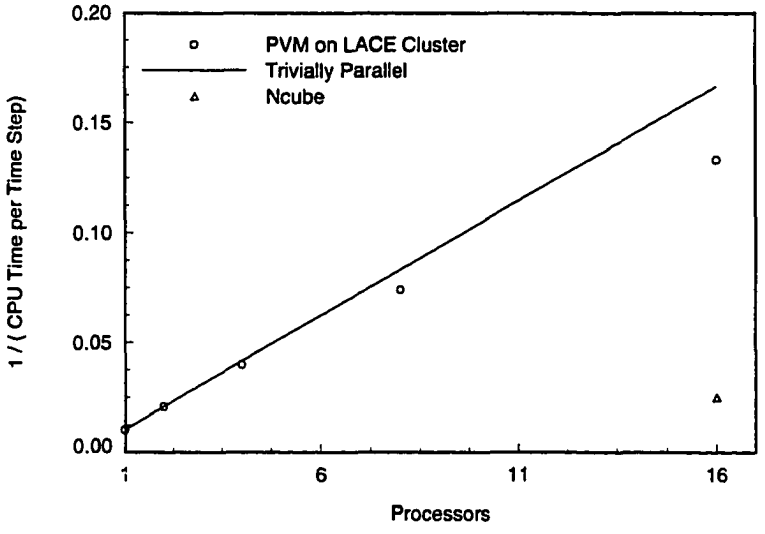

Figure 5.34: Cluster performance as a function of processor count

illustration of how a higher granularity parallelism increases the efficiency. When two processors were used, the efficiency for the larger grid case was less than for the smaller grid case. This is due to memory cache effects, showing that the problem is very large to be run on only two processors.

Based on the results shown in Figure 5.2 it can be said that for the small and the large grid the use of at least 16 processors would result in a speedup over the CRAY-YMP. Also shown in Table 5.2 are the timing results for the Ncube. The memory and buffer size limitation only allowed the small grid to be run on 16 processors. The computational speeds obtained from the Ncube are considerable slower than for the workstation cluster. This is partly a result of the inefficient message passing procedure that had to be used to manage the communication requirements of the problem. This point is discussed above.

Table 5.2: Performance of the unstructured grid code on LACE cluster

	Small Grid		Large Grid		Ncube (small grid)
Procs	steps/sec	effic.	steps/sec	effic.	steps/sec
1	0.010	100	-	-	-
2	0.021	99	0.0050	97	-
4	0.040	96	0.0103	99	-
8	0.074	89	0.0206	99	-
16	0.133	80	0.0412	99	0.025
32					0.045
CRAY-YMP	0.11	-	0.026	-	-

6. SUMMARY AND CONCLUSIONS

The main focus of this research was the development of a three-dimensional Navier-Stokes solver and its implementation on parallel computers. Good potential speedups were obtained. It was found that the three-dimensional Navier-Stokes equations have very large memory requirements if an unstructured grid is used to discretize the domain. The grain size required for solving the conservation equations for fluid flow on a massively parallel computer like the Ncube is too small for three-dimensional flows, but poses no efficiency problems in two-dimensional problems.

The research also made contributions to time-derivative preconditioning methods in computational fluid dynamics. It was demonstrated that time-derivative preconditioning can be applied to an unstructured Navier-Stokes solver using primitive as well as conserved variables using a central-difference scheme. The time-derivative preconditioning was also applied to the AUSM scheme, which is based on a flux-splitting approach.

Two- and three-dimensional Navier-Stokes algorithms have been described in the previous chapters. Testcases were comprised of two- and three-dimensional flows. The compressible Navier-Stokes equations were solved implicitly and explicitly using a block Gauss-Seidel solver and a multistage Runge-Kutta scheme. The procedures applied were verified and investigated by application to various flows. In two dimen-

sions, the flows analyzed were a developing channel flow, a driven cavity, a sudden expansion flow, and the flow over an obstacle in a channel. In three dimensions, a developing channel flow, a driven cavity and the developing flow in a curved duct were computed. A tabulation of the testcases analyzed is shown in Table 5.1. Only viscous flows were studied. The flow over an obstacle was analyzed at a very low Mach number (0.0005), so that low Mach number preconditioning could be tested. The unpreconditioned compressible flow scheme would not converge at such a low Mach number. The ability to use low Mach number preconditioning in an algorithm makes a compressible flow solver applicable to incompressible as well as compressible flows.

For some of the two-dimensional flows analyzed, the use of an unstructured grid proved to be a convenient alternative to a conventional structured grid. For example, the structured grid for a sudden expansion flow cannot be generated in a straightforward manner. The three-dimensional flows analyzed could have been analyzed using structured grids without greater difficulties. During the development of the three-dimensional algorithm it became apparent that the need for memory quickly became excessive with an increasing number of cells. Three-dimensional testcases were chosen which could be analyzed with a number of cells small enough not to exceed the capabilities of the available computer resources. The main reason why the number of cells required for a particular testcase appears to become so excessive so fast is that the grid generator (Vgrid3d, version 1.0) used did not allow for the generation of elongated cells. The use of elongated cells in the viscous region of the flowfield would allow for the adherence of the grid to the physics of the flow. In the present study, the cell density was dictated by the largest gradients which does

not necessarily imply an efficient use of control volumes. This was also the case in two dimensions, but memory limitations were not encountered for the two-dimensional testcases analyzed during the course of the study. Since the main thrust of the project was the development of flow solvers, it was intended that existing research tools for grid generation were to be used as much as possible. Two and three-dimensional grid generators were available and were not developed further.

Due to the high computational resource requirements associated with the use of unstructured grids, the future of unstructured grids as a competitive design tool depends on the increase of available computational capability. Since no drastic increase in memory and speed can be expected from the current top of the line supercomputers, parallel or scalable computers are an option to further increase accessible computational speeds. Two- and three-dimensional versions of the algorithm developed were implemented on a massively parallel computer and a workstation cluster. Potential increases in computational speeds over existing supercomputers have been demonstrated.

A two-dimensional unstructured grid Navier-Stokes solver using the cell centered approach has been implemented on the Ncube2s computer. A three-dimensional version of the algorithm has been adapted to utilize the message passing script PVM (Parallel Virtual Machine) on the LACE workstation cluster at NASA Lewis Research Center. The three-dimensional algorithm was also implemented on the Ncube but the memory requirements associated with three-dimensional flow solvers reached the limit of the Ncube's capabilities. For example, the message passing logic had to be designed so that the residence time of messages in the buffer storage space was very short, rapidly making new buffer memory available for new data to be stored. This

prevented the design of an efficient message passing algorithm.

The domains for the calculations were distributed to the processors such that each processor was allocated an equal number of cells. In order to make efficient use of computer memory, a local data structure was used on each processor with no reference to a global data structure. For a particular cell, the setup of the communication network inherent to local data structures involved all adjacent cells allocated to a different processor. In addition, vertex values opposite to a face with cells on different processors on each side were involved in the message passing network. For a second order upwind scheme, this was considered the minimum communication requirement.

The algorithm was used to analyze four two-dimensional testcases. A developing channel flow, a driven cavity, a sudden expansion flow and the flow over on obstacle in a channel were computed. Good agreement with existing test results was found. The results for the testcases were obtained on a DEC 5000 and the Ncube2s using four and eight processors. As expected, the communication overhead increased with the number of processors. For the relatively small number of processors used all runs the on Ncube were faster than on the DEC 5000 workstation. The communication overhead for the computed testcases was about 2.5 % and 5 % when four and eight processors were used, respectively. The same overhead was observed for all testcases since about the same number of control volumes was used in all cases. At the same time, more processors were not readily available to be used for the current study.

Three three-dimensional testcases were calculated. The flows that were computed were a developing straight channel flow, a driven cavity flow and the developing flow in a curved channel.

The developing straight channel flow and the curved channel flow testcases were

analyzed using PVM software on the LACE workstation cluster. The LACE cluster consists of IBM RS6000 workstations. The maximum number of workstations used for the parallel computation was 16. The grain size of the parallelism in the cases analyzed was large enough so that very high efficiencies were observed. Even with 16 processors, an efficiency of eighty percent was found for the small grid case (29260 cells).

Based on the results obtained from the parallel calculations on the Ncube and the workstation cluster, some experience has been gained as to what is the best route to take for larger problems. The conclusion can be drawn that a three-dimensional flow solver is relatively memory intensive so that it is advantageous to utilize the more capable workstations than the less capable processors of a massively parallel computer for parallel calculations. Theoretically both a massively parallel computer and a cluster of workstations can be scaled, but the memory requirements associated with interprocessor communication easily become excessive. The communication in a massively parallel computer is more efficient than in a network of workstations, but the advantages of a significantly larger grain size outweigh those of faster communication. This conclusion is applicable to a parallel computing approach where the grid is partitioned and distributed to the different processors. This strategy is well suited to be used on MIMD machines.

BIBLIOGRAPHY

[1] Mavriplis D., "Solution of the Two-dimensional Euler Equations on Unstructured Triangular Meshes", PhD Dissertation. Massachusttes Institute of Technology: Princeton, NJ, 1987.

[2] Hammond S., Barth T., "An Efficient Massively Parallel Euler Solver for Unstructured Grids," *AIAA-91-0441,* 1994.

[3] Frink N., "A Fast Upwind Solver For The Euler Equations on Three-Dimensional Meshes," *AIAA-91-0102,* 1991.

[4] Barth T.J., "On Unstructured Grids and Solvers," Notes: NASA Ames Research Center, Moffett Field, CA.

[5] Allaire, P., *Basics of the Finite Element Method,* Dubuque, IA: Brown Publishers, 1985.

[6] Hirsch, C., *Numerical Computation of Internal and External Flows* New York, NY: John Wiley & Sons, 1988.

[7] Batina J.T., "Accuracy of an Unstructured-Grid Upwind Euler Algorithm fot the ONERA M6 Wing," Presented at the Accuracy of Unstructured Grid Technique Workshop, NASA Langley, Hampton, VA, Jan. 17-18 , 1990.

[8] Barth T., Frederickson P., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction," *AIAA-90-0013,* 1990.

[9] Holmes D.G., Connel S.D., "Solution of the 2-D Navier-Stokes Equations on unstructured adaptive Grids," *AIAA-89-1932-CP,* 1989.

[10] Barth J. B., Jespersen C.J., "The Design and Application of Upwind Schemes on Unstructured Meshes," *AIAA-89-89-0366,* 1989.

[11] Knight D., "A Fully Implicit Navier-Stokes Algorithm Using an Unstructured Grid and Flux Difference Splitting," *AIAA-93-0875,* 1993.

[12] Pan D., Cheng J. "Upwind Finite-Volume Navier-Stokes Computations on Unstructured Triangular Meshes," *AIAA Journal* , Vol. 31, No. 9, pp. 1618-1625, 1993.

[13] Anderson W.K., Bonhaus D.L., "An implicit upwind algorithm for computing turbulent flows on unstructured grids," *Computers and Fluids*, Vol. 23, No. 1, pp. 1-21, 1994.

[14] Smith W.A., Spragle G.S., "Application of an unstructured grid flow solver to Planes, Trains and Automobiles," Submitted to AIAA Aerospace Sciences Meeting, Jan. 11-14, 1993.

[15] Ramamurti R., Löhner L., "Simulation of of Subsonic Viscous Flow using unstructured Grids and a Finite Element Solver," *AIAA-90-0702,* 1990.

[16] Whitaker D.L., "Three-Dimensional Unstructured Grid Euler Computations," *AIAA-93-3337-CP,* 1993.

[17] Winterstein R., Hafez M., "Euler Solutions for Blunt Bodies using Triangular Meshes: Artificial Viscosity Forms and Numerical Boundary Conditions," *AIAA-93-3333-CP,* 1993.

[18] Kominsky P.J., "Performance Analysis of an Implementation of the Beam Warming Implicit Factored Scheme on the Ncube Hypercube," *Proceedings, 1990 Third Synposium of the Frontiers of Massively Parallel Computation*, pp. 119-126, 1990.

[19] Bruno J., Capello P.R., "Implenting the Beam and Warming Method on the Hypercube," *Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications,* pp. 1073-1087, 1988.

[20] Das R., Mavriplis D.J., Saltz, J., Gupta S., Ponnusamy R., "The Design and Implementation of a Parallel Unstructured Euler Solver Using Software Primitives," *AIAA-92-0562,* 1992.

[21] Hixon D., Sankar L.N., "Unsteady Compressible 2-D Flow Calculations on a MIMD Parallel Supercompter," *AIAA-94-0757,* 1994.

[22] Venkatakrishnan V., Simon H.D., Barth T.J., "A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids," Report RNR-91-024, September 1991.

[23] Cuthill E., McKee J., "Reducing the bandwidth of sparse symmetric matrices," *Proceedings, 24th National Conference of the Association of Computing Machinery,* pp. 157-172, 1969.

[24] Pothen A., Simon H.D., and Liou K.P., "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Applied Mathematical Analysis,* Vol. 11, pp. 430-452, 1990.

[25] Cui A., Knight D., "Implementation of an Unstructured Navier-Stokes Algorithm on the Connection Machine," *AIAA-94-0411,* 1994.

[26] Chen K.H., A primitive variable, strongly implicit calculation procedure for two and three-dimensional unsteady viscous flows: applications to compressible and incompressible flows including flows with free surfaces, PhD Dissertation. Iowa State University: Ames, IA, 1990.

[27] Lewis T., El-Rewini H., *Introduction to Parallel Computing,* Englewood Cliffs, NJ: Prentice Hall, 1991.

[28] Pletcher R.H., "Compressibles N-S Equations at Low Speeds," *Personal Notes.*

[29] Lindquist D., A Comparison of Numerical Schemes on Triangular and Quadrilateral Meshes, M.S. Thesis. Massachussetes Institute of Technology, Princeton, NJ, 1988.

[30] Patankar S.V., *Numerical Heat Transfer and Fluid Flow,* New York, NY: Hemisphere Publishing Co, 1980.

[31] Venkatakrishnan V., Mavriplis D., "Implicit Solvers for Unstructured Meshes," *ICASE Report No. 91-40* , 1991.

[32] Jameson A., Schmidt W., and Turkel E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stepping Schemes," *AIAA-81-1259,* 1981.

[33] Jorgenson P., An Implicit Numerical Scheme for the Simulation of Internal Viscous Flows on Unstructured Grids, PhD Dissertation. Iowa State University: Ames, IA, 1992.

[34] Choi Y., Merkle C.L., "Time-Derivative Preconditioniong of Viscous Flows," *AIAA-91-1652,* 1991.

[35] Struijs R., Deconick H., de Palma P., Roe P., and Powell K.G., "Progress on Multidimensional Upwind Euler Solvers for Unstructured Grids," *AIAA-91-1550,* 1991.

[36] Roe P.L., "Error Estimates for Cell-Vertex Solutions of the Compressible Euler Equations," *ICASE Report No. 87-6,* 1987.

[37] Batina J.T., "A Fast Implicit Upwind Solution Algorithm For Three-Dimensional Dynamic Meshes," *AIAA-92-0447,* 1992.

[38] Anderson D.A., Tannehill J.C., Pletcher R.H., *Computational Fluid Dynamics and Heat Transfer,* New York, NY: Hemisphere Publishing Co., 1984.

[39] TenPas P.W., "Numerical solution of the steady, compressible, Navier-Stokes equations intwo and three dimensions by a coupled space marching method," PhD Dissertation. Iowa State University: Ames, IA, 1990.

[40] Gegg S.G., Pletcher R.H, Steger J.L., "A Dual Potential Formulation of the Navier-Stokes Equations," Ames, IA: ISU-ERI-Ames-90030.

[41] White F.M., *Viscous Fluid Flow,* New York, NY: Mc Graw-Hill Book Co., 1974.

[42] Simon H.D., "Partitioning of Unstructured Problems For Parallel Processing," *Computing Systems in Engineering,* Vol. 2, No. 2/3, pp. 135-148, 1991.

[43] Nallasamy N., "Numerical Solution of the separating Flow due to an Obstruction," *Computers and Fluids,* Vol. 14, No. 1, pp. 59-68, 1990.

[44] Ghia K.N., Sokhey J.S., "Laminar Incompressible Viscous Flow in Curved Ducts of Rectangular Cross-Sections," Journal of Fluids Engineering, Vol. 99, pp. 640-648, 1977.

[45] Chen K.H., Shuen J., "Three-Dimensional Coupled Implicit Methods for Spray Combustion Flows at All Speeds," *AIAA-94-3047,* 1994.

[46] Rosenfeld M., Kwak D. and Vinokur M., "A Solution Method for the Unsteady and Incompressible Navier-Stokes Equations in Generalized Coordinate Systems," *AIAA-88-0718,* 1988.

[47] Liou M.S., Steffen C.J., "A new flux splitting scheme," *NASA Technical Memorandum 104404* , 1990.

[48] Burggraf O.R., "Analytic and Numerical Studies of the Structure of Steady Separated Flows," *Journal of Fluid Mechanics,* Part 1, 24, pp. 113-151, 1966.

[49] Durst F., Melling A., Whitelaw J.H., "Low Reynolds number flow over a plane symmetric sudden expansion," *Journal of Fluid Mechanics,* Vol. 64, pp. 441-428, 1974.

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

[51] Pletcher R.H., Chen K.H., "On solving the compressible Navier-Stokes equations for unsteady flows at very low Mach numbers, " *AIAA-93-3368,* 1993.

[52] Vgrid3d User's Manual, Version 1.0, VIGYAN Corporation, Hampton, VA, 1991.

APPENDIX A. ROE's FLUX DIFFERENCE SPLITTING

An overview of the procedure involved in implementing Roe's flux difference splitting is shown below [6]. Upwind schemes apply a discretization that depends on the sign of propagation direction of the wave, which is given by the sign of the eigenvalues. General idea of upwinding:

Consider face shared by cell i and i+1:

$$F^*_{i+\frac{1}{2}} = (F_i^+ + F_{i+1}^-) = \frac{1}{2}[(F_i + F_{i+1}) - |\delta f|]$$

$$\delta f = |[A]|(U_{i+1} - U_i)$$

In two dimensions the flux Jacobian A is:

$$A = \frac{\partial F}{\partial U} = \eta x \frac{\partial f}{\partial U} + \eta y \frac{\partial g}{\partial U}$$

which can also be expressed as:

$$A = P\Lambda P^{-1} q$$

the matrices Λ, P and P^{-1} are given below.

The eigenvalues of the Jacobian [A] are:

$$\Lambda = \begin{pmatrix} V & 0 & 0 & 0 & 0 \\ 0 & V & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & V+c & 0 \\ 0 & 0 & 0 & 0 & V-c \end{pmatrix}$$

where $V = \eta_x u + \eta_y v + \eta_z w$

The right eigenvectors are columns of the matrix P , while the left eigenvectors are the rows of matrix P^{-1} :

$$P = \begin{pmatrix} \eta_x & \eta_y & \eta_z & \frac{\rho}{\sqrt{2}c} & \frac{\rho}{\sqrt{2}c} \\ \eta_x u & \eta_y u - \eta_z \rho & \eta_z u + \eta_y \rho & \frac{\rho}{\sqrt{2}c}(v + \eta_y c) & \frac{\rho}{\sqrt{2}c}(v + \eta_y c) \\ \eta_x v + \rho \eta_z & \eta_y v & \eta_z v - \eta_x \rho & \frac{1}{\sqrt{2}c}(v + \eta_y c) & \frac{1}{\sqrt{2}c}(v - \eta_y c) \\ \eta_x w - \rho \eta_y & \eta_y w + \eta_x \rho & \eta_z w & \eta_z w & \frac{1}{\sqrt{2}c}(w - \eta_z c) \\ \eta_x \frac{u^2+v^2+w^2}{2} + \rho(\eta_z v - \eta_y w) & \eta_y \frac{u^2+v^2+w^2}{2} + \rho(\eta_x w - \eta_z u) & \eta_z \frac{u^2+v^2+w^2}{2} + \rho(\eta_y u - \eta_x v) & \frac{\rho}{\sqrt{2}c}(H + cV) & \frac{\rho}{\sqrt{2}c}(H - cV) \end{pmatrix}$$

$$P^{-1} = \begin{pmatrix} \eta_x(1-M^2)+\frac{1}{\rho}(\eta_y w-\eta_z v) & \eta_x\frac{u(\gamma-1)}{c^2} \\ \eta_y(1-M^2)+\frac{1}{\rho}(\eta_z u-\eta_x w) & -\eta_z\frac{1}{\rho}+\eta_y\frac{u(\gamma-1)}{c^2} \\ \eta_z(1-M^2)+\frac{1}{\rho}(\eta_x v-\eta_y u) & \eta_y\frac{1}{\rho}+\eta_z\frac{u(\gamma-1)}{c^2} \\ \frac{1}{\sqrt{2}\rho c}(\Phi-cV) & \frac{1}{\sqrt{2}\rho c}(\eta_x c-u(\gamma-1)) \\ \frac{1}{\sqrt{2}\rho c}(\Phi+cV) & \frac{1}{\sqrt{2}\rho c}(\eta_x c-u(\gamma-1)) \end{pmatrix}$$

$$\begin{pmatrix} \eta_x\frac{u(\gamma-1)}{c^2} & -\eta_y\frac{1}{\rho}+\eta_x\frac{w(\gamma-1)}{c^2} & -\eta_x\frac{\gamma-1}{c^2} \\ \eta_y\frac{v(\gamma-1)}{c^2} & -\eta_x\frac{1}{\rho}+\eta_y\frac{w(\gamma-1)}{c^2} & -\eta_y\frac{\gamma-1}{c^2} \\ -\eta_x\frac{1}{\rho}+\eta_z\frac{v(\gamma-1)}{c^2} & \eta_z\frac{w(\gamma-1)}{c^2} & -\eta_z\frac{\gamma-1}{c^2} \\ \frac{1}{\sqrt{2}\rho c}(\eta_y c-u(\gamma-1)) & \frac{1}{\sqrt{2}c}(\eta_z c-w(\gamma-1)) & \frac{1}{\sqrt{2}c}(\gamma-1) \\ -\frac{1}{\sqrt{2}\rho c}(\eta_y c+v(\gamma-1)) & -\frac{\rho}{\sqrt{2}c}(\eta_z c-w(\gamma-1)) & \frac{\rho}{\sqrt{2}c}(\gamma-1) \end{pmatrix}$$

where

$$\begin{aligned} \Phi &= \frac{\gamma-1}{2}(u^2+v^2+w^2) \\ H &= \frac{\gamma p}{\rho(\gamma-1)}+\frac{u^2+v^2+w^2}{2} \end{aligned}$$

The characteristic variables are associated with the characteristic form of the Euler equations:

$$\frac{\partial W}{\partial t}+\Lambda\frac{\partial W}{\partial x}=0$$

where W is:

$$\delta W = \begin{pmatrix} \delta w_1 \\ \delta w_2 \\ \delta w_3 \\ \delta w_4 \\ \delta w_5 \end{pmatrix} = \begin{pmatrix} \eta x(\delta\rho - \frac{\delta\rho}{\tilde{c}^2}) + \eta z v - \eta y w \\ \eta y(\delta\rho - \frac{\delta\rho}{\tilde{c}^2}) - \eta z u + \eta x w \\ \eta z(\delta\rho - \frac{\delta\rho}{\tilde{c}^2}) + \eta y u - \eta x v \\ \frac{1}{\sqrt{2}}(\frac{\delta p}{\tilde{\rho}\tilde{c}} + \eta x \delta u + \eta x \delta v + \eta z w) \\ \frac{1}{\sqrt{2}}(\frac{\delta p}{\tilde{\rho}\tilde{c}} - \eta x \delta u - \eta x \delta v - \eta z w) \end{pmatrix}$$

The conserved variables U can be related to the characteristic variables W by

$$\delta U = P \delta W$$

which also be expressed as:

$$\delta U = \sum \lambda_j \delta w_j r^j$$

w = characteristic variable r= right eigenvector which expresses the variations of the conservative variables U as a sum of waves r^j with amplitudes δw_j.

For a linear system, one can write flux variations as:

$$\delta f = A \delta U = A \sum_j \delta w_j r^j = \sum \lambda_j \delta w_j r^j$$

$$\delta |f| = |A| \delta U = \sum |\lambda_j| \delta w_j r^j$$

Roe's Flux Difference Splitting: The idea of Roe's flux difference splitting is to extend the linear wave decomposition to non-linear problems. For any U_i, U_{i+1}, associated with a face of a control volume, a matrix $\tilde{A}$ is desired which has the following properties:

$f_{i+1} - f_i = A(U_i, U_{i+1})(U_{i+1} - U_i)$

$if \quad U_i = U_{i+1} = U \quad then \quad A(U,U) = \frac{\partial f}{\partial U}$

$A(U_i, U_{i+1})$ *has real eigenvalues with linearly independent eigenvectors*

The procedure for accomplishing this can be shown as follows:

The variable Z is introduced:

$$Z = \sqrt{\rho} \begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix}$$

then

$$U_{i+1} - U_i = [\tilde{B}](Z_{i+1} - Z_i)$$

$$f_{i+1} - f_i = [\tilde{C}](Z_{i+1} - Z_i)$$

It turns out that

$$\tilde{A} = \tilde{C}\tilde{B}^{-1}$$

if expressed in terms of the following variables (Roe variables):

$$\tilde{\rho}_{i+\frac{1}{2}} = \sqrt{\rho_{i+1}\,\rho_i}$$

$$\tilde{u}_{i+\frac{1}{2}} = \frac{(u\sqrt{\rho})_{i+1} + (u\sqrt{\rho})_i}{\sqrt{\rho_{i+1}} + \sqrt{\rho_i}}$$

$$\tilde{v}_{i+\frac{1}{2}} = \frac{(v\sqrt{\rho})_{i+1} + (v\sqrt{\rho})_i}{\sqrt{\rho_{i+1}} + \sqrt{\rho_i}}$$

$$\tilde{H}_{i+\frac{1}{2}} = \frac{(H\sqrt{\rho})_{i+1} + (H\sqrt{\rho})_i}{\sqrt{\rho_{i+1}} + \sqrt{\rho_i}}$$

Roe's flux differencing is implemented by calculating the speed of sound and the right eigenvectors in terms of the averaged variables. The wave amplitudes are also evaluated at the cell face with $\delta u, \delta p$, and $\delta \rho$ evaluated as differences across the cell face (e.g. $\delta u = u_{i+1} - u_i$).

APPENDIX B. INVERSE OF CONSERVATIVE PPRECONDITIONING MATRIX

The entries of the inverse of the conservative preconditioning matrix presented in Chapter 3 are shown below:

$$
\begin{aligned}
\bar{\Gamma}_{11}^{-1} &= \frac{-2\rho\Phi(\Phi_0(\gamma-1)+\gamma)-(u^2+v^2)(\gamma-1)}{(\gamma-1)\Phi_1} \\
\bar{\Gamma}_{12}^{-1} &= \frac{-2\rho u}{\Phi_1} \\
\bar{\Gamma}_{13}^{-1} &= \frac{-2\rho v}{\Phi_1} \\
\bar{\Gamma}_{14}^{-1} &= \frac{2\rho}{\Phi_1} \\
\bar{\Gamma}_{21}^{-1} &= \frac{-u(u^2+v^2)(\gamma-1)+(2\rho\Phi_1(\Phi_2(\gamma-1)+\gamma)+2(\gamma-1)\gamma\rho E)}{(\gamma-1)\Phi_1} \\
\bar{\Gamma}_{22}^{-1} &= \frac{\rho u(\gamma-2)+\rho v\gamma-2\gamma\rho E}{\Phi_1} \\
\bar{\Gamma}_{23}^{-1} &= \frac{-2\rho uv}{\Phi_1} \\
\bar{\Gamma}_{24}^{-1} &= \frac{2\rho u}{\Phi_1} \\
\bar{\Gamma}_{31}^{-1} &= \frac{-v(u^2+v^2)(\gamma-1)+(2\rho\Phi_(\Phi_2(\gamma-1)+\gamma)+2(\gamma-1)\gamma\rho E)}{(\gamma-1)\Phi_1} \\
\bar{\Gamma}_{32}^{-1} &= \frac{-2\rho uv}{\Phi_1}
\end{aligned}
$$

$$
\begin{aligned}
\bar{\Gamma}_{33}^{-1} &= \frac{\rho v(\gamma-2)+\rho u\gamma-2\gamma\rho E}{\Phi_1} \\
\bar{\Gamma}_{34}^{-1} &= \frac{2\rho v}{\Phi_1} \\
\bar{\Gamma}_{41}^{-1} &= \frac{\rho(\gamma-1)^2(u^4+v^4)+(u^2+v^2)(\gamma-1)(\rho\Phi_0\Phi_1-2\gamma E)}{(\gamma-1)\Phi_2} \\
&\quad +\frac{\rho(\gamma-1)^2u^2v^2+2\Phi_0\gamma E}{(\gamma-1)\Phi_2} \\
\bar{\Gamma}_{42}^{-1} &= \frac{u(\rho(\gamma-1)(u^2+v^2)-2\gamma E)}{\Phi_2} \\
\bar{\Gamma}_{43}^{-1} &= \frac{v(\rho(\gamma-1)(u^2+v^2)-2\gamma E)}{\Phi_2} \\
\bar{\Gamma}_{44}^{-1} &= \frac{\rho(u^2+v^2)}{\Phi_2}
\end{aligned}
$$

where

$$\Phi_0 = \beta M^2$$

$$\Phi_1 = \frac{\rho H}{\rho\Phi_0} - \delta$$

$$\Phi_2 = \gamma((u^2+v^2)-2\rho E)$$