Durham E-Theses

Computational Aerodynamics on unstructed meshes

Zheng, Yun

How to cite:

Use policy

COMPUTATIONAL AERODYNAMICS ON UNSTRUCTURED MESHES

by
Yun Zheng

A thesis submitted to the University of Durham
for the degree of Doctor of Philosophy

School of Engineering
University of Durham
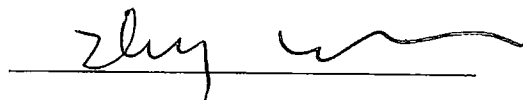Science Site, South Road
Durham DH1 3LE

2 8 APR 2004

February 2004

Declaration

I declare that no material presented in this dissertation has been submitted toward any degree, either at this university or anywhere else.

Yun Zheng

Acknowledgment

ABSTRACT

New 2D and 3D unstructured-grid based flow solvers have been developed for simulating steady compressible flows for aerodynamic applications. The codes employ the full compressible Euler/Navier-Stokes equations. The Spalart-Allmaras one equation turbulence model is used to model turbulence effects of flows. The spatial discretisation has been obtained using a cell-centred finite volume scheme on unstructured-grids, consisting of triangles in 2D and of tetrahedral and prismatic elements in 3D. The temporal discretisation has been obtained with an explicit multistage Runge-Kutta scheme. An "inflation" mesh generation technique is introduced to effectively reduce the difficulty in generating highly stretched 2D/3D viscous grids in regions near solid surfaces. The explicit flow method is accelerated by the use of a multigrid method with consideration of the high grid aspect ratio in viscous flow simulations. A solution mesh adaptation technique is incorporated to improve the overall accuracy of the 2D inviscid and viscous flow solutions. The 3D flow solvers are parallelised in a MIMD fashion aimed at a PC cluster system to reduce the computing time for aerodynamic applications.

The numerical methods are first applied to several 2D inviscid flow cases, including subsonic flow in a bump channel, transonic flow around a NACA0012 airfoil and transonic flow around the RAE 2822 airfoil to validate the numerical algorithms. The rest of the 2D case studies concentrate on viscous flow simulations including laminar/turbulent flow over a flat plate, transonic turbulent flow over the RAE 2822

airfoil, and low speed turbulent flows in a turbine cascade with massive separations. The results are compared to experimental data to assess the accuracy of the method. The over-resolved problem with mesh adaptation on viscous flow simulations is addressed with a two phase mesh reconstruction procedure. The solution convergence rate with the aspect ratio adaptive multigrid method and the direct connectivity based multigrid is assessed in several viscous turbulent flow simulations.

Several 3D test cases are presented to validate the numerical algorithms for solving Euler/Navier-Stokes equations. Inviscid flow around the M6 wing airfoil is simulated on the tetrahedron based 3D flow solver with an upwind scheme and spatial second order finite volume method. The efficiency of the multigrid for inviscid flow simulations is examined. The efficiency of the parallelised 3D flow solver and the PC cluster system is assessed with simulations of the same case with different partitioning schemes. The present parallelised 3D flow solvers on the PC cluster system show satisfactory parallel computing performance. Turbulent flows over a flat plate are simulated with the tetrahedron based and prismatic based flow solver to validate the viscous term treatment. Next, simulation of turbulent flow over the M6 wing is carried out with the parallelised 3D flow solvers to demonstrate the overall accuracy of the algorithms and the efficiency of the multigrid method. The results show very good agreement with experimental data. A highly stretched and well-formed computational grid near the solid wall and wake regions is generated with the "inflation" method. The aspect ratio adaptive multigrid displayed a good acceleration rate. Finally, low speed flow around the NREL Phase II Wind turbine is simulated and the results are compared to the experimental data.

SUBJECT TERMS: unstructured-grid, multigrid, adaptive mesh refinement, parallel computing

Contents

List of Figures

IV

List of Tables

Nomenclature

Latin Characters

a	-	sound speed
A	-	area, element, node
B	-	element, node
c_f	-	skin friction coefficient
c_p	-	specific heat at constant pressure
C_p	-	pressure coefficient
c_v	-	specific heat at constant volume
C	-	constant parameter, element, node
CFL	-	Courant-Friedrichs-Levy number
e	-	energy
E	-	total energy per unit, error
$\overline{E}$	-	normalised error
f	-	specific function
F, F^+, F^-	-	mass flux vector
G	..	viscous flux vector
H	-	Enthalpy
k	-	turbulent kinetic energy
K	-	constant
L	-	length
M	-	Mach number
N	-	number of nodes, elements
$\hat{n}$	-	outward pointing vector
p	-	pressure
Pr	-	Prandtl number
q	-	heat flux, primitive variable vector
Q	-	conservative variable vector
r	-	radius, distance to the cell centroid
res	-	residual

R	-	source term vector
Re	-	Reynolds number
s	-	area
S	-	strain, source term vector, rotation term vector
t	-	time
T	-	temperature
u,v,w	-	
V	-	volume
V_n	-	dependent variables of specific equations
W	-	residue vector
x	-	position vector

Greek Characters

Δ	-	orthogonal part of the face area vector
Δt	-	time step
ε	-	entropy
κ	-	coefficient of thermal conductivity
γ	-	heat capacity ratio
τ	-	stress tensor
ρ	-	density
μ	-	dynamic viscosity
ν	-	kinematic viscosity
$\tilde{\nu}$	-	dependent variable of a specific turbulence model
τ	-	stress tensor
ω	-	rotational speed, vorticity

Superscripts

$\wedge$	-	unit vector, Roe averaged value
+, -	-	value of left/right side of a face
n	-	iteration level

Subscripts

∞	-	far field
c	-	centre of element
i	-	value of number i (cell, points, etc)
l	-	laminar
L, R	-	value of left/right side of a face
n	-	element index
ref	-	reference value for non-dimensionisation
t	-	turbulence
$trip$	-	value on the transition point

X

| w | - | value on solid surface |
| min, max | - | minimum,maximum |

Abbreviations

API	-	application programming interface
CFD	-	computational fluid dynamics
CFL	-	courant friedrichs lewy number
DNS	-	direct numerical simulation
FDS	-	flux difference splitting
FVS	-	flux vector splitting
FVM	-	finite volume method
LES	-	large eddy simulation
PVM	-	parallel virtual machine
MPI	-	message passing interface
TVD	-	total variation diminishing
VM	-	virtual machine
2D	-	two dimensional
3D	-	three dimensional

Chapter 1

Introduction

1.1 Background

Whereas a particular aerodynamic application could concern about the complex shock wave interaction associated with a rocket, or internal flows in turbomachinery, the common feature is that they are all dependent on fluid mechanics. In order to improve their performance, it is very important to understand the characteristics of flows. A powerful alternative to classical fluid mechanics and experimental methods of aerodynamic analysis is Computational Fluid Dynamics (CFD). CFD is widely accepted as a powerful means to study complex phenomena such as turbulence in the modern engineering community. Furthermore, CFD can provide the possibility of optimising the design of products with dramatic reduction of cost and time of product development.

In the past decade, benefiting greatly from dramatic improvements in computer technology, such as the central processor, memory and network technologies, the development of modern CFD methods has been significantly enhanced so that industrial applications with complex geometries may be considered. After years' research efforts, it is clear that a modern computational method for industry applications must have certain features:

It must be able to deal with complex configurations, like multiple bodies and complex boundaries. This means that either unstructured meshes or multi-block structured grids should be used.

The method must be able to resolve complex flow features, such as shock waves, boundary layer interaction, and flow separation and attachment. Many such complex flow features are associated with turbulence. Due to complex nature of the mathematic modelling of turbulence effects, the most common used method to simulate turbulent flows is still to adopt turbulence models. This would require not only a higher order spatial discretisation scheme and adequate turbulence modelling but also either a global mesh refinement or mesh adaptation technique.

Finally, the ability to obtain results in reasonable computing time is always very important for practical CFD applications. This requires effective convergence acceleration techniques such as the multigrid method and/or the parallel computing technique to reduce overall computing time.

These requirements pose new challenges on both the design of numerical algorithms and computing techniques.

1.1.1 Unstructured-Grid Method in CFD

In past decades, much progress has been made in developing computational techniques for predicting flowfields about complex configurations. These techniques include both structured and unstructured grid methods, both of which have their own advantages and disadvantages.

Conventional CFD methods are usually based on structured grids (H-mesh, O-mesh, C-mesh, etc) with a topologically rectangular structure and usually remain fixed throughout the simulation. This approach has been the mainstream of CFD for many years due to the limitation of computing resource in the past. Unfortunately, the generation of a suitable computing grid is not easy. Many, often conflicting, issues need to be addressed: the leading edge of an aerofoil needs increased grid density to avoid excessive numerical entropy increase, whereas for regions far from solid wall

usually less grid points are desired for economic computing. Also for accurate shockwave resolution grid density is required to be increased; and so on. One method to deal with complex geometry configurations is the application of multi-blocked structured-grids. For this method, the complex computing domain is divided to several relatively regular sub-domains for grid generation. This process is nearly impossible to be automatised because it requires certain CFD expertise and human intervention. In some extremely three-dimensional geometry it is a very difficult task just to generate an adequate computing grid with this multiblock technique.

By contrast, the unstructured-grid methodology offers some significant advantages compared to the traditional structured-grid method for simulating flows over complex geometries. This is mainly attributed to the promise that the construction of unstructured grids around complex configurations, such as a multi-element airfoil, requires much less time than a comparable multiblock structured-grid. Unstructured meshes usually have irregular connectivity and contain triangular elements in two-dimensions, while tetrahedral, prismatic and/or hexahedral elements are used in three-dimensions. This gives the unstructured-mesh method ability to use fine local grids without affecting the mesh in rest of the domain. In addition, because of its irregular connectivity on an unstructured-grid, an automatic mesh refinement technique can be carried out more easily to improve the accuracy of the solution with less computing cost than dealing with a structured-grid. Furthermore, the homogeneous data structures across the computing domain used in unstructured mesh methods enable good load balancing and scalability for parallel computing on parallel computers or cluster systems.

Although the unstructured grid approach enjoys its advantages over the structured grid approach in some areas, the unstructured-grid based flow solvers usually suffer several disadvantages:

- Complex data structures and extra storage are required. A flow solver utilising the unstructured mesh method needs complicated data structure to describe the geometry connectivity. Therefore the efficiency of this method is not as high

as a structured-grid method because of indirect memory addressing, and extra memory is needed to store these information. In some three-dimensional cases, the connectivity data usually requires more storage (both memory and hard disk) than actual flow variables.

- Generation of adequate meshes for viscous computations is difficult. Although the generation of isotropic unstructured meshes for some extreme complex geometry can be done in a matter of hours, the generation of highly stretched viscous grids remains a challenge.

- Accuracy may be compromised. With years of research, many high order schemes have been developed for the structured-grid method. Unfortunately, most of them are unable to be applied directly to an unstructured-grid method.

- Poor convergence rate. The traditional structured-grid permits the use of highly efficient methods such as the Alternative Direction Implicit (ADI) iteration scheme and multigrid to accelerate the solution. Unfortunately, most of these powerful convergence acceleration methods could not be employed easily in the unstructured-grid method or the efficiency is reduced when implemented on unstructured meshes. Generally, CFD flow solvers utilising unstructured grids are slower than structured-grid solvers.

- Discretisation using triangles (2D) and tetrahedral (3D) is more expensive to evaluate than structured-grid using quadrilateral/hexahedra.

The use of unstructured meshes poses new challenges both on design of new algorithms and the grid generation technique in computational fluid dynamics. In the following sections, we will discuss the issues of accuracy, convergence, mesh generation and parallel computing on unstructured mesh methods.

1.1.2 Mesh Adaptation

The accuracy of CFD methods is an interesting topic. It is defined by the difference between a numerical solution and the actual flow, which is usually unavailable. A

numerical solution is often obtained following discretisation of the governing equations on the solution domain. The difference, also referred to as the error, is often associated with two discretisation phases. The error due to first part of the discretisation is often referred to as the modelling error, which is defined as the difference between the actual state of the flow and an exact numerical solution of the mathematical model. The discussion of the modelling error is beyond the current study. Our interest is focused on reducing the second group of the errors that originate from the spatial discretisation of the flow domain to improve the accuracy of the solution.

To reduce spatial discretisation errors, one can either use higher quality computing grids or dynamically refined computing meshes within the areas where the numerical errors are most likely to occur. In the past, much work has been done in developing high order numerical schemes such as the TVD scheme for computational fluid dynamics. It seems the use of a mesh adaptation technique may offer further accuracy improvement in CFD simulations.

Mesh adaptation is one of the major advantages of the unstructured-grid method over the traditional structured one due to its ability to concentrate computational as well as storage resources to regions where they are most needed. (i.e. regions where the numerical error mostly occurs.) Thus, flow solvers utilising mesh adaptation techniques can achieve adequately accurate results with reasonable computational costs in terms of both CPU time and storage. Furthermore, using the mesh adaptation technique can decrease the user expertise and effort required to produce satisfactory simulations by reducing the dependence on the grid used to initiate the process.

The mesh adaptation technique has achieved great success in solving the Euler equations in past years. However, solving the Navier-Stokes equations with turbulence effects using this technique has been less successful. Difficulties to reconstruct quality viscous meshes in the boundary layer and inadequate error estimation both contribute to the failure of this method in viscous flow simulations.

1.1.3 Mesh Generation

Designing a fully automatic two- and three- dimensional mesh generator is the ultimate goal of the research of the unstructured mesh generation. This fully automatic mesh generation requires software that could take the description of geometry boundary definition and produce a well-formed mesh throughout the flow domain without user intervention. There are several difficulties for this "fully automatic" mesh generation:

1. Description of boundaries. In two dimensions, these boundaries could be described as several curves that define the geometry. These curves could be splines or connected straight lines. In three dimensions, it could be difficult to specify a curved surface. A possible way would be using some sort of established geometry systems.

2. Generation of high quality elements. A desire for high quality meshes for complex geometries is the driving force for using the unstructured-grid method. Researchers realised a long time ago that poor mesh quality, often caused by non-smooth mesh and stretched elements, always leads to unsmooth (and very likely unphysical) solutions and poor convergences.

3. Viscous mesh generation problems. For inviscid flow problems, even in some extremely complex geometry configurations, some mesh generation packages could produce well-formed triangular (2D) or tetrahedral (3D) elements in a matter of hours. For viscous flow problems, stretched meshes are required in regions of viscous effect domination, such as boundary layer and wakes. Generation of this kind of mesh is extremely difficult especially for complex 3D configurations.

A huge amount of research has been invested to develop fully automatic two- and three-dimensional unstructured mesh generators. The 2D/3D inviscid mesh generation has reached a mature state. Even for some extremely complex geometry such as a full aircraft, with the aid of modern computers, the generation of high quality unstructured meshes for inviscid simulations are possible. However, the

generation of viscous meshes, especially in 3D complex configurations, is still less than mature. The main reason is that most unstructured-grid generators are isotropic based: triangles (2D) and tetrahedral (3D). Triangular and tetrahedral elements are ideal for discretisation the domain without any preferred direction. In viscous flow simulations, different resolutions in various directions are desired due to disparity of the flow. Isotropic based mesh generation techniques experience difficulties in delivering the desired directional resolution. An alternative mesh generation technique is required to resolve this problem.

1.1.4 Multigrid Method

Generally speaking, flow solvers utilising unstructured meshes are slower than those based on structured grids. The key factors behind the lower computational efficiency of the unstructured grid method are:

1) It requires complex data structures to store the connectivity of elements, introducing the access overhead of memory through indirect addressing.

2) It is difficult to construct an efficient multigrid or implicit solution procedure on unstructured meshes to accelerate the convergence.

3) Solution of viscous flows, due to using a highly stretched triangle (2D) or tetrahedral (3D) in boundary layer regions, results in slower convergence than the more body-fitted quadrilateral (2D) or hexahedral (3D) structured grids.

With carefully optimised coding, the overhead caused by indirect memory access and complex data structure in the unstructured-grid method could be relieved, but it cannot be avoided. The second and third factors can be minimized by using a carefully designed multigrid technique.

A fine mesh resolves small-scale features of the flow field, but is slow to converge. A coarse mesh converges quickly but loses small-scale features of the flow field. The goal of a multigrid method is to obtain a solution with a fine grid resolution, but in a low number of iterations that is characteristic of a coarse grid. From a time-

integration viewpoint, the convergence rate is dictated by the length of time-step, since disturbances with long length scales are propagated at relevant characteristic speeds. The use of coarse grids (thus larger time-steps) should then propagate error disturbances more quickly, leading to faster convergence. For an explicit time-marching scheme, the time-step is limited by the minimum mesh spacing due to the stability requirement. Hence a good coarse mesh level should always increase the time-step to achieve better accelerate ratio.

The multigrid method has been demonstrated as an effective means to accelerate solutions on structured grids. Multigrid on unstructured grids, especially with mixed-elements is still at very early stage of development. In recent years, some progress has been made toward developing multigrid techniques on unstructured meshes. Various successful multigrid methods have been developed for Euler solutions. However, most of these methods achieve less satisfactory results for viscous flow problems due to the presence of high grid aspect ratio.

High aspect ratio grids are commonly encountered near wall regions in high Reynolds number flows, where the grid must be refined very tightly in the direction normal to the wall to resolve the high velocity gradient. A second type of problem in which highly stretched meshes may be found is in the mixing flow regions. The magnitude of these grid aspect ratios maybe order of 10 to 1000 depending on cases. For two-dimensional problems, the grid aspect ratio (AR) of a grid may be defined as,

$$AR = \frac{\Delta x}{\Delta y} \qquad (1\text{-}1)$$

For a typical CFD problem, the time step to march the solution to a steady state can be obtained by computing the minimum time step in the two directions,

$$\Delta t_i = \min\left\{\frac{CFL \times \Delta x}{\lambda_x}, \frac{CFL \times \Delta y}{\lambda_y}\right\}_i \qquad (1\text{-}2)$$

Here, λ_x and λ_y are the acoustic eigenvalues in the respective coordinate directions.

The time step definition above illustrates the problem experienced with high aspect ratio grids. When the aspect ratio is higher than unity, the time step is likely to be restricted by the time step in the direction of the smaller grid spacing. This results in poor error damping and propagation. Thus slow to converge. The problem of solving a complex flow model such as the Navier-Stokes equations using an unstructured mesh method requires efficient and robust solution acceleration means to reduce the stiffness caused by the high grid aspect ratio.

1.1.5 Parallel Computing

CFD, as its name implies, inevitably involves computing issues, such as CPU power and memory technology etc. Obtaining results in less time has always been a major consideration in CFD. Generally speaking, there are two ways to reduce the computing time: using more efficient numerical methods to accelerate the solution, such as multigrid approaches and implicit schemes, or using computers that are more powerful.

The computing power of the fastest computers has grown exponentially from 1940's to the present, averaging a factor of 10 every five years. As computers become ever faster, it can be tempting to suppose that they will eventually become "fast enough" to solve all the computing problems in very short time. However, history suggests that as a particular technology satisfies known applications, new applications will arise that are enabled by that technology and that will demand the development of new technology. Development at the high end of computing has been motivated by numerical simulations of complex systems such as weather, climate, mechanical devices, electronic circuits, manufacturing processes, nuclear reaction and chemical reactions.

A very important trend changing the face of computing in recent years is the enormous increase in the capabilities of the networks, performance of commodity processors as costs of the computer and networking equipment simultaneously drop. This trend makes it feasible to develop applications that use physically distributed

resources as if they were part of the same computer. A typical CFD application of this sort may be clustering commodity processors together with available networking equipment, to complement mid-level high performance computing systems.

A cluster is a type of parallel or distributed processing system, consisting of a collection of networked stand-alone computer systems working together as a single computing resource. On a typical cluster system, each of the machines can be a complete system, usable for a wide range of other computing applications. This leads to the suggestion to claim all the wasted computing power of old PCs/ workstations. This idea is such a temptation since in some places one can easily find many old PCs that are suitable for clustering.

Although this type of cluster parallel computing is cheap, highly available and can scale to very large systems, there are some problems for using cluster parallel computing on CFD applications. First, most networked hardware is not designed for cluster parallel computing. The latency is generally very high and bandwidth relatively low compared to traditional parallel computer systems with attached processors. Secondly, most CFD codes are designed for serial computing. Some may need major modifications.

To take advantages of the cluster parallel computing for CFD applications, especially when using the systems not designed for high performance computing, some considerations have to be put on the network design, mesh partitioning, data structure, interface treatment and communication schemes.

1.2 Objectives and Contributions of the Present Work

The objective of the current research is to develop efficient and accurate 2D and 3D unstructured mesh flow solvers for aerodynamics applications.

The specific objectives of this project are to:

- Develop accurate and efficient flow solvers capable of solving the Euler/Navier-Stokes on 2D/3D unstructured meshes.

- Develop a solution mesh adaptation technique to improve accuracy of inviscid and viscous flow simulations for aerodynamic applications on unstructured meshes.

- Develop an efficient multigrid method capable of dealing with high grid aspect ratio in viscous flow simulations.

- Validate the numerical algorithms and assess their accuracy and efficiency.

- Explore the possibility of clustering current office PCs in University of Durham for parallel computing and identify the strengths and weaknesses of this kind of cluster system for CFD applications. Develop accurate and efficient parallel numerical algorithms and conduct preliminary testing to verify the effectiveness and potential of these algorithms.

- Conduct numerical studies on the flows around a wind turbine blade with the numerical algorithms developed.

Some specific contributions have been made in the following areas:

- Three distinct unstructured flow solvers: a triangle-based 2D solver, a tetrahedron-based 3D solver and a prism-based 3D solver (semi-structured), have been developed to solve the Navier-Stokes equations for aerodynamics applications. These flow solvers feature cell-centred Finite Volume discretisation schemes applicable to arbitrary complex geometry configurations, Roe's upwind scheme, spatially second-order order, and the explicit multistage Runge-Kutta method.

- An adaptive mesh refinement technique has been incorporated with the 2D flow solver to achieve accurate results with reasonable computing cost.

- An "inflation" strategy for generating 2D and 3D viscous meshes has been developed and validated.

- A new multigrid approach, Aspect-Ratio Adaptive Multigrid, has been developed and validated for two- and three- dimensional flow cases. The effectiveness has been demonstrated.

- The parallel computing technique has been coupled with the three-dimensional flow solver on a PC cluster system to reduce computing time.

- Flows around an NREL Phase II wind turbine blade have been simulated with the presented 3D flow solver. Satisfactory results have been achieved.

1.3 Thesis Outline

Chapter 2 provides a literature review of development of CFD methods on unstructured meshes and relevant subjects.

In **Chapter 3**, flow models and numerical discretisation are summarised. The governing equations of fluids are presented, followed by nondimensionalisation and turbulence modelling. Next, the spatial discretisation, including two- and three-dimensional finite volume schemes, a second order scheme construction, and an upwind technique, are described. In this section, the idea of using an alternative control volume in viscous layers is introduced. Subsequently, the temporal discretisation based on a multistage Runge-Kutta approach is discussed. This chapter is concluded with the description of physical and numerical boundary conditions.

Chapter 4 outlines the mesh generation method and mesh adaptation techniques. First, the advancing front method for generation of two-dimensional inviscid unstructured meshes is reviewed. Next, a new strategy for generating viscous unstructured meshes for two and three dimensions is established. The final section of this chapter discusses the mesh adaptation techniques including the error estimation and the mesh reconstruction procedure.

Chapter 5 outlines a multigrid method for improving overall convergence rate of unstructured-grid based flow solvers. It starts with an introduction of different ways to generate the sequence of grids. First, a traditional semi-coarsening method – Direct

Connected Multigrid is proposed. Then a new approach – Aspect ratio Adaptive Multigrid is presented for viscous computations on high aspect ratio grids. This chapter ends with a discussion of the timestep and stability of this multigrid method.

The parallel computing technique is presented in **Chapter 6**. It includes the mesh partitioning techniques, parallel computing environment, load balancing and communication schemes.

In **Chapter 7**, the algorithm developed in previous chapters is applied on several 2D test cases. The first case is inviscid flows in a bump channel, which is designed to check the numerical accuracy of inviscid algorithm established in previous section. To further examine the accuracy of the inviscid algorithm and demonstrate the efficiency of the multigrid method and effectiveness of the mesh adaptation technique, inviscid flows over NACA0012 airfoil are simulated. The next test case is about laminar flow over a flat plate. In this case, the accuracy of using the upwind scheme for viscous simulation and boundary condition treatment are examined. The turbulent flow cases include viscous flows over a flat plate, the RAE2822 Airfoil, and a turbine cascade. The turbulent flow over flat plate is presented to check the implementation of current turbulence models and the coupled solution method. The RAE2822 airfoil flow case is presented to demonstrate the accuracy of the current method and effectiveness of the aspect ratio adaptive multigrid method. The simulations of flows in a turbine cascade include detailed comparison with experimental data as well as results using the traditional multigrid method and the aspect ratio adaptive multigrid method. This chapter ends with the summary of the 2D algorithm and some conclusions.

In **Chapter 8**, the 3D validation and discussion are presented. The algorithms developed in previous chapters including the multigrid and parallel computing techniques are applied to 3D inviscid and turbulent flows. The first case is an inviscid flow over the ONERA M6 wing, for which experimental data is widely available in the public domain. This provides a validation on the inviscid algorithm developed in previous chapters. In the next section, the parallel computing results are presented on a PC cluster system. Laminar and turbulent flows over a flat plate are discussed to

check the viscous treatment and turbulence models used in current research. Next, viscous flow over the ONERA M6 wing is simulated to demonstrate the accuracy of the current 3D flow solvers. The result is achieved on the cluster system with a Multiple Instruction Multiple Data (MIMD) implementation. The convergence rate of the new multigrid strategy is also presented. The last 3D test case concerns flow over a wind turbine. Inviscid solution has been achieved on a single blade configuration. Pressure distribution comparisons with experimental data are presented. Attempt has been made to solve the turbulent flow around the wind turbine blade has failed. The reason has been investigated. This chapter is concluded with the discussion of overall 3D results and general conclusions.

Chapter 9 summaries the thesis and offers some conclusions and suggestions for future research.

Chapter 2

Literature Review

This chapter is an overview of unstructured mesh methods in computational fluid dynamics. A survey of numerical methods including the finite volume method, upwind scheme, turbulence modelling is also presented with particular emphasis on unstructured meshes. The discussion of unstructured mesh generation in complex geometry configurations is focused on the generation of highly stretched viscous meshes. Next, the efforts of improving the accuracy of unstructured mesh methods with solution mesh adaptation are reviewed. The discussion of the multigrid method on unstructured meshes is focused on different methods in simulating viscous flow problems with high aspect ratio grids. Further topics, such as parallelisation software, mesh partitioning and load balancing methods and state-of-art of parallelised CFD flow solver are also discussed in detail. This chapter also provides a brief review of CFD in wind turbine applications.

2.1 Unstructured-Grid Method

The use of arbitrary control volumes to solve fluid equations can be traced back to early 80's. Jameson and Marvriplis' (1986) leading work of solving the Euler equations on two-dimensional irregular meshes is the earliest results in this field. They effectively extended the Finite Volume Method (FVM) established on structured grids and use a central difference scheme with some dissipative terms to suppress the odd-even decoupling. Their finite volume scheme is based on a cell-centred setting on triangular meshes, which are obtained from subdividing structured grids. Since then, much research has been carried out on the development of flow solvers based on unstructured meshes, including the mesh generation methods, discretisation schemes, higher order schemes, convergence acceleration methods, data structures and parallel computing techniques.

2.1.1 Finite Volume Scheme

The majority of discretisation of the governing equations on unstructured meshes is based on two lines of methods, Finite Element Method (FEM) and Finite Volume Method (FVM). The finite volume scheme has achieved great success both on structured meshes (Jameson et al. 1981; Denton 1983; He 1993) and unstructured meshes (Frink 1996; Marvriplis 1992; Holmes 1994).

In the framework of a finite volume scheme, there are several distinct choices of the control volume for unstructured meshes, such as cell-centred (Figure 2.1a) and cell-vertex (Figure 2.1b) depending on where to store the flow variables. The cell-vertex method exploits an efficient edge-based data structure, and has been demonstrated to be easier in implementation of parallel computing and multigrid (Marvriplis 1990; Marvriplis 1992; Venkatakrishnan and Mavriplis 1994). The cell-centred method seems more expensive than the cell-vertex method on a given mesh (Barth 1991) but the solution quality of the cell-centred setting is clearly superior to the cell-vertex one (Pothen et al. 1993). The issue of cell-vertex vs. cell-centred approximations is still an open one. In the present work, a cell-centred

finite volume scheme on both two and three dimensional unstructured meshes is implemented.

(a) 2D cell-centred scheme   (b) 2D cell-vertex scheme
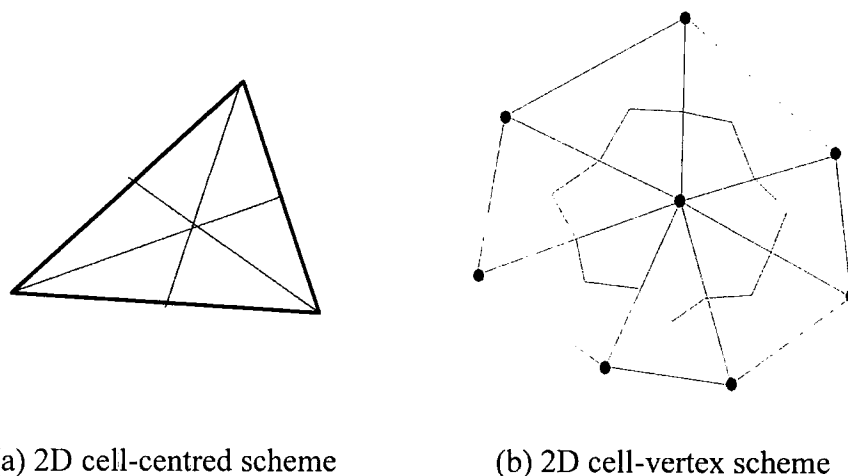
Figure 2.1 Cell-centred vs. cell-vertex scheme

The three-dimensional simulations of inviscid flows based on tetrahedral cells (Figure 2.2) have achieved tremendous success (Frink et al. 1991; Dawes 1992; Marvriplis 1992; Crumpton and Giles 1997). However, using this kind of control volume in viscous flow simulations has been less successful. This is mainly due to the difficulty to generate adequate computing meshes in viscous effect dominated regions and poor convergence rate caused by using highly stretched tetrahedral elements in these regions. Aftosmis *et al* (1994) examined the accuracy of viscous flow simulations by using various triangular meshes and quadrilaterals. The conclusion is that using triangular elements in boundary layer regions can not achieve improvement in solution accuracy. This leads to the development of the hybrid discretisation scheme.

Connel and Braaten (1994) adopted a hybrid approach in 3D viscous flow simulations. In their approach, structured/semi-structured grids are used in the near wall regions to overcome the difficulty of mesh generation in these regions. Sbardella *et al* (1997) also presented a hybrid discretisation scheme for solving Navier-Stokes equations in turbomachinery applications. In both cases, layers of structured-grids are used near wall regions and several successful cases have been demonstrated. Therefore, the surfaces of solid walls are discretised with structured-grids. This

method is proven to be effective in some applications (Sbardella and Imregun 2000; Sayma et al. 2000), such as flows around turbine blades, in which the geometry of the blade surface is relatively simple. However, when simulating flows around an object whose geometry of its surface is complex, such as a whole aircraft, it is often very hard to discretise the surface with a single structured-grid, and this method becomes less useful.
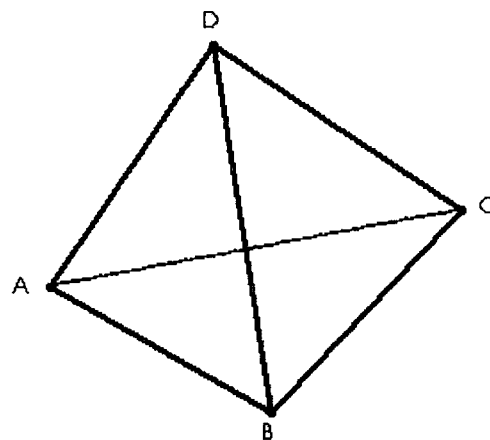
Figure 2.2 Three-dimensional control volume: tetrahedron

2.1.2 Upwind Scheme

In the solution of hyperbolic equations such as Euler equations, the theory of characteristics is crucial in determining the directions of the signal propagation. The information gained from the characteristics theory has been very useful not only in the boundary condition treatment but also in the development of a computational method: upwind scheme. The use of upwind schemes offers several advantages over a central-difference formulation (Amaladas and Kamath 1998). Due to the fact that the upwind scheme being a characteristic based method with the introduction of physical properties of the equations into the discretised formulation aiming at preventing numerical oscillations, while artificial dissipation terms have to be added to a central-difference scheme for stability reasons. For viscous flow simulations, with the upwind scheme the resolution of the boundary layer typically requires only half as many points as with a central-difference code (Zheng and He 2001).

Venkatakrishnan and Barth (1989) proposed the idea of using an upwind scheme on two-dimensional triangular meshes to improve the accuracy of the solution. They employed a variable extrapolation - MUSCL scheme (van Leer 1979) on a cell-vertex setting, which is devised by Desideri and Dervieux (1988) on unstructured-grids, to achieve higher accuracy, as done by other authors (Frink et al. 1991; Frink 1992; Knight 1993). However, oscillations are still present in their results. The following work by Barth and Jespersen (1989) presented a monotonicity principle in multidimensional cases. The idea is to reconstruct the distribution of flow variables in a control volume to be bounded by the values of its neighbour elements. This reconstruction satisfies the monotonicity principle by constructing a truly multidimensional limiter. They employed Roe's upwind scheme (Roe 1981) to evaluate the inviscid fluxes. This method demonstrated smooth results even in transonic flow cases.

Frink (1994) presented another approach to achieving oscillation-free in transonic flow cases. A weighted averaging procedure is employed to interpolate the flow variables from the cell centre to mesh points in a cell-centred finite volume setting. The weighted averaging is based on the distance of a mesh point to its cell centre. Flow variables on the centre of elements and mesh points are used to compute gradients within the cell. Roe's approximate Riemann solver (Roe 1981) and an explicit multistage Runge-Kutta scheme (Jameson et al. 1981) are employed to compute inviscid flux contributions and to advance the solution to steady state. Although this reconstruction is linear and not monotonicity preserving, it seems that the averaging process generates enough dissipation to overcome the oscillation in transonic flow simulations. In this method, the averaging procedure is performed on elements and nodes, while a limiter has to be applied on edges (2D)/ faces (3D), which are outnumbered elements and nodes in most cases. Thus, this approach seems to be slightly more efficient than an MUSCL scheme. Holmes and Connell (Holmes 1994) proposed a modified reconstruction procedure which is linearity preserving. Frink et al (1996) extended the reconstruction procedure to three-dimensional unstructured meshes.

2.1.3 Turbulent Flow Simulation

In the light of the success of the three-dimensional Euler solution (Marvriplis 1992; Frink 1996; Barth 1995; Frink et al. 1991; Crumpton and Giles 1997; Venkatakrishnan et al. 1991), some attempts have been made toward solving the Navier-Stokes equations on unstructured meshes (Wang et al. 1999; Mavriplis 2000; Barth 1995; Sbardella and Imregun 2000; Haselbacher et al. 1999). In the early stage, the gradients in an element used for viscous fluxes computation are worked out by simply integrating all the edges (2D)/ faces (3D) which composite the element. This procedure is very computationally expensive because of the need for integrating over all the faces. Barth (1991) proposed the idea of discretisation of viscous terms using a finite element procedure which is less expensive to compute. Another contribution of his work is an edge-based data structure, which greatly improved the efficiency of CFD flow solvers based on unstructured meshes and is widely used in other researchers' work (Frink 1994; Sbardella et al. 1998; Marvriplis 1992) ever since. The idea of the discretisation of viscous terms using a finite element procedure can also be found in Frink's work (1996).

One important phenomenon for complex viscous flows is turbulence, which is very difficult to simulate due to the existence of a wide range of scales. The turbulence effect is normally modelled by using turbulence models in the CFD community. The most popular turbulence models can be classified by the number of equations used to calculate turbulence effects as: zero-equation models, one-equation models and two equation models. Due to the lack of structured-like grid lines, most popular zero-equation models such as the Baldwin-Lomax model (Balwin and Lomax 1991) are very hard to be implemented on unstructured meshes. Nevertheless, there are several reports of success with this model (Marvriplis 1991). One equation and two-equation models have the advantage of being easier to be implemented on unstructured-grids and could potentially achieve better results. A one-equation model, the Spalart-Allmaras model (Spalart and Allmaras 1992) gains popularity on unstructured meshes; partially because it is less expensive than most two-equation models and easier to be implemented on unstructured-grids than most algebraic

models. Frink (1996) incorporated the Spalart-Allmaras model in his three-dimensional flow solver with a wall function for computing turbulent flow over the ONERA M6 wing. His results show reasonable agreement with the experiment (Schmitt and Charpin 1979). Wang *et al.* (1999) report their extensive research on two two-equation models, the k-ε and SST k-ω model (Wilcox 1993), and Spalart-Allmaras one equation model using an unstructured-grid flow solver. Their results show advantage of the two-equation models over the one-equation model in some cases. However, Spalart-Allmaras model produces excellent results in most cases and it is less computationally expensive.

2.2 Unstructured Mesh Generation

Unstructured mesh generation is a relatively new field for most CFD researchers. Within a few years tremendous advances in many diverse fields have been made toward fully automatic mesh generations both in two- and three-dimension.

There are two major unstructured mesh generation methods for the two-dimensional flow computation, the advancing-front method (Lo 1985) and the Delaunay triangulation method (Bowyer 1981; Waston 1981). The advancing-front method starts with boundaries of the domain as the initial front. Then triangles are generated from the current front into empty domain, and the front is updated. The operation is repeated until the whole domain is triangulated. The Delaunay method adopts the empty circumcircle property of the computational domain. It is generally more efficient than the advancing-front method (Liu and Hwang 2001). However, the advancing-front method has the advantage of being more robust because the boundary integrity is guaranteed. A full review of unstructured mesh generation theories and methods can be found in Ref. (Barth 1995). Currently, the automated generation of unstructured mesh for simulations of inviscid flows has reached a fairly mature state (Lohner and Parikh 1988; Jin and Tanner 1993; Muller 1996). Generating computing meshes for some complex configurations such as full aircraft can be complete in a matter of hours.

In simulations of viscous flows with high Reynolds number, the gradients normal to the wall are several orders of magnitude larger than those along the wall are. Thus, highly stretched elements are required to resolve the rapid changes of the flow. Generation of high quality unstructured meshes for viscous computations in complex geometries remains a difficult task.

Despite the difficulties, some attempts have been made toward the generation of highly stretched grids. Löhner and Cebral (2000) presented their non-isotropic mesh generation method. In this method, an isotropic mesh (suitable for Euler solution) is generated prior to a procedure of enrichment with points in order to achieve highly stretched grids. Element reconnection is carried out with a constrained Delaunay approach. The stretched tetrahedron-based mesh of a three-dimensional generic hypersonic flyer is demonstrated in their report. This method is very efficient in most cases because it does not require any surface recovery. However, a potential problem for this method is that the quality of the final mesh highly depends on the Delaunay procedure, which could fail in some extreme complex configurations.

Recently, there has been renewed interest in hybrid structured-unstructured grids and mixed element unstructured grids (Sbardella et al. 1997; Sayma et al. 2000; Sbardella et al. 1998). Such methods offer the advantage of reduced complexity of the grid and possibly increased accuracy compared with equivalent pure triangular (2D) or tetrahedral (3D) meshes. This is particularly true for viscous flow simulations since with a hybrid or mix-element methods, hexahedral or prismatic elements could be easily used to mesh the regions in/near the solid wall. The hybrid method combined with an advancing layer method and the Delaunay reconnection has been used successfully in several cases (Haselbacher et al. 1999; Haselbacher and Blazek 2000; Sayma et al. 2000). In spite of its great flexibility in the three-dimensional mesh generation, the mixed-grid method shares the difficulty of generating high quality 3D mixed-grid meshes in a generic 3D computing domain. Furthermore, due to the difficulties to implement an efficient multigrid method and mesh adaptation technique, the performance and accuracy of the flow solver using this kind of discretisation could suffer.

Holmes (Holmes 1994) proposed an "inflation" method for viscous unstructured mesh generation. In this method, a few layers of structured-grids are used to "wrap" the airfoil, and triangles are deployed in the rest of domain. This method has the advantage of being easy to implement and high quality grids in the boundary layer. However, the extension of this method to three dimensions is not easy and may lose the some of the flexibility if hexahedron is used in these layers (Sbardella and Imregun 2000; Sayma et al. 2000). In two dimensions, the superior discretisation properties of employing several layers of structured grids were investigated numerically by Haselbacher (1999). Peiró and Sayma (1995) reported a similar mesh generation scheme for 3D turbomachinery applications.

A brief survey of some of the fundamental algorithms in unstructured mesh generation was presented by Owen (1998). Triangle, tetrahedral, quadrilateral and hexahedral mesh generation methods currently in use in academia and industry are discussed and categorised. His report also includes an informal survey of currently available mesh generation software in public domain and a comparison of their main features.

2.3 Solution Mesh Adaptation

The accuracy of numerical simulations is one of the main concerns in modern computational fluid dynamics. There are various ways to achieve accurate results, one can either employ a higher accurate scheme such as a high order upwind scheme, better turbulence models, or enrich the computing grids, i.e. using finer mesh. In this section, we will discuss the mesh enrichment approach on the unstructured mesh: mesh adaptation.

In order to improve the accuracy of solution, an appropriate change of mesh resolution in the region of high error may be needed. In the rest of the domain, where the error is sufficiently small, such changes may not be necessary. This method is often called mesh adaptation or refinement.

There are several ways to achieve adaptivity:

- Increase the degree of polynomial approximation to improve the overall solution quality;

- Relocate the grid points in the regions of rapid change of solution.

- Relocating grid points and enrich grid.

Obviously last one is the best way, because it provides the greatest way to control the cell size to resolve flow feature such as shock wave, shear layer, wake, separation and reattachment. In this method, the problem is firstly solved on a coarse mesh to roughly capture the basic feature of flows; the resulting solution is then analysed to determine where more grid points are needed, and an improved mesh is generated. The problem is solved again on the new mesh using the solution of the coarse mesh as an initial guess. The process is repeated until the required accuracy is achieved.

Solution adaptive grids are increasingly being used in simulation of steady and unsteady flows (Hawken 1991; Dawes 1993; Dawes 1994). The adaptation concept of unstructured mesh applied to the Euler equations in complex geometry has gained great success (Marvriplis 1990; Marvriplis and Jameson 1987), but solution adaptive grids for the Navier-Stokes equations at high Reynolds numbers are less well developed. The main reason is that the presence of different scales in viscous flows makes the adaptation complicated (Vilsmeier and Hanel 1993). Despite this difficulty, significant progress has been made in numerical computations of the Navier-Stokes equations on fully unstructured adaptive meshes (Dawes 1993; Pelletier and Ilinca 1997; Holmes 1994; Ilinca et al. 1997; Roehl and Simon 1999).

The over-resolved problem described by Sidén and Dawnes (1990) also has significant impact on the usefulness of the solution adaptation method in viscous flow simulations. In most current solution adaptation methods, the indicator used for refinement is a scalar and hence not directional. It tends to refine the grid in boundary layer regions without considering the local mesh topology and flow features: highly stretched grid and strong 1D flow. Sidén et al. (1990) first proposed the idea of using thin layers of structured-grids, consist of stretched elements. However, the mesh

refinement producer used in his work still produced too many inefficient near-equilateral elements in the boundary layer, shock, and wake regions. This over-resolved problem remains a challenge in viscous flow simulations with a mesh adaptation technique.

2.4 Multigrid Acceleration

Multigrid has been demonstrated as an effective means to accelerate the solution for both traditional structured-grid methods (Jameson 1994; Denton 1983; Swanson 2001) and unstructured-grid methods (Marvriplis and Jameson 1987; Carre 1997; Mavriplis 1996; Mavriplis 1999). For structured-grid applications, multigrid has become a routine practice (Steinthorsson et al. 1993; He 1993; Roberts et al. 1997). But multigrid on unstructured grids, especially with mixed-elements is still at very early stage of development. Depending on how the sequence of grids is constructed, multigrid can be classified as two families: nested methods and non-nested methods.

Most non-nested methods start from generating several completely independent meshes for a specified geometry. These meshes are essentially independent from each other and the sequences of meshes do not always have common points. The connection between the various meshes is an inter-grid operator to interpolate flow variables and to transfer residuals. This method has demonstrated efficiency and robustness in some 2D inviscid flow cases (Marvriplis and Jameson 1987). In 3D complex geometry cases, the generation of such sequence of unstructured meshes becomes a very difficult task considering the current state of three-dimensional unstructured mesh generation methods. Furthermore, the introduction of complex bilinear interpolation, which is essential to transfer flow variables and residual between various meshes of the sequence, means extra expensive computing (Marvriplis 1992) and performance penalty. Peiro and Sayma (1995) reported their 3D multigrid implementation for unstructured meshes. In their method, a hybrid approach is adopted in which the near wall mesh is generated by a hyperbolic type of mesh generator. The main drawback of this method is that the quality of generated elements in coarse mesh levels is very difficult to control.

There are various possibilities for implementing a nested multigrid on unstructured grids. An obvious choice is to couple multigrid method with an adaptive mesh refinement technique and to use the sequence of meshes generated by the adaptive mesh refinement procedure as difference mesh levels. This approach starts with a relatively coarse mesh. The finer levels are generated by subdivision of coarser levels. In this way, strictly nested levels are generated. This nested multigrid method coupled with adaptive refinement approach has been adopted by several authors to use the adapted meshes as the multigrid levels (Marvriplis and Jameson 1987; Marvriplis 1990; Connell and Holmes 1994). This method has the advantage of being easier to generate the coarse levels. However, multigrid efficiency is limited by the efficiency of solution on the fine mesh and the adaptive strategies are greatly restricted, as pointed out by Mavriplis (1996).

Another nested multigrid approach starts with a fine mesh definition, and then constructs the sequence of coarse mesh levels automatically by the connection relation of mesh elements. This method is embodied in automated coarsening approach. Many researchers (Lallemand *et al.* 1992; Venkatakrishnan and Mavriplis 1994; Diskin 1999) proposed to generate the coarse levels using neighbouring relations (volume-agglomeration) on a dual mesh. Marvriplis and Venkatakrishnan (1995) have demonstrated the efficiency of a multigrid method based on this volume agglomeration technique. The advantage of the method lies on reliability of automated coarsening algorithm. However, there are some difficulties of these kinds of method: firstly, the convergence is not good enough for low mach flows (Carre 1997). Secondly, this method cannot deal with stretched meshes with high aspect ratio very well in solving viscous flow problems.

High aspect ratio grids are commonly encountered near wall regions in high-Reynolds number flows, where the grid must be refined very tightly in the direction normal to the wall to resolve the high velocity gradient. A second type of problem in which highly stretched mesh may be found is in the mixing flow regions. In both cases, the high aspect ratio grid causes a strong disparity in the wave

propagation speeds in two coordinate directions resulting in serious convergence deterioration.

For multigrid in traditional structured-grids, directional coarsening can remedy the problem. The idea is to coarsen the grid only in the direction normal to the grid stretching. Denton (1983) and He (1993) employed this idea to alleviate the stiffness in unsteady viscous flow simulations. A similar idea can also be found in Giles and Haimes' (1993) work.

On unstructured meshes, the directional coarsening method can also be implemented using a graph algorithm. It starts by removing mesh points from an element containing the points and anisotropy in the grid. The criterion to remove a mesh points is based on the values of the stencil coefficients. After the points removing procedure is completed, the rest of the mesh points are to form a coarser mesh level. A major drawback of this method is that it may result in a coarse mesh of higher complexity.

Marvriplis (Mavriplis 1999) proposed another directional multigrid to improve the effectiveness of multigrid for viscous flows. In his implementation, the graph-based coarsening algorithm is only employed in the boundary layer and wake regions. Once these regions have been coarsened, a new unstructured mesh is generated for the rest of domain with specified element size. This method has been demonstrated to be successful in several two dimensional viscous flow cases. However, the convergence rate of his method is still not ideal and remeshing the inviscid domain rather than coarsening it increases the complexity of the method.

2.5 Parallel Computing

Although parallel processing has been used for many years in many systems, it is still somewhat unfamiliar to most CFD researchers. The idea of parallel computing is initially developed by Massive Parallel Processors (MPP) vendors to provide high performance computing for energy applications, weather prediction, and earthquake simulations. Because of high cost and low accessibility of these supercomputers, the

parallel computing technique did not gain momentum until the later 1980s when high performance microprocessors, high speed networks and standard tools for parallel computing were widely used. The trend in parallel computing is moving away from supercomputers to cheaper, general purpose cluster systems consist of loosely coupled systems, such as PCs. This approach has a number of advantages, including being able to scale to large system and cost/effectiveness. However, there are some differences between this cluster system and traditional parallel computer:

1) On the traditional parallel computer, the parallel computing software package is supplied by the system vendor. It requires very little modification of the CFD code to utilise parallel computing. On a cluster PC system, communication software is required and some major changes have to be made in some CFD codes to adopt parallel computing.

2) This kind of cluster system is usually interconnected through some kind of network which is not designed for parallel computing. The latency is generally very high and bandwidth is limited compared to supercomputers. Communication costs become crucial to the computing performance of a cluster system.

3) On some shared memory parallel computers, mesh partitioning is not required. But on a cluster system, since the memory is located on each node in the cluster system, the computing mesh has to be divided into several parts for efficiency reasons.

2.5.1 Parallelisation Tools

There are many software packages suitable for parallel computing for a cluster system in public domain, including Chameleon (Gropp and Smith 1993), BSP (Bulk Synchronous Parallel Model) (Oxford University Computing Laboratory 2000), OPlus (Crumpton and Giles 1993), PVM (Dongarra, Geist et al. 1995), MPI (Gropp and Lusk 1999).

PVM is a freely-available, portable, message-passing library generally implemented on top of sockets. It is supported in a widely range of hardware and software

platforms. These include single-processor and SMP Linux machines, as well as clusters of Linux or Windows machines linked by networks. In fact, PVM will even work across groups of machines in which a variety of different types of processors, configurations, and physical networks are used. Best of all, PVM is freely available and is clearly the de-facto standard for message-passing cluster parallel computing. PVM also provides facilities for parallel job control. It is important to note, however, that PVM message-passing calls generally add significant overhead to standard socket operations, which already had high latency. Furthermore, the message handling calls do not constitute a particularly "friendly" programming model, so PVM is commonly used as the "portable message library target" for high-level language parallel compilers.

The MPI (Message Passing Interface) is developed with the intent to be a standard message passing specification for Massive Parallel Processor (MPP) vendors. It provides a set of message passing constructs and supports features of various MPP and networked clusters.

Both PVM and MPI can be used for parallel computing either on a cluster system or multiprocessor system. However, there are some fundamental differences between them. The MPI is designed to deliver high performance on MMP systems. Thus, MPI focuses on message passing and explicit resource management. It contains a larger set of point to point and collective communication routines than PVM does. By contrast, the PVM is built around the concept of the "virtual machine", which supports dynamic resource management (add/remove hosts from virtual machine, spawn/kill jobs). This virtual machine may consist of heterogeneous hosts. It means hosts may be different architectures (PCs, workstations), running different operational systems (windows, UNIX, Linux, etc), connected by different networks (Ethernet, ATM, etc) and user's program may be developed using different programming language (C/C++, FORTRAN). Generally speaking, MPI is faster within a large multiprocessor system, even though some reports suggest PVM and MPI have very comparable performance (Wiel et al. 1996). Furthermore, PVM is capable of running applications over heterogeneous networks to provide fault tolerance, dynamic resource management

and job control functions. A full comparison of features provided by PVM and MPI can be found in ref. (Geist et al. 1996).

2.5.2 Mesh Partitioning

In a cluster system, processors and memory are distributed across the system. Thus, a computing problem has to be divided into several block/zones in order to distribute the data to individual processors in the system. For a CFD application, the computing grid needs to be partitioned to number of sub-grids and mapped to processors. There are several popular algorithms for mesh partitioning, such as the coordinate bisection, graph bisection and spectral bisection methods.

The coordinate bisection method is a very intuitive method, which comes immediately into mind when considering a mesh partition problem. It takes advantage of easy to compute and requires least memory. However, this method completely ignores the communication between partitions in parallel computing. When a computing grid is divided into several smaller sub-grids, it is inevitable that the parallel flow solver needs to exchange information of the interface at each synchronisation point to result in a physically relevant solution. In a cluster system, this exchanging of information is done by a communication technique: message passing. Because the sub-grids are located in processors which are interconnected by some sort of network, the message passing is likely to incur some communication overhead. In such a cluster system, overhead costs such as the message formation, package sending/receiving, latency of communication are so high that it severely restricts the parallel programs that can run efficiently. The overhead must be driven down to more reasonable range for efficient computing. One optimisation technique to reduce the impact of the high overhead is to reduce the total communication volume during the parallel computing in a cluster system.

In response to the incentive to reduce the communication overhead in a cluster system the number of edges cut during the partitioning of the mesh has to be minimized. A fair amount of research has been put into the general problem of partitioning data assuming the data are representable as a graph. The resulting problem is called the

graph partitioning problem. The spectral bisection method proposed by Simon (1991) is based on the graph partition algorithm proposed by Pothen, Simon and Liou (1990). Simon (1991) examined these three methods for parallel computing of CFD problems and concluded that the spectral bisection method has significant improvement over the other two in terms of resultant partitions shape and edge cut number. Barth (1995) and Venkatakrishnan (1991) also evaluated and reviewed these partitioning methods and the conclusion is that the graph-based methods are more computationally expensive and yield better partitions.

2.5.3 Parallel Computing With CFD

The concept of parallel computing technique for CFD has been around for many years. It has not been widely accepted in production engineering environments mainly due to the complexity of parallel programming and prohibitive price of massive parallel processor system. Due to increase in the capabilities of the networks, performance of processors as costs of the computer and networking equipment simultaneously drop, many researchers begin to parallelise their CFD solvers in order to reduce the computing time (Crumpton and Gile 1997; Venkatakrishnan et al. 1991).

Mavriplis (2000) presented a parallelised unstructured flow solver on unstructured meshes. In his work, a viscous flow solver is ported to parallel machines with distributed memory using an explicit domain-decomposition and message passing approach. The message passing in his work is via an MPI implementation and openMP. A weighted partitioning strategy is described that incurs minimal additional communication overhead. Several two- and three- dimensional high lift cases are demonstrated on a 128 processor SGI Origin 2000 machine and a 512 processor Cray T3E machine. Very good speedup (up to 300 on the 512 processor machine) has been observed. The scalability of unstructured mesh method is demonstrated. However, these computing are performed on MPP systems, which has very high bandwidth and low latency compared to a cluster system consists of networked PCs.

Hammond and Barth (1992) developed a data parallel mesh-vertex upwind finite-volume scheme for solving the Euler equations on triangular unstructured meshes. The main contribution of their work is the introduction of a novel vertex-based partitioning scheme that minimises the computation and communication costs associated with parallel computing on a massively parallel computer CM-2 with 8K processor.

Baggag et al. (1999) presented their work of a parallelisation of an object-oriented unstructured aeronautics solver. They used the idea of the Object-Oriented Programming (OOP) technique in computational fluid dynamics. The parallel implementation is based on MPI. A compact Galerkin method combined with an explicit time marching method is used in their solver for time accurate computations. The main contribution of their work is the code structure, object model and data structure in their parallelisation. Extensive benchmarks have been carried out on IBM-SP2 and Origin2000 workstations. Significant speedup has been achieved in all cases. However, all test cases presented in their report are carried out on 2D unstructured meshes with relatively low computation and communication requirements.

Gopalaaswamy et al. (1997) presented a paper on parallelisation and dynamic load balancing of NPARC codes. In their work, several two- and three- dimensional flow cases are carried out to study the dynamic loading balance algorithm. To achieve loading on each processor during execution, they decompose the computing domain into more blocks than the number of machines available. In the initial phase, blocks and interfaces are allocated to processors on the basis of block and interface sizes, speed of individual processor and even network speed. In the dynamic load balancing phase, the execution and communication cost of every processor are gathered and analyzed to change the initial distribution of blocks. Good load balancing has been proved to be important to overall performance. However, the communication overhead caused by decomposing flow domains to more blocks than processor numbers has been ignored by authors. This may seriously affect the overall performance when communication volume becomes the bottleneck of performance.

A fully distributed unstructured Navier-Stokes solver for large scale aeroelasticity computations has been developed by Barakos et al. (2001). They presented the development and validation of a parallel unsteady flow and aeroelasticity code for large scale numerical models used in turbo machinery applications. MPI library is used in the parallelisation of their code to exploit the Single Process Multiple Data (SPMD) paradigm. METIS (Karypis and Kumar 1995) is used to decompose the computing meshes. Five test cases have been carried out on CRAY T3E, SGI Origin2000, clustered DEC alpha workstations, DELL Poweredge1500 and Networked LINUX PCs. In most of their test cases, the CRAY T3E is leading in the speedup comparison (speedup is about 23 using 32 processors) and networked LINUX PCs (speedup is about 3 at 4 processors) is the worst. These test cases also reveal that the communication cost is vital to parallel computing performance. Overall the parallel computing performance is very good. However, all test cases on networked LINUX PCs have carried out on only 4 processors. It is hard to assess the full strength of this kind of cluster system.

Mavriplis (2002) summarised his results obtained with the NSU3D unstructured multigrid solver for the AIAA Drag Prediction Workshop. In his report, most these cases were run in parallel on commodity cluster-type machines while the some were run on an SGI Origin machine using 128 processors. He found that inexpensive clusters of commodity computers are able to compute large numbers of cases using an unstructured solver.

It should be noticed that most of the research reviewed here are carried out on MPP machines, some with shared memory and attached processors. Therefore, many of their conclusions may not be applicable to a cluster system with a moderate number of nodes. The trend in parallel computing is moving away from supercomputers to cheaper, general purpose cluster systems consist of loosely coupled systems, such as PCs. The performance of parallel computing on middle-end cluster systems, especially on PC cluster systems, has not been fully investigated.

2.6 CFD on Wind Turbine Applications

Many countries in the world are strongly committed to increasing renewable energy usage in order to help reduce greenhouse gas emissions and thereby contribute to national targets for emissions reductions (Department of Trade and Industry 2002; Danish Wind Industry Association 2001). Renewable forms of energy are those continuously available sources which do not rely on exhaustible fossil fuels (e.g. coal, oil and gas). Wind is an ideal form of renewable energy, which is highly available. In the past, windmills were used to grind corn and pump water. Today, wind turbines use wind to generate electricity. Figure 2.3 illustrates a 2-blade wind structure. The speed of turbines is controlled by a yaw (14 in Figure 2.3) motor which turns the nacelle and rotor to face wind to gain maximum power.
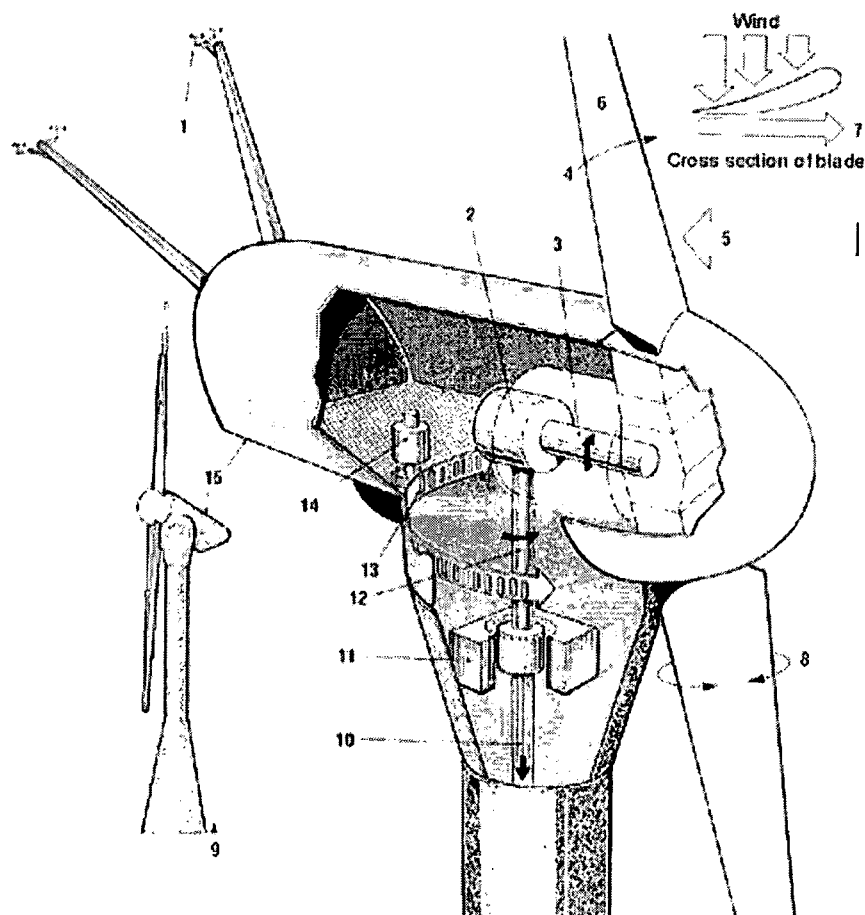
Figure 2.3 A modern wind turbine used in commercial wind farms

European countries such as Holland, Norway and France have been the world leaders in the design and manufacture of wind turbines due to their historical experience of several centuries, in building complex wind mill structures, which were used in water pumping, grain grinding and for lumbering. In other part of world, the use of wind energy has been limited mainly because power productivity from wind is too low and ~~unreadable struck-through text~~ wind turbines and improve the efficiency and reliability, better understanding of flows around the wind turbine is required.

It is nearly impossible to conduct wind tunnel tests because of the size of turbine (typical radius of a single blade is about 10m). Schepers at al. (1997) reported their early field test on a NREL Phase II HAWT blade. Pressure distributions on the blade surface on several spanwise positions are measured. This provides some validation data for CFD codes, but not enough for fully understanding the flows because of limited data available. Numerical simulation becomes a very important tool for optimal designing of the shape of the blade and the control system.

Kang (2001) and Duque et al. (1999; 2000) reported their Navier-Stokes simulation results of the NREL turbine. In their simulation, a multi-block structured-grid with more than one million grid points is used around a single blade. The numerical results show good agreement with experiment except near hub regions at low wind speed conditions.

2.7 Summary Comments

Significant progress has been made in the area of spatial and temporal discretisation, adaptive, multigrid and parallel algorithms. Unstructured grid technology is almost on par with structured grid technology. The areas that require further research include better and faster multigrid flow solvers that is sensitive to cell aspect ratio and grid stretching, improved adaptive method for viscous flow simulations, better mesh generation technique, and application of parallel computing technique to highly available computing platforms.

1. Solution adapted grid method is increasingly used to compute complex steady and unsteady flows. When adapting grids to flow features, a common problem that may occur is that certain regions are over resolved at the expense of other regions, especially in viscous flow simulations. This could serious affect the overall convergence as well as the solution accuracy.

2. It is well known that the grid quality is very important in order to achieve an accurate solution. Developing a general-purpose unstructured mesh generator for viscous flow computation is proved to be a difficult task. However, many unstructured mesh generators in public domain (Owen 1998) could produce high quality inviscid grid. Thus, the task remaining is to find a way to generate 2D/3D unstructured-grid for viscous computation utilising an inviscid mesh generator available in the public domain. The strategy in the current work is based on the "inflation" idea.

3. All efforts reviewed above provide encouraging evidence of the usefulness of the multigrid method, particularly for the simulation of viscous flows with high grid aspect ratio. In general, how to construct coarse mesh levels is the key for a robust and efficient multigrid method.

4. The parallel computing technique could offer massive reduction of computing time in the future. In all the efforts reviewed, significant development has been made toward efficient parallel computing on super computers with attached processors or high performance workstations. The objective of the current research is to explore the possibility of clustering highly available PCs. With slower network connection on most PC system, many conclusions drawn on high performance computing with parallel computers may need to be reconsidered, including loading balance, communication and partition schemes.

5. The unstructured-grid could offer a better alternative to the structured-grid method in renewable energy applications because of the flexibility in mesh

generation: use a fine grid near the blade and a very coarse grid in far from blade regions where the flow is largely uniform.

Chapter 3

Flow Models and Discretisation

This chapter presents the governing equations of the fluid motion and numerical discretisations for solving steady, compressible, inviscid/turbulent viscous flows on 2D/3D unstructured-grids.

The first part of the discussion is an introduction of the three dimensional Euler/Navier-Stokes equations. To simplify the equations, a nondimensionalisation procedure is described in some detail. Subsequently, turbulence modelling and a coupled solution procedure for numerical simulations of turbulent flows are presented.

In the next section, the spatial and time discretisations are discussed. The spatial discretisation, including two- and three- dimensional finite volume schemes, a spatial second order construction, and an upwind scheme, is described in detail. Next, the temporal discretisation based on a multistage Runge-Kutta approach is presented. This chapter is concluded with the presenting of physical and numerical boundary conditions.

3.1 Governing Equations

The governing equations for the three-dimensional compressible fluid motion are the time-dependent compressible Navier-Stokes equations. The equations are expressed as a system of conservation laws relating the rate of change of mass, momentum, and energy in a control volume: Ω. In an integral form, the equations are given as

$$\frac{\partial}{\partial t}\iiint_\Omega Q dV + \oiint_{\partial\Omega}F(Q)\cdot\hat{n}dS = \oiint_{\partial\Omega}G(Q)\cdot\hat{n}dS \tag{3-1}$$

Where V is the volume of Ω, $\hat{n}$ is the outward pointing normal to the control volume boundary $\partial\Omega$. The vector of dependent variables Q, inviscid flux vector F and viscous flux vector G are given as

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix} \quad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (E+p)u \end{bmatrix}\hat{i} + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho wv \\ (E+p)v \end{bmatrix}\hat{j} + \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (E+p)w \end{bmatrix}\hat{k} \tag{3-2}$$

$$G = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{bmatrix}\hat{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y \end{bmatrix}\hat{j} +$$

$$\begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \end{bmatrix}\hat{k} \tag{3-3}$$

In these equations, ρ is density, p is pressure, E is total energy per unit volume, u, v and w are the Cartesian components of velocity in x, y and z directions, respectively. τ_{ij} and q_i are shear stress and heat conduction terms. Apart from basic assumptions

39

of no chemical reaction, mass diffusion, or heat addition and the absence of body force, we have to make further assumptions to close the equations, because the shear stress and heat conduction terms are left undefined. To specify these terms, the fluid is assumed to be Newtonian fluid, known as Stokes' hypothesis. For Newtonian fluid, the shear stress is linearly related to the fluid strain rate and heat conduction is linearly dependent on temperature gradients, written as,

$$\tau_{xx} = \frac{2}{3}\mu_l (2u_x - v_y - w_z)$$

$$\tau_{yy} = \frac{2}{3}\mu_l (2v_y - u_x - w_z)$$

$$\tau_{yy} = \frac{2}{3}\mu_l (2w_z - v_y - u_x) \hspace{4cm} (3\text{-}4)$$

$$\tau_{xy} = \tau_{yx} = \frac{2}{3}\mu_l (v_x + u_y)$$

$$\tau_{xz} = \tau_{zx} = \frac{2}{3}\mu_l (w_x + u_z)$$

$$\tau_{yz} = \tau_{zy} = \frac{2}{3}\mu_l (v_z + w_y)$$

$$q_x = -\kappa \frac{\partial T}{\partial x}$$

$$q_y = -\kappa \frac{\partial T}{\partial y} \hspace{4cm} (3\text{-}5)$$

$$q_z = -\kappa \frac{\partial T}{\partial z}$$

Where T is temperature, κ is coefficient of thermal conductivity, μ is molecular viscosity and Pr is the Prandtl number.

In addition to the Stokes' hypothesis, the equations are closed with relations of state for perfect gas

$$p = (\gamma - 1)\left[E - \frac{1}{2}\rho(u^2 + v^2 + w^2) \right]$$

$$p = \rho RT \qquad\qquad\qquad\qquad (3\text{-}6)$$

Here R and γ are gas constants.

When perfect gas is assumed, the molecular viscosity is only related to temperature. The molecular viscosity is determined by Sutherland's law, as

$$\frac{\mu_l}{\mu_{ref}} = \frac{(T_{ref} + C^*)}{(T + C^*)}(\tfrac{T}{T_{ref}})^{\frac{3}{2}} \qquad\qquad (3\text{-}7)$$

Where C^* is a constant related to the gas property, μ_{ref} is the molecular viscosity at the reference temperature T_{ref}.

Rotational Terms

The governing equations previously defined can also be derived for a moving control volume in space, such as in turbomachinery applications. This can be done by adding extra rotational terms to the original equations.
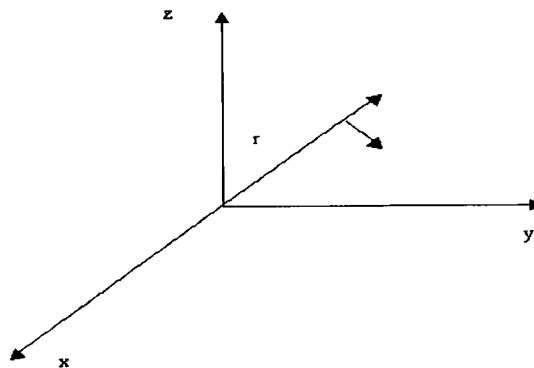
Figure 3.1 Rotating Cartesian coordinate

41

For a Cartesian coordinate system rotating at a constant speed ϖ around the x axis, as shown in Figure 3.1, the Navier-Stokes equations can be expressed as,

$$\frac{\partial}{\partial t}\iiint_\Omega Q dV + \oiint_{\partial\Omega} F \cdot \bar{n} dS = \oiint_{\partial\Omega} G \cdot \bar{n} dS + \iiint_\Omega S dV \qquad (3\text{-}8)$$

Where the Q is the state vector in a relative frame. The flux vectors F and G are identical to the non-rotating equations except that all the variables are in a relative frame. The source term S represents the centripetal and coriolis force. The vector of the source terms takes into accounts for the rotation of the frame is given as,

$$S = \begin{bmatrix} 0 \\ 0 \\ \rho\left(\varpi^2 y - 2\varpi w\right) \\ \rho\left(\varpi^2 z + 2\varpi v\right) \\ 0 \end{bmatrix} \qquad (3\text{-}9)$$

Here y and z denote the centroid of the element assuming the frame is rotating about the x axis. The total energy becomes,

$$E = \rho e + \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right) - \frac{1}{2}\rho\varpi^2 r^2 \qquad (3\text{-}10)$$

Where $r = \sqrt{y^2 + z^2}$ is the distance from the x axis.

The perfect gas relation becomes,

$$p = (\gamma - 1)\left[E - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right) + \frac{1}{2}\rho\varpi^2 r^2\right]$$

$$H = E + \frac{p}{\rho} = \frac{\gamma}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}\left(u^2 + v^2 + w^2\right) - \frac{1}{2}\varpi^2 r^2 \qquad (3\text{-}11)$$

The algorithm developed in this thesis can be used to solve the flow field associated with a non-rotating frame and a rotating frame in its own Cartesian coordinate system. The algorithm for a rotating frame should return to a non-rotating form when the rotation speed is zero.

3.2 Non-Dimensionalisation

Non-dimensionalisation of the above equations is useful because the resulting equations are easy to understand. Although the selection of reference could be different for internal and external flows, the number of reference variables is the same. In the present study, all dependent variables and given conditions are non-dimensionalised by a reference length (L_{ref}), sound speed (a_{ref}), density (ρ_{ref}), molecular viscosity (μ_{ref}), temperature (T_{ref}), and heat conduction coefficient (k_{ref}).

$$x_i = \frac{\overline{x}_i}{L_{ref}}, \qquad t = \frac{\overline{t}}{L_{ref}/a_{ref}}, \qquad \rho = \frac{\overline{\rho}}{\rho_{ref}}, \qquad u_i = \frac{\overline{u}_i}{a_{ref}},$$

$$p = \frac{\overline{p}}{a_{ref}^2 \rho_{ref}} \qquad E = \frac{\overline{E}}{a_{ref}^2 \rho_{ref}}, \qquad T = \frac{\overline{T}}{T_{ref}}, \qquad \mu = \frac{\overline{\mu}}{\mu_{ref}},$$

$$(3\text{-}12)$$

For shear stress and heat transfer terms, these terms are non-dimensionalised according to their definition:

$$\tau = \frac{\overline{\tau}}{\mu_{ref} a_{ref}/L_{ref}} \qquad\qquad q = \frac{\overline{q}}{\mu_{ref} a_{ref}^2/L_{ref}} \qquad (3\text{-}13)$$

In addition, the state constants and relations become,

$$R = \frac{1}{\gamma}, \qquad p = \frac{\rho T}{\gamma}, \qquad a = \sqrt{T}$$

$$E = \frac{p}{\gamma-1} + \frac{1}{2}\rho(u^2 + v^2 + w^2) \qquad\qquad (3\text{-}14)$$

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

With this procedure, the form of the non-dimensional equations becomes identical to the dimensional one except an extra term Re before the viscous term.

$$\frac{\partial}{\partial t} \iiint_{\Omega} Q dV + \oiint_{\partial\Omega} F \cdot \bar{n} dS = \frac{1}{\text{Re}} \oiint_{\partial\Omega} G \cdot \bar{n} dS \qquad (3\text{-}15)$$

$$\text{Re} = \frac{\rho_{ref} a_{ref} L_{ref}}{\mu_{ref}} \qquad (3\text{-}16)$$

3.3 Turbulence Modelling

One important phenomenon for complex viscous flows is turbulence, which is very difficult to simulate due to the existence of a wide range of scales. There are many types of methods to deal with turbulence, ranging from the simplest algebraic model to the more accurate Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS). While the LES and DNS methods are more accurate than other turbulence models, they require prohibitive amount of computing power. This is especially true for CFD codes based on the unstructured-grid method. Therefore, the most widely used way to deal with turbulence flow problems in industrial applications is still turbulence modelling. For most of turbulence models, the Reynolds stresses are assumed to be related to the mean strain rate by the eddy viscosity. Such models may be classified as zero-equation model, one-equation model, and two-equation model, depending on the number of transport equations needed to be solved to obtain the eddy viscosity.

The closure of the Navier-Stokes equations requires the definition of turbulent Reynolds stress. For eddy viscosity models, the stress tensor is modelled as proportional to the mean strain-rate tensor.

$$\tau_{ij} = \tau_{lij} + \tau_{tij}$$

$$\tau_{lij} = 2\mu_l (S_{ij} - S_{nn}\delta_{ij}/3)$$

$$\tau_{tij} = 2\mu_t (S_{ij} - S_{nn}\delta_{ij}/3) - 2\rho k\delta_{ij}/3 \qquad (3\text{-}17)$$

44

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

Where,

$$\delta_{ij} = \begin{cases} = 1 & while \quad i = j \\ = 0 & while \quad i \neq j \end{cases}$$

Heat conduction terms

$$q_i = q_{li} + q_{ti} = -\frac{1}{\gamma - 1}\left(\frac{\mu_l}{\mathrm{Pr}_l} + \frac{\mu_t}{\mathrm{Pr}_t} \right)\frac{\partial T}{\partial x} \qquad (3\text{-}18)$$

3.3.1 One-Equation Spalart-Allmaras Model

The Spalart-Allmaras model is an eddy viscosity model based on a transport equation for the turbulent viscosity. It was inspired from an earlier model: Baldwin-Barth one equation model (Balwin and Barth 1991). The formulation and coefficients are defined based on dimensional analysis, Galilean invariance, and some selected empirical results. The empirical results used in its development are two-dimensional mixing layers, wakes, and flat-plate boundary layer flows.

The aim of this model is to improve the predictions obtained with algebraic mixing-length models and to provide an alternative to two-equation models. Furthermore, this model is easier to adapt to an unstructured-grid method compared to an zero equation model such as Balwin-Lomax model (Balwin and Lomax 1991), because it requires only the distance from the wall to work out the eddy viscosity at a grid point rather than integration along lines normal to the wall (Balwin and Lomax 1991).

In the Spalart-Allmaras model (Spalart and Allmaras 1992), the eddy viscosity function is defined in terms of a dependent variable, $\tilde{v}$, and a wall function, f_{v1}, as following:

$$\mu_t = \rho\tilde{v}f_{v1} \qquad (3\text{-}19)$$

In zones far from wall, the wall function equals 1.

The convective transport equation of the eddy viscosity is modelled as

$$\frac{\partial \rho \tilde{v}}{\partial t} + \frac{\partial \rho u \tilde{v}}{\partial x} + \frac{\partial \rho v \tilde{v}}{\partial y} + \frac{\partial \rho w \tilde{v}}{\partial z} = C_{b1}(1 - f_{v2})\rho \tilde{S} \tilde{v} +$$

$$\frac{1}{\sigma}[\frac{\partial}{\partial x}\left(\rho(\upsilon_l + \upsilon_t)\frac{\partial \tilde{v}}{\partial x}\right) + \frac{\partial}{\partial y}\left(\rho(\upsilon_l + \upsilon_t)\frac{\partial \tilde{v}}{\partial y}\right) + \frac{\partial}{\partial z}\left(\rho(\upsilon_l + \upsilon_t)\frac{\partial \tilde{v}}{\partial z}\right)$$

$$+ C_{b2}\rho\left(\frac{\partial \tilde{v}}{\partial x}\frac{\partial \tilde{v}}{\partial x} + \frac{\partial \tilde{v}}{\partial y}\frac{\partial \tilde{v}}{\partial y}\right)] - \left[C_{w1}f_w - \frac{c_{b1}}{K^2}f_{trip2}\right]\rho\left[\frac{\tilde{v}}{d}\right] + f_{trip1}\rho\Delta U^2$$

$$(3\text{-}20)$$

Where the right-hand-side terms represent the turbulence eddy viscosity production, conservative diffusion, near wall turbulence destruction, transition damping of production, and transition source of turbulence.

The model constants for free-shear flows to control the production and diffusion of turbulent eddy viscosity are

$$C_{b1} = 0.1355 \quad C_{b2} = 0.622 \quad \sigma = 2/3 \qquad\qquad (3\text{-}21)$$

The additional constants and auxiliary functions for destruction of turbulent eddy viscosity in the boundary layer are

$$C_{w1} = C_{b1}/\kappa^2 + (1 + C_{b2})/\sigma \qquad r \equiv \frac{\tilde{v}}{\tilde{S}\kappa^2 d^2} \qquad C_{w2} = 0.3$$

$$g = r + C_{w2}\left(r^6 - r\right) \qquad\qquad f_w = g\left(\frac{1 + C_{w3}^6}{g^6 + C_{w3}^6}\right)^6 \quad C_{w3} = 2 \qquad (3\text{-}22)$$

The auxiliary functions for near wall regions are given as

$$\tilde{S} = S + \frac{\tilde{v}}{(\kappa d)^2}f_{v2} \qquad\qquad S = \sqrt{2S_{ij}S_{ij}} \qquad\qquad \chi \equiv \frac{\tilde{v}}{v_l}$$

$$f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3} \qquad\qquad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \qquad C_{v1} = 7.1 \qquad\qquad (3\text{-}23)$$

The auxiliary functions to control the laminar region of the shear layers and transition to turbulence are defined as:

$$f_{trip1} = C_{trip1} g_{trip} \cdot \exp\left(-C_{trip2} \frac{\omega_{trip}^2}{\Delta U^2} \left[d^2 + \left(g_{trip} d_{trip}\right)^2 \right] \right)$$

$$f_{trip2} = C_{trip3} \cdot \exp\left(-C_{trip4} \chi^2 \right)$$

$$g_{trip} = \min\left(0.1, \Delta U / (\omega_{trip} \Delta x_{trip}) \right)$$

$$C_{trip1} = 1.0 \qquad C_{trip2} = 2.0 \qquad C_{trip3} = 1.2 \qquad C_{trip14} = 0.5 \qquad\qquad (3\text{-}24)$$

Where ω_{trip} is the vorticity at the boundary trip point, ΔU is the norm of difference between velocity at a field point and the velocity at the trip point, Δx_{trip} is the grid spacing along the wall at the trip, and d is distance from the wall.

Boundary conditions must be supplied for the turbulent model. Ideally, when the mesh near the wall is fine enough (y^+ <3), the eddy viscosity on solid wall should be set to zero. In most high Reynolds number flow cases, a wall function has to be used in near wall regions to provide enough resolution. In this case, the eddy viscosity on a wall surface is given by the wall function relation, as we will discuss in the boundary condition section. For the inlet far from wall and without incoming distortion, the eddy viscosity should be set to zero too, but for numerical reasons, a very small value of the dependent variable ($\tilde{v} < 10^{-6}$) should be used as the inlet condition (Spalart and Allmaras 1992), which implies a very small value for the eddy viscosity. For an outlet, a simple extrapolation is used to transport information from the computational domain to outside.

3.3.2 Coupled Solution Method

The solution procedure for the turbulence transport equation is similar to that for solving the Navier-Stokes equations, so it is convenient to couple the turbulence transport equation with the main flow equations. There are advantages and disadvantages of using this approach. There are two main reasons for a coupled method.

- Coding is relatively easier with this coupled method, especially when more than one turbulence models are adopted in a CFD code. Turbulence dependent variables can be aligned with other state variables using a vector.

- Turbulence dependent variables can use the same second spatial order method, upwind scheme and diffusion term treatment in a single vector.

This makes the code slightly more efficient than a decoupling method. Because the turbulent model equations are tightly coupled with the main equations, the time discretisation is an explicit multistage Runge-Kutta scheme, the same as that for the main equations, and the same time step has been used. Therefore, the turbulent model equations are tightly coupled with the main equations.

After coupling the turbulence equations with the main equations, the governing equations become:

$$\frac{\partial}{\partial t} \iiint_{\Omega} Q dV + \oiint_{\partial\Omega} F(Q) \cdot \hat{n} dS = \frac{1}{Re} \oiint_{\partial\Omega} G(Q) \cdot \hat{n} dS + \iiint_{\Omega} R dV \qquad (3\text{-}25)$$

The state variable vector Q, inviscid flux vector F and viscous vector G are

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \\ \rho V1 \\ \vdots \\ \rho Vn \end{bmatrix} \qquad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (E+p)u \\ \rho uV1 \\ \vdots \\ \rho uVn \end{bmatrix} \hat{i} + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho wv \\ (E+p)v \\ \rho vV1 \\ \vdots \\ \rho vVn \end{bmatrix} \hat{j} + \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (E+p)w \\ \rho wV1 \\ \vdots \\ \rho wVn \end{bmatrix} \hat{k}$$

$$G = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \\ Q_1 \dfrac{\partial V1}{\partial x} \\ \vdots \\ Q_n \dfrac{\partial Vn}{\partial x} \end{bmatrix} \hat{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y \\ Q_1 \dfrac{\partial V1}{\partial y} \\ \vdots \\ Q_n \dfrac{\partial Vn}{\partial y} \end{bmatrix} \hat{j}$$

$$+ \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \\ Q_1 \dfrac{\partial V1}{\partial z} \\ \vdots \\ Q_n \dfrac{\partial Vn}{\partial z} \end{bmatrix} \hat{k} \qquad\qquad (3\text{-}26)$$

Where n is the number of turbulence transport equation numbers, and Q_n and V_n are constants and dependent variables relating to a specific turbulence model. R is the source term vector.

3.4 Spatial Discretisation

The discretisation of the governing equations follows an approach, which decouples the spatial discretisation and temporal discretisation. Discretisation of the solution domain produces a computational mesh on which the governing equations are solved. In the current study, the spatial discretisation is carried out based on a Finite Volume Method (FVM). The name of finite volume method comes from the technique that the integral formulation of conservation laws. For the unstructured-grid method, it is natural to discretise the computational domain using a finite volume scheme.

3.4.1 Two-Dimensional Finite Volume Method

In the current research, unstructured meshes in two dimensions consist of only triangular elements (Figure 3.2), which are non-overlapping with each other and fully cover a computational domain. As previous stated, the control volume is cell-centred based, i.e. all the flow variables are stored in centre of the triangle, and flux evaluation occurs on the three edges which define the triangle.
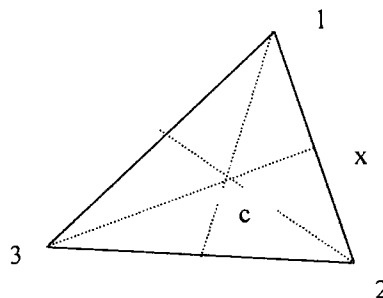

Figure 3.2 2D control volume: triangle

The numerical evaluation of the surface integrals in conservative equations is done separately for the inviscid and viscous contributions. For a finite volume formulation, the inviscid contribution can be approximated using midpoint integration of the flux over each edge of the triangle that defines the control volume.

$$\oint_{\partial\Omega} \vec{F} \cdot \hat{n} dl = \oint_{\partial\Omega} \vec{F} \cdot d\vec{L} \approx \sum_{i=1}^{3} \Phi(Q^+, Q^-, \hat{n}_i) \cdot l_i \qquad (3\text{-}27)$$

Here l is the length of the edge, which two neighboured triangles are attached together, $\Phi(Q^+, Q^-, \hat{n}_i)$ is the numerical flux calculated from the states of the left side Q^+ and the right side Q^- of the edge, which is determined by some kind of interpolation from the left and right side cells of the interface.

3.4.2 Three-Dimensional Finite Volume Method

In three-dimensions, a computational domain is normally discretised by tetrahedral control volumes (Figure 3.3), with four points forming the volume and every three out of the points forming four faces. This kind of control volume has the ability of rapidly increasing control volume size to cover the whole domain for accurate and economical computations.
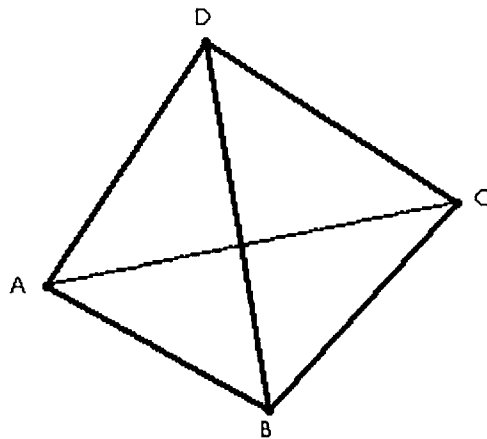
Figure 3.3 3D control volume: tetrahedron

When computing viscous flows on unstructured meshes, it is inevitable that highly stretched grids are required in order to resolve boundary layer near solid walls. This is because of gradients in the normal direction to the solid wall being several orders higher compared with gradients along the wall direction. We find that using layers of prismatic control volumes (Figure 3.4) in the viscous effect dominated regions, such

51

as near a solid wall, wake, etc, can immensely improve accuracy of the solution and effectively reduce the difficulties of viscous mesh generation. For a prismatic control volume as shown in Figure 3.4, the bottom and top face consist of three points and three side faces are formed by four points that are not necessarily coplanar.
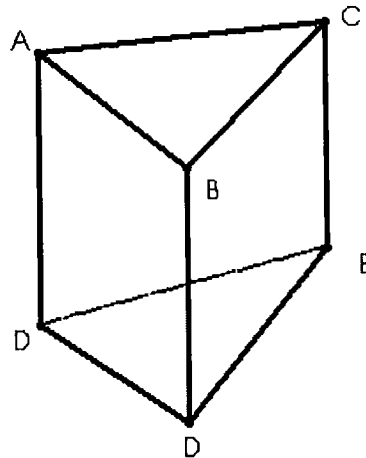
Figure 3.4 3D control volume: prism

3.4.3 Fluxes Evaluation

The fluxes evaluation procedure on a finite volume method is simply summing up contribution of the fluxes on each face of the control volume. The procedure can be described as

$$\iint_{\partial\Omega} \vec{F} \cdot \hat{n} ds = \iint_{\partial\Omega} \vec{F} \cdot d\vec{s} \approx \sum_{i=1}^{k} \Phi(Q^+, Q^-, \hat{n}_i) \cdot s_i \qquad (3\text{-}28)$$

Here s and $\hat{n}$ are the area and out-pointing unit vector of the face. k is the number of faces which define the control volume. $\Phi(Q^+, Q^-, \hat{n}_i)$ is the numerical flux calculated from the state variables of the left side Q^+ and the right side Q^- of the face.

In early 1980s, researchers extended much of what was established for the traditional structured method to unstructured meshes (Jameson and Mavriplis 1986; Jameson, et

al. 1986; Jameson et al. 1981). That includes the using of central-difference approximation of inviscid fluxes and dissipation terms (Jameson and Mavriplis 1986). The use of an upwind scheme offers several advantages over a central-difference formulation. An upwind scheme is a characteristic based method; it has natural dissipation terms, while spatial dissipation terms have to be added to central-difference schemes for stability reasons. For high Reynolds number flows, highly stretched grids are required to resolve the high gradient in near wall regions. With a central difference scheme, excessive dissipation may be introduced in the flow direction (Kunz and Lakshminarayana 1992), and may seriously reduce the accuracy and convergence rate. With an upwind scheme, the resolution of boundary layer details typically requires only half as many points as with a central-difference code (Zheng and He 2001). Furthermore, the poor performance of central difference formulation is attributed to the artificial dissipation formulas commonly used to damp odd-even oscillations and to provide non-linear stability (Anderson and Bonhaus 1994).

Upwind schemes are categorised as either FDS (Flux Difference Splitting) or FVS (Flux Vector Splitting). For the current study, a flux difference splitting scheme is used for computing the inviscid contribution to the fluxes.

Roe (Roe 1981) FDS scheme is the family of Flux Difference Splitting schemes that use the approximate Riemann solution. His scheme is one of the most popular methods among the FDS schemes because of its accuracy and efficiency.

For Roe's flux difference splitting scheme, the flux is given as a central difference term in addition to dissipation terms,

$$\Phi(Q_L, Q_R) = \frac{1}{2}(\Phi(Q_L) + \Phi(Q_R)) - \frac{1}{2}\sum_{k=1}^{4}|\bar{a}_k|^* \Delta V_k \hat{R}_k \qquad (3\text{-}29)$$

And the second term can be written as

$$\frac{1}{2}\sum_{k=1}^{4}|\bar{a}_k|^* \Delta V_k \hat{R}_k = \left|\Delta \vec{F}_{\bar{U}}\right| + \left|\Delta \vec{F}_{\bar{U}+a}\right| + \left|\Delta \vec{F}_{\bar{U}-a}\right| \qquad (3\text{-}30)$$

All the three terms can be written as

$$\left|\Delta \vec{F}_{\bar{U}}\right| = \left|\overline{U}\right| \left\{ \left(\Delta\rho - \frac{\Delta p}{a^2}\right) \begin{bmatrix} 1 \\ \hat{u} \\ \hat{v} \\ \hat{w} \\ \dfrac{\hat{u}^2 + \hat{v}^2 + \hat{w}^2}{2} \end{bmatrix} + \hat{\rho} \begin{bmatrix} 0 \\ \Delta u - \hat{n}_x \Delta U \\ \Delta v - \hat{n}_y \Delta U \\ \Delta w - \hat{n}_z \Delta U \\ \hat{u}\Delta u + \hat{v}\Delta v + \hat{w}\Delta w - \hat{U}\Delta U \end{bmatrix} \right\}$$

$$\left|\Delta \vec{F}_{\bar{U}+a}\right| = \left|\overline{U} + a\right| \cdot \left\{ \frac{\Delta p + \hat{\rho}a\Delta U}{2a^2} \begin{bmatrix} 1 \\ \hat{u} + \hat{n}_x a \\ \hat{v} + \hat{n}_y a \\ \hat{w} + \hat{n}_z a \\ \hat{h} + \hat{U}a \end{bmatrix} \right\} \qquad (3\text{-}31)$$

$$\left|\Delta \vec{F}_{\bar{U}-a}\right| = \left|\overline{U} - a\right| \cdot \left\{ \frac{\Delta p - \hat{\rho}a\Delta U}{2a^2} \begin{bmatrix} 1 \\ \hat{u} - \hat{n}_x a \\ \hat{v} - \hat{n}_y a \\ \hat{w} - \hat{n}_z a \\ \hat{h} - \hat{U}a \end{bmatrix} \right\}$$

Where the $\Delta(\ )$ represents the jump between the left and the right states

$$\Delta(\ ) = (\ )_R - (\ )_L \qquad (3\text{-}32)$$

And the $(\wedge)$ quantities are the Roe-averaged variables which can be work out as:

$$\hat{\rho} = \sqrt{\rho_L \rho_R}$$

$$\hat{u} = \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

$$\hat{v} = \frac{\sqrt{\rho_L}v_L + \sqrt{\rho_R}v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \qquad (3\text{-}33)$$

$$\hat{w} = \frac{\sqrt{\rho_L} w_L + \sqrt{\rho_R} w_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

$$\hat{h} = \frac{\sqrt{\rho_L} h_L + \sqrt{\rho_R} h_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

Where a, $\hat{u}_n$ and $\hat{u}_t$ are calculated directly from $\hat{\rho}$, $\hat{u}$, $\hat{v}$ and $\hat{h}$, so

$$\hat{p} = \frac{(\gamma - 1)\hat{\rho}}{\gamma}\left[\hat{h} - \frac{1}{2}\left(\hat{u}^2 + \hat{v}^2 + \hat{w}^2\right)\right]$$

$$a = \sqrt{\frac{\gamma \hat{p}}{\hat{\rho}}}$$

$$\hat{U} = \hat{u} \cdot \hat{n}_x + \hat{v} \cdot \hat{n}_y + \hat{w} \cdot \hat{n}_z$$

$$\Delta U = \Delta u \cdot \hat{n}_x + \Delta v \cdot \hat{n}_y + \Delta w \cdot \hat{n}_z \qquad (3\text{-}34)$$

Roe FDS introduces expansion shock waves that are physically unacceptable. To prevent the expansion shocks, an entropy fix is imposed. A smoothed value of $\left|\hat{a}_k\right|$ is defined for the acoustic waves

$$\left|\hat{a}_k\right|^* = \begin{cases} \left|\hat{a}_k\right| & \left|\hat{a}_k\right| \geq \frac{1}{2}\delta a_k \\ \dfrac{\hat{a}_k^{\,2}}{\delta a_k} + \dfrac{1}{4}\delta a_k & \left|\hat{a}_k\right| < \frac{1}{2}\delta a_k \end{cases} \qquad (3\text{-}35)$$

With

$$\delta a_k = \max\left(4\Delta a_k, 0\right)$$

This provides a parabolic curve where the wave speeds change signs.

3.4.4 Higher-Order Scheme

In the current framework of a cell-centred finite volume scheme, the computational domain has been spatially discretised into an amount of control volumes, triangles in 2D and tetrahedron or prism in 3D. The challenge of constructing an effective higher order scheme is to determine an accurate estimation of the state variables at either sides of a cell faces for flux evaluation.

First Order Scheme

A first spatial order scheme on unstructured meshes is described in this section to introduce the concept of construction a second order scheme. Figure 3.5 shows a typical two-dimensional layout of an unstructured mesh. For a first order accurate approximation, the left and right side value Q^+ and Q^- are simply set to equal the cell-centred values of the left and right cells. This means that a constant distribution is assumed in each cell. This scheme is highly dissipative because of the constant solution assumption.
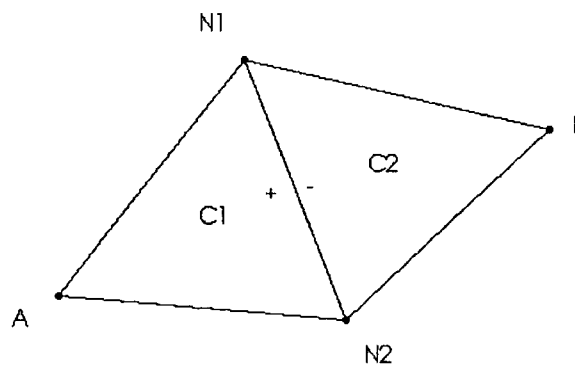

Figure 3.5 Fluxes across a cell interface

Second Order Construction

For a cell-centred finite volume setting, a second-order accuracy can be obtained by expanding the cell-centred states to each cell face with a Taylor series (Frink et al. 1991), as following,

$$q|(x,y,z) = q_c + \nabla q_c \cdot \Delta r + O(\Delta r^2) \tag{3-36}$$

Where $q = \begin{bmatrix} \rho \\ u \\ v \\ w \\ p \end{bmatrix}$ and q_c represents solution at the centre of an element.

In this formulation, the solution gradient in the cell centre ∇q_c is required. The gradient can be achieved by using a midpoint integral of the surface around the cell (Barth and Jesperson 1989; Wood and Kleb 1998), as follows,

$$\nabla q_c = \frac{1}{V} \oiint_{\partial\Omega} q\hat{n}dS \tag{3-37}$$

Where V is the volume, q is the solution on the cell surface and $\hat{n}$ denotes the surface unit vector.

A new second order scheme (Barth and Jesperson 1989) for a cell-centred setting proposed by Barth and Jesperson with a version of multidimensional linear reconstruction approach, which forms the basis for the present scheme. This method is based on the simple thinking that the reconstructed distribution in a control volume should be bounded by the values of its neighbours. Frink (1991) took the initiative to further simplify the method so the solution gradients need not be evaluated explicitly.

His simplification exploits the geometrical invariant of triangular and tetrahedral cells, which the distance of a cell-vertex to a cell-centroid is always two-thirds (for a triangle) or three-fourths (for tetrahedron) of the length from cell-vertex to the

opposing face. By using these invariants, the solution gradient for the cell $C1$ (Figure 3.5) along the linear extending of from a cell-vertex through cell-centroid to the opposing face can be written as,

$$\nabla q_L \approx \frac{1}{3\Delta r}\left[\frac{1}{2}(q_{n1}+q_{n2})-q_A\right]$$
(3-38)

Here Δr is the distance from the centre of an interface to centre of the element.

In 3D, the formula for a tetrahedral cell becomes,

$$\nabla q_L \approx \frac{1}{4\Delta r}\left[\frac{1}{3}(q_{n1}+q_{n2}+q_{n3})-q_A\right]$$
(3-39)

Where subscripts $n1$, $n2$ and $n3$ denote the nodes composing the face of a cell, and A is the opposing node. This resulting scheme is a second-order scheme and proven to be two times faster than original scheme presented by Barth (Barth and Jesperson 1989; Frink et al. 1991).

Weighted Averaging

In the second order implementation described previously, the nodal quantities are required to construct a second order scheme on a cell-centred finite volume setting. For a typical cell-centred finite volume scheme, the flow variables are stored at the geometrical centroid of each element. An averaging process is required to determine the solution on the nodes. A widely used weighted averaging is to distribute the solution from a cell centre to nodes according to their distance to the centroid of the cell. The following formula gives a simple formulation to extrapolate node quantities from a cell-centred solution,

$$q_n = \frac{\displaystyle\sum_{i=1}^{N}\frac{q_{c,i}}{r_{i,c}}}{b} \quad , \qquad b = \sum_{i=1}^{N} r_{i,c}$$

$$r_{i,c} = \sqrt{(x_n - x_c)^2 + (y_n - y_c)^2 + (z_n - z_c)^2} \qquad (3\text{-}40)$$

Here, N denotes the number of cells surrounding node n and r represents the distance from this node to the centroid of a cell.

3.4.5 Discretisation of Viscous Fluxes

The discretisation of viscous fluxes requires the first derivatives at the centroid of faces (3D) or the midpoint of edges (2D). The viscous fluxes are approximated at centroids of faces by a linear reconstruction which provides a continuous representation of solution variables across the face.

A widely used scheme is to apply the mid-point trapezoidal rule to evaluate the surface integral over edges (2D) or faces (3D) composing the cell (Sbardella and Imregun 2000; Barth and Jesperson 1989; Frink 1994). In 2D form (Figure 3.6), the procedure can be described by the following formula,

$$\nabla q_c = \frac{1}{A} \int_{\partial\Omega} q \cdot \vec{n} dl$$

$$= \frac{1}{2A} \sum_i^3 (q_c + q_{Ci}) \cdot \hat{n} L_i \qquad (3\text{-}41)$$

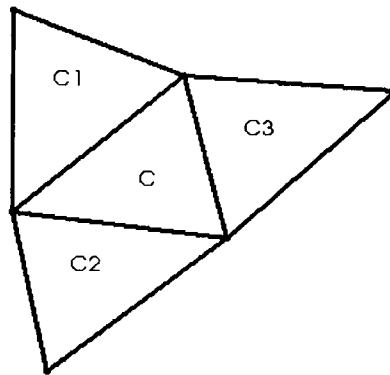Where q represents any of the state variables and A is the area of the cell C.

Figure 3.6 Gradients valuation of the cell C using surrounding cells

After the gradient at each cell centre is known, the gradient on the interface of two cells can be obtained by averaging the gradients in these two cells. The main drawback of this method is the additional dissipation introduced by the midpoint integration of the edges that compose the cell.

Another version of an interface gradient evaluation scheme exploits a stencil presented by Mitchell (1994), which is widely used by many other researchers (Frink and Pirzadeh 1998; Frink 1996; Zheng and He 2001). The stencil utilises the solution quantities on the nodes, which is composing of the interface, and cell-centred values of two cells sharing the interface.
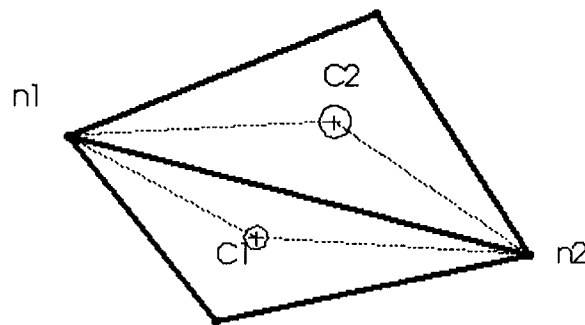
Figure 3.7 Evaluation of first derivatives on 2D triangular cells

In a two-dimensional setting, shown in Figure 3.7, the first derivatives for $q \equiv \begin{bmatrix} \rho & u & v & T \end{bmatrix}$ are derived from a Cramer's rule (Frink and Pirzadeh 1998) solution to

$$\begin{bmatrix} x_{c2} - x_{c1} & y_{c2} - y_{c1} \\ x_{n2} - x_{n1} & y_{n2} - y_{n1} \end{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix} = \begin{bmatrix} q_{c2} - q_{c1} \\ q_{n2} - q_{n1} \end{bmatrix}$$

(3.42)
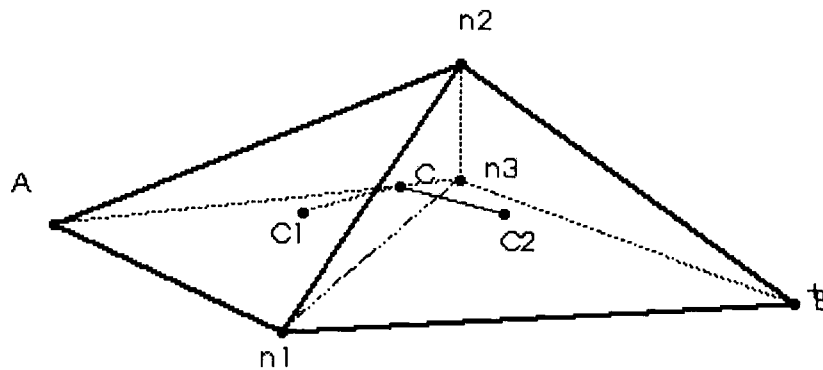
Figure 3.8 Evaluation of first derivatives on cells sharing a triangular interface

In a three-dimensional setting with two cells (tetrahedron or prism) sharing a common triangle interface, as show in Figure 3.8, first derivatives for $q \equiv \begin{bmatrix} \rho & u & v & w & T \end{bmatrix}$ are derived as,

$$
\begin{bmatrix}
x_{c2} - x_{c1} & y_{c2} - y_{c1} & z_{c2} - z_{c1} \\
\frac{1}{2}(x_{n2} + x_{n3}) - x_{n1} & \frac{1}{2}(y_{n2} + y_{n3}) - y_{n1} & \frac{1}{2}(z_{n2} + z_{n3}) - z_{n1} \\
\frac{1}{2}(x_{n1} + x_{n3}) - x_{n2} & \frac{1}{2}(y_{n1} + y_{n3}) - y_{n2} & \frac{1}{2}(z_{n1} + z_{n3}) - z_{n2}
\end{bmatrix}
\begin{bmatrix}
q_x \\
q_y \\
q_z
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
q_{c2} - q_{c1} \\
\frac{1}{2}(q_{n2} + u_{n3}) - q_{n1} \\
\frac{1}{2}(q_{n1} + q_{n3}) - q_{n2}
\end{bmatrix}
$$

(3-43)

When two prismatic cells share a quadrangular interface (Figure 3.9), the scheme has to be modified to result in accurate representation of gradients on the interface. In a quadrangular interface setting, the flow states on the four nodes composing the interface and cell centre values at either side of the interface are used to compute the gradients, as following,

$$\begin{bmatrix} x_{c2} - x_{c1} & y_{c2} - y_{c1} & z_{c2} - z_{c1} \\ x_{n3} - x_{n1} & y_{n3} - y_{n1} & z_{n3} - z_{n1} \\ x_{n2} - x_{n4} & y_{n2} - y_{n4} & z_{n2} - z_{n4} \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} (q_{c2} - q_{c1}) \\ (q_{n3} - q_{n1}) \\ (q_{n2} - q_{n4}) \end{bmatrix} \qquad (3\text{-}44)$$
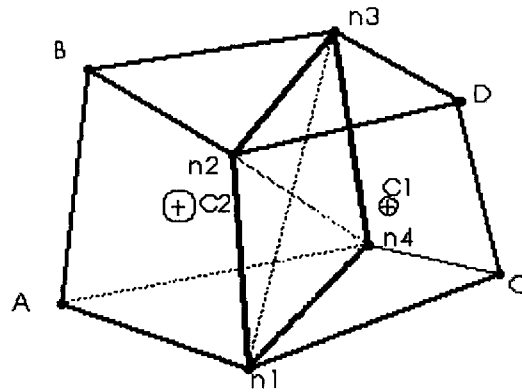
Figure 3.9 Evaluation of first derivatives on cells sharing a quadrangular interface

3.5 Time Discretisation

The 2D/3D unstructured flow solvers used in the present study incorporate a standard multi-stage Runge-Kutta scheme, as first introduced by Jameson et al (1985). Some researchers (Mavriplis and Venkatakrishnan 1995) report implicit schemes for solving two- and three- dimensional compressible Navier-Stokes equations on unstructured meshes. However, the draw back of the implicit scheme is the amount of memory required in three-dimensional cases.

For simplicity, the Navier-Stokes equations can be written in following form

$$\frac{\partial Q}{\partial t} = W(Q) \qquad (3\text{-}45)$$

Where W represents the numerical flux vector consisting of the inviscid and viscous fluxes. In the current research, three-stage and four-stage Runge-Kutta explicit scheme can be applied to integrate the above set of equations in time. A k-stage time stepping scheme with time step Δt would be

$$Q^{(0)} = Q^n$$

$$Q^{(1)} = Q^n + a_1 \Delta t \cdot W(Q^{(0)})$$

$$Q^{(2)} = Q^n + a_2 \Delta t \cdot W(Q^{(1)}) \qquad \text{(3-46)}$$

$$Q^{(3)} = Q^n + a_3 \Delta t \cdot W(Q^{(2)})$$

$$\vdots$$

$$Q^{(k)} = Q^n + a_k \Delta t \cdot W(Q^{(k-1)})$$

$$Q^{n+1} = Q^{(4)}$$

Many such schemes are possible and stable for range of time steps. Two widely used schemes are:

K=3: $a_1 = 0.3$, $a_2 = 0.5, a_3 = 1.0$ $\qquad$ (3-47)

K=4: $a_1 = 0.083$, $a_2 = 0.2069$, $a_3 = 0.4265$, $a_4 = 1.0$ $\qquad$ (3-48)

In order to accelerate a solution to steady state, a local time stepping technique is adopted:

$$\Delta t = CFL \cdot \frac{V}{(u+c) \cdot S_x + (v+c) \cdot S_y + (w+c) \cdot S_z} \qquad \text{(3-49)}$$

Where CFL ($< 2\sqrt{2}$) is the Courant-Friedrichs-Levy number; V is the volume of the element; S_x, S_y and S_z are area of the element projected on y-z, x-z and x-y planes, which are computed prior to the time integration process.

For turbulent flow simulations, the transport equations usually contain non-linear production and destruction terms, which can be very large near the solid wall regions. Such terms can severely reduce the convergence rate using a pure explicit scheme. Kunz and Lakshiminarayana (1992) recommended a quarter of the stable mean flow

timestep for turbulence transport equations in high Reynolds number flows to reduce the restriction of the CFL number to the mean flows by the turbulence transport equations.

3.6 Boundary Conditions

In numerical simulations, the flow domain has to be truncated for efficiency reasons to a finite field which includes the flow phenomena of most interest. The truncated domain edge is often referred as a boundary. On the boundary, appropriate boundary conditions have to be applied to produce a physically relevant solution. For the physical problems presented in the current research, there are two different types of flows: internal flows (or bounded flows) and external flows.

For an internal flow problem, the flow is often bounded by solid walls. No mass flux through solid walls is assumed. In the current research, only stationary and adiabatic walls (no heat flux across the wall) are considered. Other boundary conditions involved in the current internal flow problems are inflow and outflow boundaries. The number of physical boundary conditions to be imposed at these boundaries is determined by characteristic properties of the flow.

For an external flow problem, a farfield boundary is always present. The farfield boundary is often put far away from a wall or body, where the flow approaches some known uniform conditions.

All boundary conditions are applied at each stage of the Runge-Kutta time integration, prior to the calculation of fluxes and evaluation of residues. The boundary conditions for internal/external aerodynamics flows often include inflow, outflow, free stream, solid wall, and periodic conditions.

The cell-centred finite volume scheme requires an implementation of "ghost" cells for the boundary condition treatment. A ghost cell is produced by constructing an image cell across the boundary adjacent the interior cell when the boundary is NOT a block

interface, as shown in Figure 3.10. The flow state variables and derivatives are stored in the ghost cell for computing the flux across a boundary face.
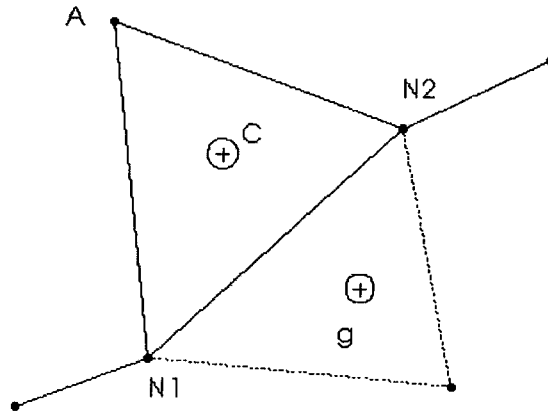
Figure 3.10 Ghost cells for boundary condition treatment

When a boundary is a block interface, the corresponding geometry information to this cell in another block is copied to the ghost cell in the pre-processing stage.

3.6.1 Inflow

Two types of inflow boundary conditions for internal flows are provided, fixed total pressure and temperature and fixed inflow parameters. For inlet cells in which the flow is supersonic, all the flow variables on the ghost cell are given by the incoming flow. For the subsonic case, three boundary conditions must be specified in two-dimensions. For most internal flows the three boundary conditions are:

- Total temperature

- Total pressure

- The flow angle or the velocity tangential to the boundary.

These three conditions leave one flow variable to be extrapolated by the inner flow field. In the present study, pressure is extrapolated.

3.6.2 Outflow

For exit cells in which the Mach number normal to the boundary is supersonic, all the flow variables for the boundary "ghost" cells are extrapolated from the inner flow field. For subsonic flows, only one flow parameter can be specified. A relatively simple choice is static pressure. The other three variables: velocity, velocity tangent, and density can be extrapolated from the inner flowfield.

3.6.3 Farfield

In external aerodynamics, the flow has relatively uniform free-stream conditions far from the body. In numerical simulations, flow domains have to be truncated to a finite distance from the body. The truncated faces are known as farfield boundaries. Physical boundary conditions to be imposed on these boundaries are entropy and the value of the Riemann variables (Hirsch 1990; Thomas and Salas 1986).

3.6.4 Solid wall boundary

In inviscid flow simulations, no the mass flux through the wall is specified. A simple velocity reflection technique (Allmaras 1989) is used to determine the velocity on all ghost cells.

For viscous flows, fluid in contact with a non-moving solid wall must not move related to the wall; the so called no-slip condition. For a stationary wall, this gives $u_w = 0$, $v_w = 0$ and $w_w = 0$.

For turbulent flow simulations with a wall function, shear stress and turbulence viscosity on the solid wall are determined by the wall function. To ensure this condition, the velocity on the ghost cell is approximated by,

$$V_g^\tau = V_c^\tau - \frac{2 y_n \tau_w}{(\mu_L + \mu_T)}$$

$$V_g^n = -V_c^n \qquad\qquad (3\text{-}50)$$

Where g and c represent the ghost and interior element, τ and n are tangential and normal to the boundary direction, and y_n is distance from the wall of the interior cell.

3.6.5 Wall function

In order to apply a turbulence model to wall-bounded flows, boundary conditions appropriate to the solid wall for velocity of the flow and turbulence parameters are required. For a typical turbulent flow simulation, applying a "no-slip" wall condition normally requires a very fine mesh ($y^+ < 5$) near the wall boundary and integration through the viscous sub layer. The wall function is introduced to reduce the need for resolving the flow in the turbulent boundary layer with a very fine mesh. With this method, the inner region of the boundary layer is modelled by an empirical function. Thus, a coarse mesh ($30 \le y^+ \le 150$) can be used in this region. This approach has the advantage of significantly reducing the computing time by eliminating the large portion of cells normally required to resolve the boundary layer and improving the overall convergence by removing the thin inner layers which add extra stiffness to the solution.

The wall function procedure uses the law of the wall as the relation between the velocity and surface shear stress. A universal law of the wall can be represented by the Spalding formula (Hirsch 1990), which models the inner laminar sub layer, a transition region and the logarithmic layer of the turbulent boundary layer,

$$y^+ = u^+ + e^{-\kappa B}\left[e^{\kappa u^+} - 1 - \kappa u^+ - \frac{\left(\kappa u^+\right)^2}{2} - \frac{\left(\kappa u^+\right)^3}{6} \right]$$

(3-51)

$$\kappa = 0.41 \qquad B = 5.5$$

Where,

$$y^+ = \frac{\rho_w y_1 u_\tau}{\mu_w} \qquad\qquad u^+ = \frac{|V_1|}{u_\tau}$$

(3-52)

Here ρ_w, μ_w are the fluid density and laminar viscosity on the surface, respectively, and $|V_1|$ the velocity magnitude at the adjacent cell located a normal distance y_1 away; u_τ is the friction velocity.

A wall boundary condition for turbulent viscosity is computed from the relation presented in Ref. (Abdol-Hamid et al. 1995; Wang et al. 1999; Bardina et al. 1997; Frink 1996) as

$$\mu_t = \mu_l e^{-\kappa B}\left(e^{\kappa u^+} - 1 - \kappa u^+ - \frac{\left(\kappa u^+\right)^2}{2} \right) \qquad (3\text{-}53)$$

In solving of the turbulence transport equations, the boundary conditions of the turbulence dependent variables are also required on the solid wall when the wall function is used. The turbulence parameters are determined by the friction velocity.

For the one-equation Spalart-Allmaras model, the dependent variable can be given as

$$\tilde{v}_1 = \kappa y_1 u_\tau \cdot \mathrm{Re} \qquad (3\text{-}54)$$

The wall function sometimes does not give satisfactory solutions for separated flows because the law of the wall does not hold in the regions where separation occurs, so caution should be taken when dealing with separated flows. Furthermore, the numerical solution is very sensitive to the points above the solid wall where the wall function is used. This leads to our implementation of the wall function.

In a typical 3D tetrahedral control volume setting as shown in Figure 3.11, the wall function is applied on the cells just off the boundary. The velocity at the cell centre and the distance to the wall are used to work out the friction velocity on the wall according to equations (3-51) and (3-52). Shear stress on the wall is given as (Abdol-Hamid et al. 1995),

$$\tau_w = \mathrm{Re} \cdot \rho u_\tau^2 \qquad (3\text{-}55)$$

The shear stress should be applied to the calculation of viscous fluxes across the surface, and the turbulence viscosity on the surface is given in (3-54).
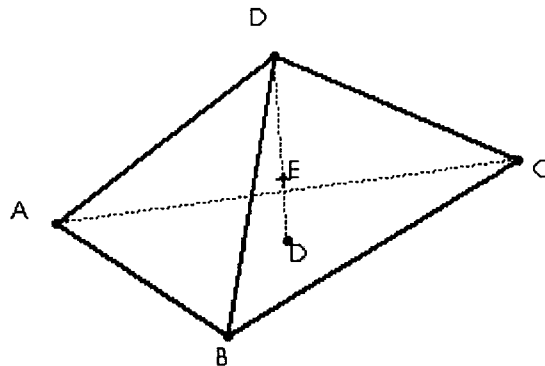

Figure 3.11 Wall function for tetrahedral elements near the solid wall

In a semi-structured 3D prismatic setting as shown in Figure 3.12, we exploit the inherent structure present in the mesh produced by an "inflating" method (Chapter 4). As shown in Figure 3.12, the nodes are aligned along the line normal to the boundary surface. The surface centred friction velocity and shear stress are determined by a two-step process. Our implementation is inspired by Frink's (1996) similar implementation for tetrahedral volumes produced by an advanced layers method.
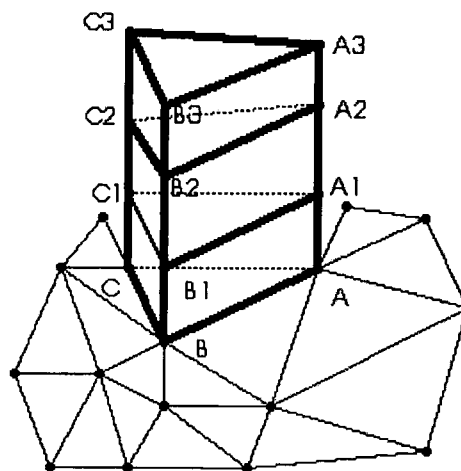

Figure 3.12 Wall function for prismatic cells near a solid wall

First, the friction velocity at nodes A, B and C (nodes on the surface of the wall) are calculated based on the equation (3-51) and (3-52) using node quantities at A1, B1

and C1, which are on the first layer off the wall. Next, friction velocity at the surface ABC is computed by averaging the friction velocity on nodes A, B and C. Thus the shear stress is calculated using equation (3-55). Then, the turbulence viscosity and parameters on the surface are defined by equations (3-53) and (3-54).

The wall function implementation based on prismatic volumes with the layer structure is generally more efficient and robust than the previous one on tetrahedral elements. Because the wall function procedure involves a Newton iteration procedure to compute the friction velocity as shown in equation (3-51) and (3-52), it is likely to be very computational expensive. In a typical 3D surface mesh setting, there are two times as many triangular surfaces as nodes. The wall function iteration is likely to be nearly two times faster in the second implementation than in the first one. Secondly, the distribution of the friction velocity on the wall in the prismatic implementation is smoother because of the averaging process; a discontinuity is less likely to appear.

3.6.6 Periodic boundary

A periodic boundary is present in most turbomachinery applications. Strictly speaking, the periodic boundary condition is not really a boundary condition. With a mesh with a periodic condition, periodic nodes and edges are treated as inner nodes and edges, the fluxes through the pair of edges are calculated only one time and given to two periodic cells.

Chapter 4

Mesh Generation and Adaptation

The primary objective of the present research is to develop efficient and accurate numerical algorithms for solving the Euler/Navier-Stokes equations on unstructured-grids. It is well known that quality of computational grids is very important to the accuracy and efficiency of a CFD solution. However, the main focus of the present research is not about developing a new unstructured mesh generator. The aim of the mesh generation part of the work is focused on generating well-formed viscous grids with current available isotropic mesh generators in the public domain and manipulating the grids for viscous flow computations. In this chapter, the unstructured mesh generation and adaptation technique for inviscid and viscous flow computations are presented.

The first section of this chapter is about unstructured mesh generation for aerodynamic computations. First, different approaches for the isotropic unstructured mesh generation are introduced. Next, the viscous mesh generation method is reviewed and the "inflating" strategy is presented. This section ends with the details of generating 2D/3D unstructured meshes with this "inflating" method.

A mesh adaptation approach for unstructured meshes is developed in the second section of this chapter. It begins with the definition of rules for the mesh refinement and error estimation. Then, the mesh refinement and reconnection technique are described in some detail. This chapter ends with the discussion of the mesh adaptation techniques in viscous flow simulations.

4.1 Unstructured Mesh Generation

There are various reasons that the unstructured-grid method becomes widely used in industrial applications. The most important one is the geometry flexibility of this method, i.e. the ability to generate high quality grids for virtually any geometric configurations, at least in theory. It is well known that mesh quality greatly influences the accuracy and convergence of a solution. A skewed and non-smooth mesh generally results in slow convergence rate of the solution and less accurate results. Generation of high quality unstructured grids for simulating flows in complex geometry configurations becomes increasingly important.

In the current research, the unstructured mesh is based on triangles in two-dimensional and tetrahedron or/and prismatic elements in three-dimensional. In the following section, strategies for generating unstructured meshes for inviscid/viscous flow simulations are outlined.

4.1.1 Isotropic Mesh Generation

In inviscid flow computations, isotropic unstructured meshes are required for economical computing and accurate results. The reason is that in inviscid flows the information propagates at almost the same rate in all directions. It is sensible to use a grid that has the same resolution in all direction. Currently, methods for generating isotropic unstructured meshes in 2D computational domains have reached a fairly mature state. There are two major lines of methods for the 2D unstructured mesh generation: the Advancing Front method (Merriam 1991) and the Delaunay method (Bowyer 1981; Baker 1989). Both of them are widely used in unstructured mesh generation community.

The Advancing Front method has become a routine practice for two- and three-dimensional unstructured mesh generation. The process could be described as elements creeping into a domain from its boundaries. In 2D, it starts with boundaries of the domain as an initial front. Then triangles are generated from current front into empty domain, and the front is updated. The operation is repeated until the whole

domain is triangulated. In three dimensions, the front becomes a layer and this method is also called the advancing layer method.

The Delaunay method adopts the empty circumcircle property of a computational domain. It is generally more efficient than the advancing-front method (Liu and Hwang 2001). However, the advancing-front method has the advantage of being more robust because the boundary integrity is guaranteed.

In the present work, a two-dimensional unstructured mesh generator developed by the author in Beijing, China, 1994-1995 (Zheng 1995) is used to discretise a 2D computational domain. This 2D unstructured mesh generator is based on the Advancing Front method described previously and able to produce high quality triangles for virtually any 2D complex geometry configurations. It features a very coarse background structured grid and a series of source points to control the size of local elements.

The discussion of the method for 3D isotropic unstructured mesh generation is beyond the present research. In the current work, the 3D computational meshes for inviscid flow computations are generated by free software in public domain such as GMSH (Geuzaine and Remacle 1999) and GRUMMP (Ollivier-Gooch 1998; Freitag and Ollivier-Gooch 1997; Ollivier-Gooch 2001).

For applications with viscous flows, the Advancing Front method described previously is inadequate because the method tends to generate "good" equilateral triangles even in a boundary layer, which is not desirable. Highly stretched grids are required in viscous effects dominated regions, such as in a boundary layer or wake, to resolve high gradients of the flows. A major bottleneck in the application of simulations of 2D/3D viscous flows in complex configurations is still the generation of highly stretched and body-fitted viscous grids.

4.1.2 Viscous Mesh Generation and Multiblock Method

The unstructured grid method has shown to be very successful in simulating of inviscid flows both in 2D and 3D (Frink et al. 1991; Marvriplis and Jameson 1987).

This is partially because the generation of high quality computing meshes for inviscid flow simulations is relatively easy. For viscous flow simulations, highly stretched elements are required near the wall regions, in which viscous effects dominate the flow. The traditional unstructured mesh (consists of only triangles in 2D and tetrahedron in 3D) method has been less successful. The first reason is the great difficulty associated with generating a good viscous mesh, especially for complex 3D geometries. This is generally caused by the difficulty to generate ideal stretched elements near solid wall regions. A more important reason is that poorly stretched triangular or tetrahedral elements usually lead to less accurate results and poor convergence.

It seems sensible to use elements that are more suitable for viscous flow simulations, like quadrangular (2D) and prismatic or hexahedral (3D) elements in regions that viscous effects dominate the flow; and in the outer regions, i.e. far from solid wall regions, triangles or tetrahedron could be used to cover the domain with great flexibility. This leads to the development of the hybrid (or mixed-grids) method, in which the computing mesh may compose of several types of elements. Some authors (Sayma et al. 2000) presented their solution methods with a hybrid method. In spite of its flexibility in three-dimensional mesh generation for some relatively simple geometries, mixed-grid methods share the difficulty of generating a high quality 3D mixed-grid mesh, and because of the complexity of the solution method and the efficiency penalty by introducing different type of elements.

In most viscous flow problems, flow domains consist of two parts: viscous regions (where viscous effects dominate the flow) and the rest of the domain, in which viscous effects are less significant. The viscous regions most likely appear near to solid walls, or in wake and mixing regions, where flows change rapidly in one direction but slowly in the other directions. In these regions, highly stretched grids are required to resolve high gradients of the flows. In the rest of the domain, such as far from walls and farfield regions, flows change virtually at the same rate in all directions. In these regions, isotropic grid elements are more desirable.

Based on the evidence presented, it seems best to divide the flow domain into viscous regions and other regions for the mesh generation. In the viscous regions, semi-structured blocks, which consist of hexahedral or prismatic elements in 3D and triangles or quadrangles in 2D, should be used to discretise the domain, while for the remainder of the domain, triangular (2D) or tetrahedral (3D) elements should be used. By doing this, ideally stretched elements can be easily generated according to the local flow features in the viscous regions. In the rest of the domain, the use of isotropic elements enables rapidly changing of mesh density without compromising the grid resolution.

The use of this multiblock method in mesh generations has following advantages over a hybrid method:

- The mesh in the viscous regions is semi-structured or structured, which enables the possibility of using high order schemes and convergence acceleration techniques well developed in past years. Furthermore, it gives more choice of turbulence models and viscous wall treatment, and possibly higher numerical resolution in these regions.

- Since the flow domain has been divided into several blocks with different element types, different solvers based on these elements can be used. Thus, the complexity of a cell-centred flow solver is lower than a hybrid solver, and the performance is higher. However, the difficulty of coding is increased because more than one flow solvers are required to march the flow in the whole computational domain.

- Using the semi-structured blocks enables flow solvers to change the computing mesh dynamically, which is vital for a mesh adaptation or dynamic mesh technique. With a hybrid method, the mesh adaptation by subdivision becomes nearly impossible or too expensive to perform because the reconstruction of the grid structures is too difficult due to the presence of different types of elements. While in the present method, a mesh adaptation technique by subdivision is relatively easier to be implemented

- More importantly, using this semi-structured mesh enables the exploitation of an efficient multigrid method on unstructured meshes, which will be discussed in next chapter.

In the present research, a few viscous layers are generated in the viscous regions in both 2D and 3D to resolve rapid changes of flows in these regions. Details of the implementation of the algorithm are described in following sections.

2D Viscous Mesh Generation

In two dimensions, the process of generating a viscous mesh is very similar to "inflating" a solid wall. The details of the implementation on a single airfoil could be described as following:

1. Calculate the thickness of the viscous layer. Prior to the mesh generation, the maximum boundary layer thickness can be estimated with a simple empirical formula for each solid wall. Then the thickness of the viscous layer can be given as 2 ~5 times the thickness of the boundary layer. This will be an approximate thickness of the semi-structured layer.

2. Shift solid walls. As a typical single airfoil case, depicted in Figure 4.1 (suppose we are not interested in the wake), a new artificial "boundary", whose distance from the solid wall roughly equals the layer thickness established in the first step, can be worked out and placed in the flow domain. This process is very similar to inflating an object. This new artificial boundary described by a series of connected points for each solid wall separates the viscous layer and the rest of the regions.

3. Generate an isotropic mesh. After the walls are shifted, the outer region, which is bounded by the artificial boundary or boundaries and other physical boundaries, can be easily discretised with an isotropic unstructured mesh generator.

4. Generate viscous layers. A structured mesh can be generated in the viscous layer using an algebraic method. In this process, the grid lines virtually parallel to the wall should be carefully placed according to the flow problem. Depending on the nature of the geometry and which part of the flowfield is concerned, either an 'O', 'H' or 'C' type of structured-grid could be used. For an isolated airfoil (Figure 4.1), when the wake is not a concern, an 'O' type of structured-grid is the best choice, otherwise, a 'C' mesh may be used to resolve the wake and boundary layer flows.

5. Triangulate the viscous mesh. The structured mesh is triangulated by subdividing each quadrangle to two triangles. This stage is necessary for the present 2D cell-centred finite volume flow solver, which can only handle triangles as control volume.
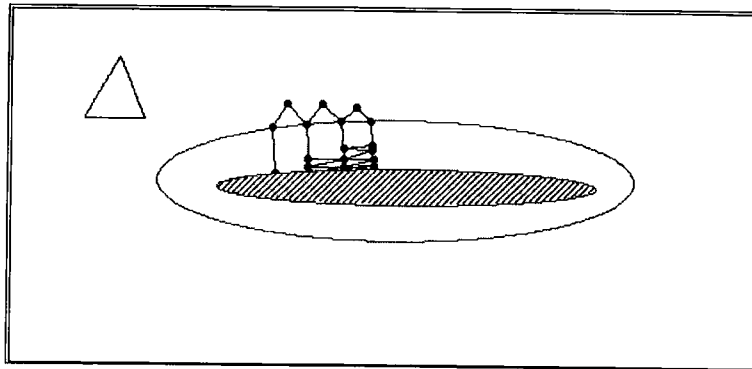

Figure 4.1 Mesh generation for an airfoil using an inflation method

Since the structured-grid generation and triangulation procedure for the viscous layer are simple and straightforward, it may be best to integrate the generation of the semi-structured grid and triangulation process into the flow solver. The flow solver needs to take the actual profile of viscous walls and triangular elements in the outer regions to generate a semi-structured mesh with a given spacing of the layers. The structured-grid generation procedure is usually controlled by a few parameters, including a spacing factor on the direction normal to the wall, the position of the first grid line.

By doing so, a mesh adaptation procedure could always reuse this subroutine to produce better structured layers according to the local flow state when the mesh in this region needs to be refined.
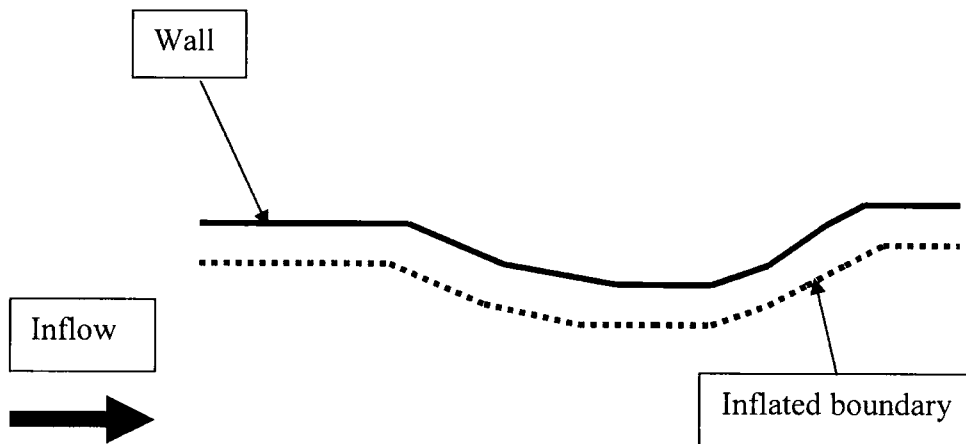

Figure 4.2 Shift an open wall

When a solid wall is not entirely closed, the artificial boundary can be worked out in a similar fashion. Shifting of an open wall is illustrated in Figure 4.2. It should be noted that the artificial "boundary" does not necessarily have same number of points or same shape as the wall. This is because when the outer regions are meshed with triangles, some points may be inserted into the inflated "boundary". In order to result in a body-fitted and ideally stretched viscous grid in this region, the wall must be remeshed to the same number of points and relative position as the artificial boundary.

The sharp ended object

Inflated boundary
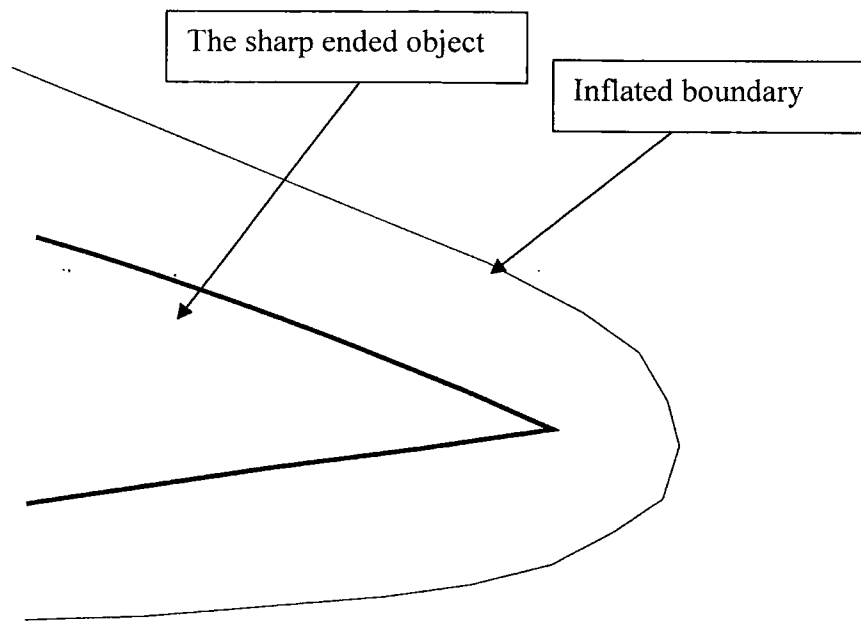
A. Inflating a sharp ended object when resolving wake is not required

Flow direction
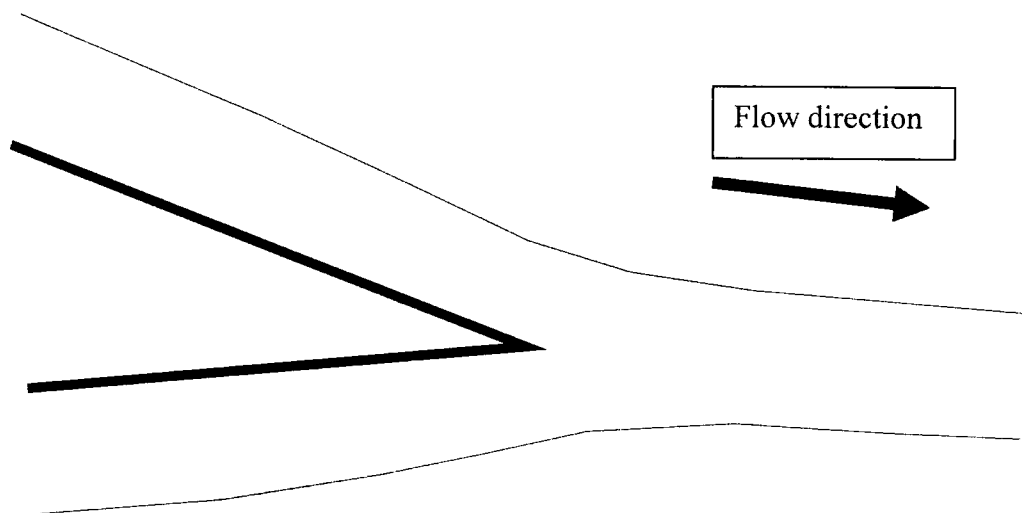
B. Inflating a sharp end object when resolving wake is required

Figure 4.3 Inflating sharp ended objects

Special care must be taken when the trailing edge or leading edge of an airfoil or a blade has a sharp end. When the trailing edge is round, the mesh in viscous regions could be made very smooth with little effort. In the case of a sharp end, extra points may need to be inserted or points relocated to ensure the artificial bluntness of the

inflated "boundary" (Figure 4.3a). Alternatively, a 'C' or 'H' type of inflation should be used to result in a body-fitted and ideally stretched viscous grid (Figure 4.3b).

3D Viscous Mesh Generation

In three dimensions, this "inflating" strategy becomes more important considering the current status of the 3D viscous mesh generation. Furthermore, the difficulty of creating the artificial boundary and dealing with corner points increases as well.

In three dimensions, there are two choices of the semi-structured mesh: either prismatic or hexahedral based. Our preference is given to the prismatic based control volume for following reasons:

- The use of prismatic elements offers geometric flexibility. When prismatic elements are used, the surface of a solid wall is represented by a full triangular mesh instead of structured grid. This is very important for complex configurations where the solid wall is hard to be represented with structured-grids. Furthermore, it has the advantage of generating a better surface mesh, i.e. using a fine grid when necessary without changing the entire surface mesh.

- Multi-block boundary condition treatment consideration. When using prismatic-based semi-structured block, the block boundary between a viscous mesh and isotropic meshes in outer regions will always be triangular elements. When hexahedron-based elements are used, the treatment of the block boundary requires expensive bilinear interpolation.

Similar to the 2D method, the artificial boundary that separates the viscous zone and outer zone have to be established by an algebraic algorithm. Next, the outer region is discretised using tetrahedral elements. Then the triangulated boundaries (the artificial boundary that separates the viscous and outer regions) with new sets of points are mapped back to the solid wall. In this stage, the solid wall has exactly the same number of mesh points and triangulation as the artificial boundary. In the flow solver for prismatic elements, prismatic layers are generated based on these two sets of mesh

points and the triangulation. Again, the layer number and relative position could be adjusted in the flow solver according to the flow state and local flow features.

In the current work, when a 3D sharp ended object is present in the flow domain, either a 'C' or 'H' type semi-structured grid is used to discretise the near wall and wake regions to ensure the valid elements in these regions. Alternatively, small changes to the sharp end part of the object may be needed to make the object artificially "blunt".

4.2 Mesh Adaptation

There are several possible ways to achieve adaptivity on unstructured meshes. One can either increase the degree of polynomial approximation to improve overall solution quality, or move grid nodes in the regions where there is a rapid change of solution. Alternatively, one can both move the grid nodes **and** enrich the grids. This is the method used here, because it provides the most flexible way to control the element size to resolve flow features such as shock waves, shear layers, wakes, flow separation and reattachment.

There are various mesh refinement techniques that may be employed. New refined points may be created and inserted into the mesh using a Delaunay point insertion. Alternatively, original elements that contain refined nodes can be simply subdivided into several elements, followed by a local smoothing or relaxation.

In the method adopted, the flow-field is firstly solved on a coarse mesh generated by the Advancing-Front Method (Pirzadeh 1993; Zheng 1995) to roughly capture the basic flow features. The resulting solution is then analysed to determine where to insert more grid nodes, and a refined mesh is generated by subdividing elements and relocating nodes. The problem is solved again on the new mesh using the solution of the coarse mesh as an initial guess. The process is repeated until the required accuracy in terms of a refinement criterion is achieved.

The advantages of this element subdivision method are higher efficiency and the connectivity is guaranteed. This is especially important when highly stretched meshes

are required in the viscous flow simulations. In viscous effect dominated regions where high stretched meshes are required, the Delaunay construction is no longer optimal and less reliable.

4.2.1 Rules to Refine Meshes

The refinement algorithm is a key factor for a successful mesh adaptation. It determines the way that the mesh is modified for the grid and is based on a set of refinement rules. The rules are:

1. Always refine the grid in the region with high gradient. High gradients always mean high truncation errors.

2. Ensure the resulting refined grid is valid.

3. Ensure the resulting refined grid is in some sense a good grid. So after each refinement, grid relaxation or smoothing can significant improve the mesh quality (Bottasso et al. 1994).

Obviously, a successful adaptation highly depends on the adaptive criteria, i.e. estimation of the errors. In following section, we will discuss the error estimation.

4.2.2 Error Estimation

The error estimation determines whether or not a current mesh cell should be further refined. For a given mesh with flow variables associated with it, there are various ways to define the numerical error in the flow field.

To capture different flow features, different criteria may be used to define the error. For shock wave associated problems, the pressure difference of each edge is more relevant than other variables. For viscous layers (attached or separated boundary layers, wakes), vorticity and velocity gradients are dominant and thus the differences of these variables are used. Whilst if there is a strong shock-wave/boundary layer interaction, it is more effective to resolve the boundary layers by using a velocity/vorticity difference before using a pressure difference to capture shock waves, because a typical time-scale for viscous diffusion is much longer than that for

a pressure propagation. Thus, for complex flow problems, the error estimation of combined pressure and velocity/vorticity differences has to be used to give an effective and efficient mesh refinement.

In the present edge-based flow solver, the error estimation is carried out on each edge. Then the error is normalised by the global maximum and minimum error as:

$$\overline{E} = \frac{E - E_{min}}{E_{max} - E_{min}} \qquad (4\text{-}1)$$

In the viscous flow simulations, multiple indicators are usually used for the error estimation to capture the high velocity gradients. In this case, the final error is estimated as:

$$\overline{E} = \sum_{i=1}^{N} \alpha_i \overline{E}_i \qquad (4\text{-}2)$$

Here N is the number of the error indicators, $\overline{E}_i$ is the normalised error of the ith error indicator and α_i is the coefficient of this indicator.

$$\sum_{i=1}^{N} \alpha_i \equiv 1 \qquad \left(0 \leq \alpha_i \leq 1 \right) \qquad (4\text{-}3)$$

A tag is marked if the error on the edge is higher than a predefined factor. The predefined factor is largely based on the experience. In the present study, the factor is from 0.1 to 0.2 for inviscid flow simulations and from 0.1 to 0.3 when multiple indicators are used for the error estimation with viscous flow simulations. For multiple mesh refinement, a sequence of factors should be given in ascending order to avoid an overcrowded mesh.

4.2.3 Mesh Refinement and Reconnection

The refinement strategy is to adjust a mesh cell for which any of its edges has been tagged to refine. In order to implement this technique, various allowable subdivision types for triangular elements can be defined as following:
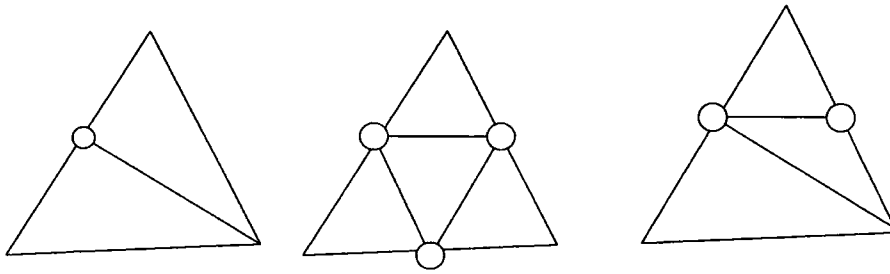
Figure 4.4 Sketch of subdivision types for a triangle

After all points have been inserted into the mesh system, the mesh is smoothed to improve the mesh quality. Smoothing is done by moving the point slightly to an optimal position. The very commonly used Laplacian-type smoother is easy to implement and fast to compute.

4.2.4 Mesh Adaptation in Viscous Flow Simulations

In the simulations of high Reynolds number flows, very thin boundary layers are presented near solid walls, where the flow quantities are subject to strong local one-dimensional gradients. Highly stretched cells in the boundary layers are normally used to effectively capture these gradients. The mesh refinement approach, as described above, has been shown to be less effective (Vilsmeier and Hanel 1993). Because of the high gradients in these regions, large amount of refinement will occur. This problem is well documented by several authors (Vilsmeier and Hanel 1993) (Warren et al. 1991). Several solutions have been developed including limiting the maximum amount of subdivision of an element in the most coarse mesh or minimum size of the element. Most of these have been unable to preserve the highly stretched body-fitted elements in the viscous regions, which is essential for economical and accurate computing. Therefore, another approach has been developed to deal with viscous flow problems.

In the present viscous mesh generation scheme, several layers of stretched triangular cells are employed around solid walls and the rest of the domain is covered with

normal triangular cells. In order to overcome the over-resolved problem by adaptive mesh refinement in the near wall region described previously, a two-phase refinement procedure is required to deal with the outer region and viscous regions separately and different refinement rules are adopted in these regions. Since the subroutine for generation of these layers in viscous dominated regions is integrated into the flow solver, these regions can be easily remeshed without affecting the rest of the domain. Details of the implementation are described in the next section.

4.2.5 Algorithm Description

In the simulation of inviscid flows, the refinement procedure in the current 2D edge-based flow solver could be as following:

1. Clean all tags on the mesh system.

2. Estimate the error across each edge. The error is modelled by the gradients of given variables across the edge.

3. Tag edges. When the error on an edge is higher than a predefined threshold, the edge is marked to be refined.

4. Mark cells based on the tags on three edges that compose the triangle. If any of its three edges is marked to be refined, a cell is marked.

5. Insert nodes based on the tagged edges.

6. Reconstruct mesh topology.

7. Smooth new mesh when necessary.

8. Interpolate the solution from the original mesh to the refined mesh.

After all the new points are inserted and a new flow field solution obtained, the flow solver is restarted from this new state.

In simulations of viscous flows, the procedure described previously could produce a very large amount of undesirable elements in the viscous effect dominated regions,

especially near walls. Since the over-resolved problem most likely occurs in the regions where highly stretched elements are used, an alternative refinement procedure and rule should be applied for these regions. In the present research, a two-phase refinement procedure is implemented to overcome the problem.

The first phase is to refine the outer region, where the viscous effects are less significant. The refinement procedure for inviscid flow simulations described previously is capable of producing well-formed triangular elements in these regions.

The second phase is to refine viscous regions where highly stretched elements are used. After point-insertion and reconnection of the outer part of the mesh, some nodes have been inserted into the artificial boundary between the viscous layer and the outer part. The artificial boundary is reconnected using nodes already in the mesh and newly inserted nodes. A new viscous grid is generated using the new boundary with the method described in the mesh generation section. In this stage, an interpolation procedure is performed on the solid wall to ensure a correct representation of the original wall profile. Currently, no extra layer is inserted into the viscous layers because the layers are already placed with consideration of the flow state and are suitable for resolving the boundary layer.

Chapter 5

Multigrid Method

In this chapter, a multigrid method to accelerate the solution of Euler/Navier-Stokes equations on unstructured meshes is developed. The objective of this chapter is to develop an efficient, robust and effective multigrid for simulating compressible inviscid and viscous steady flows on unstructured meshes.

The discussion of the multigrid method begins with the introduction of different multigrid approaches on unstructured meshes. Next, how to generate a sequence of mesh levels is discussed. Two distinct multigrid methods are developed: a direct connectivity based method and an aspect ratio sensitive approach. This is followed by the inter-grid operators and presentation of the multigrid time-marching scheme.

5.1 Introduction

Multigrid has been demonstrated as an effective means to accelerate the solution both for traditional structured methods and unstructured-grid methods. For traditional structured applications, the multigrid method has become a routine practise. Multigrid on unstructured grids, especially with mixed-elements is still at a very early stage of development.

There are different approaches for adopting a multigrid technique on unstructured meshes. The first approach begins with a coarse mesh definition and generates finer mesh levels by refinement. This approach is usually coupled with an adaptive mesh refinement technique. The second approach uses completely independent coarse and fine meshes. Since the various meshes of the sequence do not always have common points, linear interpolation has to be performed to transfer flow variables between them. Both the methods share a common difficulty in generating the coarse mesh in a complex configuration. The third one is to coarsen a given mesh by using directly neighbouring connections of fine mesh elements. This method is able to generate coarse mesh levels in virtually any complex configurations and has been proven to be effective in inviscid flow calculations. However for the high Reynolds number problems, this method is less effective due to the presence of high aspect-ratio cells near solid wall regions.

A successful multigrid approach usually consists of three key components:

1. A nicely constructed sequence of meshes ranging from coarse levels to the finest level.

2. A suitable inter-grid transfer operator to transfer solutions between coarse and fine levels.

3. An effective iterative solver to damp high frequency error components in every level of the mesh.

The following sections are descriptions of the basic multigrid method for unstructured meshes and details of generating multiple meshes, inter-grid transfer operators and multigrid cycling.

5.2 Generation of a Sequence of Mesh Levels

For a structured grid, a sequence of coarse levels can be easily generated with its inherent structure. For unstructured grids, it is not straightforward to generate a coarse mesh because of the very nature of the unstructured-grid itself, i.e. lack of simple structure and connectivity. One possible way is to use a volume agglomeration method. The aim of the current research is to extend and enhance a volume agglomeration method for 2D/3D inviscid and viscous flow calculations and to improve the efficiency of this method for high Reynolds number flows.

Central to the design of a volume agglomeration method is to group together cells that have neighbouring relations, to form a control volume of a coarse block, starting from the base fine mesh. Repeating this process allows the obtaining of coarser meshes until a sufficient number of levels are obtained. Two distinct approaches to coarsening are adopted:

1) The first one would be directly based on the element connectivity. This is relatively easy to implement, but is not very effective for highly stretched viscous meshes. It can be called 'Direct Connected Multi-Grid' (DCMG).

2) The second approach, which we advocate, is to *adaptively* coarsen the mesh, aimed at forming coarse mesh blocks with more uniform spacing in all directions. Consequently, on a coarse mesh, error disturbances would propagate at the same rate in all directions. This method can be called 'Aspect-ratio Adaptive Multi-Grid' (AAMG).

5.2.1 Direct Connected Multigrid

The Direct Connected Multigrid is essentially a volume agglomeration method based on the connectivity of the mesh. To generate coarse levels automatically from an unstructured mesh, it is possible to group together control volumes that have direct neighbouring relations or nodes that are associated with contiguous volumes. Repeating this process allows a coarse mesh to be obtained. In this process, the size of coarse mesh cells should increase, and the coarse mesh solution should accurately approximate the fine mesh solution to obtain a good preconditioner.

The coarsening approach we present here is only a modified version of the volume agglomeration method (Lallemand et al. 1992). The direct neighbour relation means that two cells have at least one shared node. The algorithm based on the direct-connectivity relations is as following (Figure 5.1):

Consider successively every cell C for the fine mesh:

1. If the cell C has already been included in a group (the cell in the new coarse mesh) then consider the next cell.

2. Create a new group containing C and put into this group each cell neighbouring C that is not already included in any existing group.

3. If the new group contains only the cell C, merge the group with an existent group, which contain a neighbour nearest to cell C.
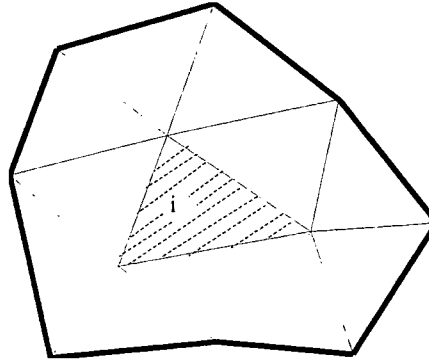
4. Go to the next cell.

Figure 5.1 2D coarser level from fine mesh for far from wall regions

The algorithm above allows an automatically coarsening triangular based 2D meshes and tetrahedron based 3D meshes. However, in order to produce optimal coarse levels, the procedure should start from the smallest element in the mesh.

5.2.2 Aspect-Ratio Adaptive Multigrid

An aspect-ratio adaptive multigrid is an enhanced version of volume agglomeration method aimed at viscous flow computations with highly stretched grids. If the base fine mesh only contains regular triangular/tetrahedral cells, the Aspect-ratio Adaptive Multi-Grid will reduce to the Direct Connected multigrid method. The coarsening algorithm based on the direct-connectivity relations is also described in Ref. (Zheng and He 2001).

For high Reynolds number viscous flow computations using highly stretched meshes with high grid aspect ratio near solid wall regions, the Aspect-ratio Adaptive Multi-grid method is activated. In regions far from solid walls where isotropic unstructured grids are used, the direct connected multigrid method outline previously is used to build coarse levels by volume agglomerating. With this method, the size of the elements in a coarse level increases almost at same rate at all directions. In viscous flow computations, with an 'inflating' viscous mesh generation scheme described in Chapter 4, semi-structured viscous layers are used in or near solid wall and wake regions to resolve high gradients and rapid changes of the flow. In these regions, where the highly stretched triangular/prismatic cells are used, coarser levels are built by stacking viscous layers in the normal wall direction. By doing so, the grid aspect

ratio in coarse levels is reduced and the time step is increased. Thus, at a coarse mesh level, the solution can effectively be marched with a larger time step.

In the present research, the Aspect-ratio Adaptive Multigrid exploits the structure of viscous layers introduced in the process of the mesh generation (Chapter 4) both in 2D and 3D. Because the viscous layers generation is integrated within the flow solver, coarser levels can be easily built up by stacking layers of highly stretched grids in the viscous layer.

In the viscous layers near solid wall or wake regions, the grid aspect ratio of elements in every layer are compared with a predefined grid aspect ratio value (30 in our case) to determine how the coarser level should be built. If the grid aspect ratio higher than predefined value, four triangles (2D) in the two layers next to each other are to be stacked to form an element in the coarser level, as shown in Figure 5.2a. When the grid aspect ratio is smaller than the predefined value, as depicted in Figure 5.2b, two layers and one of their neighbouring rows, which are not included in any coarse level, are to form an element in the coarser level. This process is repeated until desired coarse level meshes are constructed. In this process, the size of elements are always increasing whilst the aspect ratios of the grid are decreasing, which allow a larger timestep on coarser levels.

In three dimensions, a viscous layer consists of semi-structured prismatic grids. A strategy similar to 2D is adopted. The aspect ratio of a prismatic element is defined by the height of the element and the typical length of its bottom triangular surface: usually the longest edge. Starting from the first layer next to a solid wall, two neighbouring layers are combined to form an element in a coarser level. This process is repeated this process until a sequence of coarse mesh levels is constructed. However, no element combination along the wall is considered in the process. The process is described in Figure 5.3.
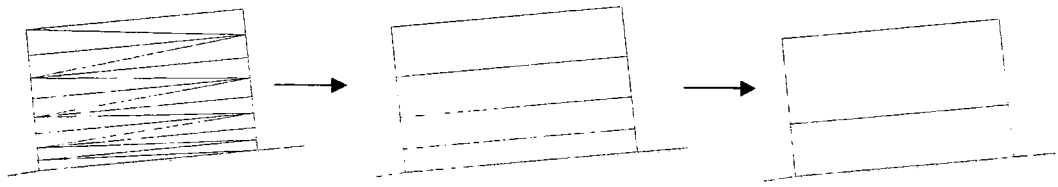
a. Coarse levels for cells with high aspect ratio in near wall regions
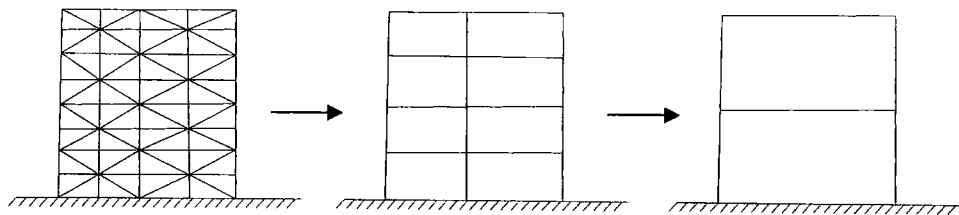
b. coarser levels for normal triangular cells in near wall regions

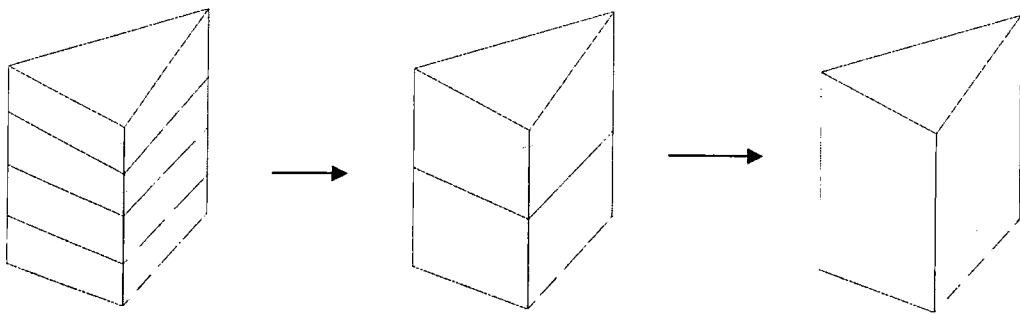**Figure 5.2 2D coarser meshes generation in visous layers**

Figure 5.3 3D coaser meshes generation in viscous layers

5.3 Intergrid Transfer

In order to transfer a solution from one grid to another, consideration needs to be given to the design of the multigrid method. In the traditional multigrid approach, linear or nonlinear interpolation is used to transfer flow variables and residuals

between the various meshes of the sequence. In the approach adopted in the present study, an efficient collecting/distributing process is adopted to transfer solutions.

In the Fine-to-Coarse stage, fluxes of a finer mesh are summed to obtain the fluxes of the coarser meshes instead of using a first order upwind solver, which is considered too expensive to perform on a highly polyhedral grid.

In the Coarse-to-Fine stage, the coarse mesh residual are directly distributed back to the fine mesh cells, which has been shown to be effective on structured mesh solvers for steady flows (Denton 1983), and unsteady flows (He 1993).

In this way, only fluxes on the basic fine mesh are evaluated by the upwind scheme, and the fluxes of elements on a coarse mesh can be directly obtained from those on the finer mesh of the sequence:

$$R_m^{i+1} = \sum_{k=1}^{N} R_k^i \qquad (5.1)$$

where N is the number of cells in the finer mesh that contained in the element m of the coarse mesh level. i and $i+1$ indicate the sequence of the meshes. The solution is accelerated by distributing the residuals on cells of the coarse mesh to the finer mesh. Consider a one-stage time integration of a cell on k levels of meshes:

$$\left(Q^{n+1} - Q^n\right) = \frac{\Delta t^i}{A^i} R^i + \frac{\Delta t^{i+1}}{A^{i+1}} R^{i+1} + \dots \qquad (i=1,2,.. k) \qquad (5.2)$$

The left side is the effective change of flow variable in one step of time-marching. Δt^i, A^i and R^i denote local time step, area, and fluxes of mesh level i, respectively.

This method is straightforward to implement because of the conservation relation for both fluxes and areas. Another key feature of this method is its speed, because it does not require calculating fluxes on coarse levels by integration. For instance, on a typical three level multigrid mesh configuration (2D), the number of cells of the three level meshes (fine, coarse and coarser mesh) are in the ratio of 12:4:1, the total

computing time of the 3-level multigrid will increase by about 10% per step compared to that for the baseline single grid.

5.4 Stability and Timestep of Multigrid Approach

The efficiency and robustness of the multigrid approach depends on accurate estimation of the local timestep. For steady problems, maximum local time step is essential for efficiency. Since the coarse meshes are very likely to be highly non-uniform polyhedral, it is very difficult to compute a stable local time step for a coarse mesh by Fourier analysis. Hence, we prefer a simple estimation of the local timestep.

$$\Delta x = \tfrac{1}{2}\sum_{i=1}^{n}\left|\Delta ex_i\right|, \qquad \Delta y = \tfrac{1}{2}\sum_{i=1}^{n}\left|\Delta ey_i\right|, \qquad \Delta z = \tfrac{1}{2}\sum_{i=1}^{n}\left|\Delta ez_i\right|$$

$$\Delta t = CFL \cdot \frac{V}{(u+c)\cdot \Delta x + (v+c)\cdot \Delta y + (w+c)\cdot \Delta z} \tag{5.3}$$

Here n is the number of edges or faces which form the polyhedral element and Δex_i, Δey_i and Δez_i are the projected length/area of the edge/face in three coordinate directions, respectively. Δt is the time step of the element, which volume is V.

Chapter 6

Parallel Computing

In this chapter, we review the parallel computing technique in CFD with unstructured meshes, covering the subjects of parallel computer, parallel programming models, data distribution and mapping, and communication schemes.

Parallel computing for CFD involves software, hardware and algorithm design. It promotes a view of parallel computing as an engineering discipline, in which programs are developed in a methodical fashion and both cost and performance are considered in a design. The discussion of parallel computing in this chapter is divided into five parts. The first part is the introduction of the parallel computing environment and programming model. Then, the multi-block method for unstructured meshes is reviewed. Consequently, the mesh partitioning, data mapping method and related software are discussed. Next, the issue of parallel computing performance on a PC cluster system is discussed. The last section describes in detail the implementation of the parallel computing technique in the present CFD code.

6.1 Parallel Computing Environment

"A *parallel computer* is a set of processors that are able to work cooperatively to solve a computational problem" (Foster 1998). By the configuration of their memory system, the parallel computer can be classified as multiprocessor with uniform shared memory system, cache-based processors with uniform shared memory system, cache-based processors with non-uniform shared memory access system and cache-based processors with distributed memory system (Mavriplis 2000). It is also possible for a group of computers (for example, a group of PCs each running Linux or windows system) to be interconnected by a network to form a parallel-processing cluster system.

Parallel computing has not been widely accepted in the production engineering environment mainly due to the complexity of parallel programming and low accessibility of parallel computers. On a parallel computing system, a task has to be partitioned and distributed appropriately among processors. In the mean time, the communication cost should be minimised and loads among processors balanced. More importantly, even with careful partitioning and mapping, the performance of an algorithm may still be unsatisfactory, since conventional sequential algorithms may be serial in nature and may not be implemented efficiently on parallel machines. In order to achieve optimal performance, in addition to partitioning and mapping, a careful performance study should be conducted for a given application and parallel system to identify the strength and weakness of this system.

6.1.1 Parallel Programming Model

Although the concept of parallel processing has been used for many years in many systems, it is still somewhat unfamiliar to most CFD researchers. Thus, before discussing details of the implementation, it is important to become familiar with two parallel architectures: SIMD and MIMD.

SIMD (Single Instruction stream, Multiple Data stream) refers to a parallel execution model in which all processors execute the same operation at the same time, but each processor is allowed to operate upon its own data. This model naturally fits the

concept of performing the same operation on every element of a mesh on multiprocessor machines. Because all operations are inherently synchronized, interactions among SIMD processors tend to be easily and efficiently implemented.

MIMD (Multiple Instruction stream, Multiple Data stream) refers to a parallel execution model in which each processor is essentially acting independently. This model most naturally fits the concept of decomposing a program for parallel execution on a functional basis. This is a more flexible model than SIMD execution, and it is fit for both multiprocessor systems and networked systems.

SIMD has the advantage of being easy to implement. However, when a parallel execution requires several different programs working together, MIMD is the choice. In the present study, when computing a 3D viscous flow problem, there are two types of blocks, prismatic and tetrahedron, present in the viscous mesh. A prismatic block requires a prism based flow solver and a tetrahedron block requires a tetrahedron based flow solver to march the solution on given meshes. Our preference is given to a MIMD implementation of parallel computing.

6.1.2 Cluster of PC Systems

In the University of Durham, there are numbers of workstations available for computing. These workstations are operated by the IT department in the university. Most of systems are running SUN OS and each of them has up to four processors. These workstations are providing generic computing, Email and WWW services for students in the university. They are inter-connected with high-speed connection within the IT centre. Because these workstations are not fully accessible to the author and jobs may be subject to over-crowded users, they are not suitable for any dedicated parallel computing tasks.

In the thermo-fluid division in School of Engineering, University of Durham, there are some desktop PCs and old workstations are available for cluster computing. These PCs are running the Windows 95 operating system for a range of functions including office applications, Email and WWW services. They are connected to a 10M/s HUB, which is connected to the local network within School of Engineering. It

is well known that the Windows 95 has limited network capability, although there are some successful attempts made toward using PVM in Windows 9x systems (Fischer 1999). Extra software packages (Fischer 1998) are required for running parallel jobs on a windows 95 system and it is not famous for its stability. Therefore, it is not suitable for parallel computing tasks. The good thing is that the Linux system can be easily installed on these PCs with full network functions and support of most parallel computing software. A dedicated computing cluster system was built using these PCs within School of Engineering as Figure 6.1.
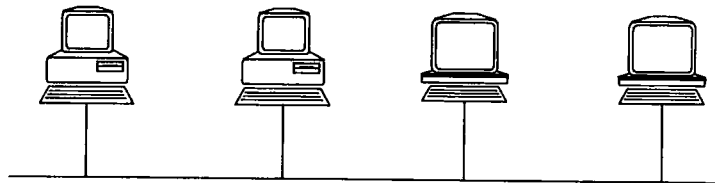
Figure 6.1 Cluster of PCs in the University of Durham

Cluster computing offers great potential, but that potential may be very difficult to achieve for most applications. However there is quite a lot of software support that will help to achieve good performance for programs that are well suited to this environment, and there are networks designed specifically to widen the range of programs that can achieve good performance. These include a software system widely used to for parallel computing: PVM.

6.1.3 The Software Package: PVM

The development of PVM started in 1989 at Oak Ridge National Laboratory. Central to the design of PVM is the notion of a "virtual machine" (VM), a set of heterogeneous computers connected by a network that appears logically to a user as a single computing resource.

PVM has some features suitable for the current implementation of parallel computing on a PC cluster system.

- It is portable. In PVM, communication between hosts is done by message passing. PVM supplies a range of message passing API which is available to

most known platforms: Windows, Linux, UNIX and UNIX compatible systems. Thus, programs developed on one platform with PVM can be easily port to another platform without major modifications.

- Process control and dynamic resource management. In PVM, program not only can add/delete hosts any time when necessary, but also can spawn/kill tasks at any nodes within VM any time. This enables the current design of the MIMD parallel computing model and gives the program maximum control of the computing process with minimum user intervention. Furthermore, the dynamic process control is also essential to dynamic load balancing for achieving maximum speedup gains on a parallel system.

- Error tolerance. PVM can detect errors during the message transmission and notify the user of the errors.

- It supports heterogeneous hosts. PVM support a range of hardware, such as X86 PCs, PPCs, Workstations, multiprocessor systems. It also supports the co-existence of a wide range of operating system in the VM: Win9X, WinNT, Mac OS, Linux and UNIX. It is well known that some of these systems are not binary compatible. PVM message passing library can translate the message when the source and destination hosts are not compatible. This enables the possibility of making use of all the old PCs and workstations running different operation systems.

In the present study, PVM is used as the parallel computing platform on a PC cluster system. Message passing between flow solvers running on hosts across the network is via PVM message passing library. A control program is developed to dynamically manage the virtual machine and computing processes.

6.2 Multi-Block Method and Parallel Computing

The multi-block concept has been widely and successfully used in structured-grid flow solvers (Rizzi et al. 1992) to deal with the difficulties of grid generation in complex configurations for many years. In this method, the problem to be solved over

a given domain is divided into many sub-domains, called **blocks**. By doing this, a more complex geometry may be divided into a sequence of simple geometries. Then, a structured grid in each block can be easily generated. The blocks are interconnected to each other through block boundaries. This method becomes very popular in structured codes because of the capability of dealing with complex geometries.

The multi-block approach in unstructured meshes does not receive much attention, partially because computational meshes for complex geometries can be relatively easy to generate using an unstructured-grid method without the aid of the multi-block approach. However, due to the increasing needs for generating well-formed and body-fitted viscous grids for 3D turbulent flow simulations, the multi-block method becomes increasingly important for the unstructured-grid method. Initially, the multi-block approach is employed in unstructured solvers to reduce the memory requirement. Sheng etc (Sheng, Tylor et al. 1995) reported multi-block approach both for structured code and unstructured codes (Sheng, Whitfield et al. 1999). They found the multi-block technique can significantly reduce the memory requirement for both structured and unstructured methods. Most importantly, the multi-block method exposes opportunities for parallel execution.

Most computing problems have several parallel solutions. The best solution may differ from that suggested by existing sequential algorithms. The design methodology that we describe is intended to foster an exploratory approach for data parallelism for CFD applications with unstructured grids. The multi-block method meets the need for data parallelism: blocks are served as partitions for concurrent computing.

6.3 Partitioning Unstructured Meshes

Partitioning the computing grid is a fundamental component in parallel computing. In a cluster system, the main memory of the system is distributed over the networked hosts. Therefore, the program and its associated data, such as the computational grid and solution, must be distributed between processors. This leads to the issue of how to partition a large unstructured grid.

One partitioning scheme is to ensure every element of the mesh is assigned uniquely to a partition which is often associated with a processor and inter-partition boundaries consist of faces from the original mesh. The nodes and faces on an inter-partition boundary are duplicated. This method is often called a non-overlapped method. Another approach called an overlapped method involves constructing a halo zone between partitions, where grids are overlapped in these regions. Our choice is given to the non-overlapped method because the numerical efficiency of this method is higher than an overlapped method.

The first objective of a non-overlapped partitioning scheme is to ensure an even distribution of computational workloads among the processors according to their computing powers. Secondly, the amount of time spent on inter-processor communication and on waiting for other processes to finish their computing is minimised. The first requirement is termed load balancing. If the workload is not well balanced on a distributed system, some processors may have to wait at synchronization points for other processors to finish their computing in order to commence communication. Inter-processor communication is generated by the mesh surface that straddles two adjacent mesh partitions. The second requirement comes from the fact that the communication time cannot always be ignored especially on a cluster system.

Partitioning can be done recursively starting with the problem of dividing one domain to N sub-domains. The mesh could be partitioned by a variety of methods. An obvious approach is to partition the domain according to the geometric feature of the particular problem. For example, a single airfoil computation could be performed using two domains, one on top the airfoil and one below it. This approach is popular in simple geometries, because it is simple, robust and quick. However, this method is unable to produce well-balanced partitions and optimistic partitioning, i.e. minimised communication volume. Our preference is given to a graph based method, such as METIS.

6.3.1 Software Package: METIS

METIS (Karypis and Kumar 1998) is a software package for partitioning large irregular graphs and partitioning large meshes. It is capable of using multi-constraint partitioning graphs and providing high quality partitions with the option of minimizing the total communication volume and minimizing the maximum connectivity of sub-domains.

The algorithms in METIS can be used to compute a balanced k-way partitioning that minimizes either the number of edge-cuts or the total communication volume. The communication volume definition in METIS is different from the communication cost in our parallel computing. In the present implementation, communication occurs where the two adjacent elements are separated into two blocks. Therefore, the edge-cut is more appropriate to define the total communication volume. The objective of partitioning is down to minimise the edge-cut while balancing the load in each partition.

METIS provides two programs PMETIS and KMETIS for partitioning a graph into k parts. Both programs provide high quality partitions. However, depending on the application, KMETIS is preferred when partitioning the graph into more than eight partitions, and PMETIS is preferred when partitioning a graph into a smaller number of partitions. METIS also provides a library interface that can be used in a user's partitioning program to partition a graph. In the current implementation, the METIS library that can be used be partition graphs into unequal-size partitions is linked to the main control program that is responsible for mesh partitioning.

6.3.2 Mesh to Graph Conversion and Mapping

As we discussed previously, METIS is a software package capable of partitioning a graph into partitions. In the present cell-centred finite volume setting, all the fluxes are computed along faces and accumulated to cell centres during the residual evaluation. To ensure the interface of two adjacent cells will not be broken, the triangle or tetrahedron based unstructured mesh has to be converted to its compatible graph for partitioning.

A mesh can be transformed to its compatible graph by connecting all adjacent element centroids. Figure 6.2 demonstrates a 2D unstructured mesh and its compatible graph. In three dimensions, the connection is similar to 2D, except each element has up to four edges connected to its adjacent elements. The graph is often referred to as a dual of the original mesh. In this procedure, the connection topology and element-cell mapping must be stored for reconstructing meshes from a partitioned graph.
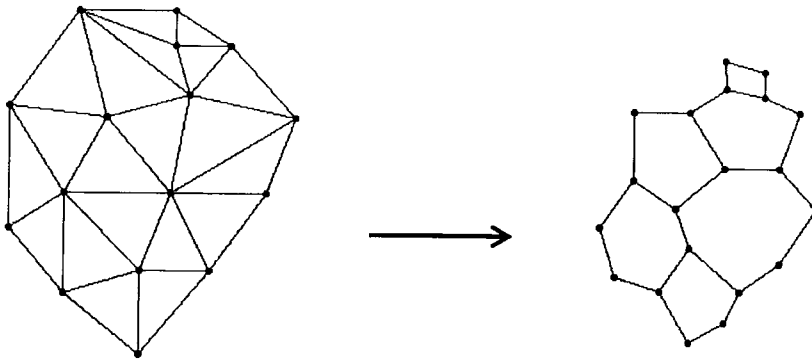

Figure 6.2 A 2D unstructured mesh and its compatible graph

After a mesh is successfully transformed to its compatible graph, the METIS subroutine is used to partition the graph to a number of sub-graphs. It should be noticed that in the current setting, all the weightings on every nodes are set to be 1 because of the fact that the amount of communication at every edge is the same. When the partitioning of the graph is done, the partition information based on the graph is mapped back to the mesh based on the connection topology and element-cell mapping. Thus, the original mesh is divided into several unstructured-grid blocks for parallel computing. Each block has a completely valid unstructured mesh, which is a subset of the original mesh. In this process, block boundaries between two adjacent blocks is constructed by duplicating nodes and faces shared by blocks according to the partitioning information. Natural boundaries are divided when necessary and assigned to blocks accordingly.

6.4 Performance of Parallel Computing

The nature of the parallel computational problem, and to a lesser extent, the programming model, dictates the degree of communication that is required. Thus, unlike its counterpart serial computing, the performance of the parallel computing is not only evaluated based on the computer power, but also the costs of the communication between distributed processors over the network. In order to obtain optimal computing efficiency, reducing both computation and communication costs should be considered.

For a given CFD problem, the total computational costs are normally bounded by the size of the computing mesh and numerical algorithm adopted. Additional computation overhead introduced by parallel computing, such as packing/unpacking communication data and computing block boundary multiple times, can be reduced by optimal program design and communication pattern.

The communication costs are usually decided by two factors, bandwidth and latency of a communication system. Latency of a communication system is the minimum time taken to transmit one message, including any send and receive software overhead. Latency is very important in parallel computing because it determines the minimum useful gain size, the minimum run time for a segment of code to yield speed-up through parallel execution. Basically, if a segment of code runs for less time than it takes to transmit its result value (i.e., latency), executing that code segment serially on the processor that needed the result value would be faster than parallel execution; serial execution would avoid the communication overhead. The bandwidth of a communication system is the maximum amount of data that can be transmitted in a unit of time. Bandwidth for serial connections is often measured in bits/second (b/s), which generally corresponds to 1/8 of the number of Bytes/second (B/s). For example, a 10M Ethernet transfers about 1.25 MByte/s, whereas an up-to-date PC with Intel Pentium IV processor with RDRAM has up to 4GB memory bandwidth. High bandwidth allows large amounts of data to be transmitted efficiently between processors.

6.4.1 Load Balancing

Apart from the numerical algorithmic efficiency, one also needs to consider the performance of the overall computation, such as processor speed and communication speed between processors. For the traditional computing, i.e. serial computing, computer central processor speed is always the bottleneck. For a cluster system, the bottlenecks could arise because the computational loads of processors are not even or the communication costs are too high.

In a cluster system, the computing time on each sub-domain is decided by dividing the amount of computation with the processor's power of the node. Considering the overall computing performance, the computing time is only affected by the slowest domain. Obtaining maximum efficiency leads to the amount of computation (load) of a partition balanced by the power of the processor, with which the partition is associated.

For a parallel computing problem, the amount of computation of a partition is a function of the total number of elements in this partition. Therefore, to balance the load, the number of elements in a partition should be,

$$M_k = \frac{P_k}{\sum\limits_{i=1}^{N} P_i} \bullet M_{Total} \tag{6-1}$$

here P_k is the computing power of the processor, N is the total number of processors and M_{Total} is the number of elements in the global computational mesh.

In some cases, load balancing not only means balancing of computational loads, but also communication costs. The execution time of a partition is denoted as:

$$T_{exe} = T_{comp} + T_{wait} + T_{comm} \tag{6-2}$$

Here T_{comp} is the computing time for the flux evaluation, boundary condition treatment and time integration and T_{wait} is the waiting time at each synchronization

point when the computational load is not well balanced. The third term in the right hand side of the equation is the communication time, which represents the performance penalty introduced by partitioning. It is a linear function of the communication volume, which is modelled by the number of duplicated faces or edge-cut in a partition. For a typical CFD problem on a cluster system, the communication volume for a block at each time step can be expressed as:

$$V_{comm} = \sum_{k=1}^{N_{ne}} (\beta E_k) \qquad (6-3)$$

Where N_{ne} is number of neighbouring partitions of the block and E_k is the number of duplicated faces with the current neighbouring partition. β is a constant depending on the computing problem and message pattern. From the equations (6-2) and (6-3), it is clear that load balancing also requires the number of edge-cuts in each partition to be balanced to obtain maximum efficiency. However, it is often very hard to achieve the minimum edge cut globally while edge cut numbers in every partition are equal.

The efficiency (or speedup) of a parallel computing can be modelled as:

$$\eta_{sp} = \frac{T_s}{T_{exe}} \qquad (6-4)$$

here T_{exe} is parallel execution time and T_s is serial computing time.

In a small cluster, the communication costs are relatively small compared to execution time for a typical CFD problem because relatively fewer edges have been cut and each partition generally has few neighbours. The communication volume and number of edge-cuts and neighbours increase when the computing domain is partitioned to more sub-domains. At this stage, while the execution time in each partition is decreased, the speedup of the parallel computing is increased. In the mean time, the communication time is increased, due to the more edge-cut and each partition has more neighbours. When the communication time exceeds the computing time in the slowest partition, the speedup will decrease even when the number of

processors and partition is increased. The maximum efficiency is reached when the communication time equals to the computing time.

6.4.2 Fast Communication

When computing on traditional parallel computers, such as a multiprocessor system, load balancing is very important, because the communication costs are relatively low due to the high bandwidth and low latency of the system. On a PC cluster system, it is often very important to optimise the communication because of the high latency and limited bandwidth of the system, which means that the communication would take more time.

In the present parallel implementation on cell-centred finite volume based unstructured-grid solvers, flow variables of the elements that lie on both sides of a block boundary are required to be updated at each synchronisation point. As described in chapter 3, ghost elements are employed as the receiving buffer to store the flow data from the other side of the boundary. In addition, the flow data on the nodes that lie on the block boundary have to be updated to keep the interface consistence across the flow field. The exchanging of data between different zones on every synchronization point is done by message passing over the network.

The message is a package of data that consist of message type, destination, message length and actual message data. Depending on cases, the length of the message can vary from a few bytes to several Megabytes. The transmission time could be ranging from several hundreds of microseconds to several seconds on a typical Ethernet compared to be normally several microseconds for random memory access on a shared memory parallel system. Therefore, minimization of communication for parallel computing on a distributed system is vital for performance.

6.5 Parallel Implementation on Distributed Systems

The implementation of distributed-memory explicit message passing parallel computing on unstructured-grid flow solvers has been discussed extensively in references (Venkatakrishnan, Simon et al. 1991; Mavriplis 2000) In this section, we

focus on issues of the present multi-block parallel implementation, including the data structure, block interface treatment and message passing pattern.

6.5.1 Data Structure and Interface Treatment

A very important issue in the parallel computing is the interface treatment, i.e. the block boundary treatment, which is vital to accuracy and stability of a solution.

A partition or block interface is where two blocks are next to each other. Figure 6.3 depicts block interfaces between three 2D unstructured meshes. It is clear that faces on an interface are shared by two blocks, and a point may be shared by more than two blocks depending on its location. After partitioning, these points and faces are duplicated and distributed to corresponding blocks, which are often associated with processors across the network. This information must be stored to be available to relevant flow solvers. In the present edge-based data structure implementation, a set of interface arrays is declared to store this information. An interface node array is allocated to store IDs of nodes on interfaces in this partition, corresponding node IDs in the other block, ID of the partition, and how many times this point has been duplicated. An interface edge array is used to store the current face IDs in this block, corresponding face IDs in the other block, and the ID of the other block.
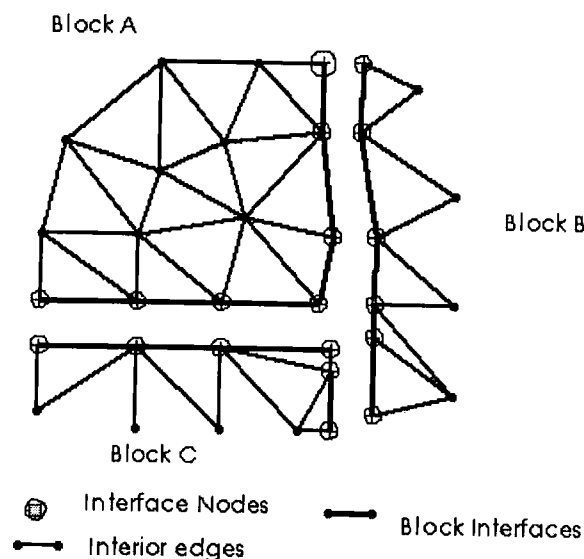
Figure 6.3 Block interface

Block 1
P1

A   B'

P2

Block 2
P1

A'   B

P2
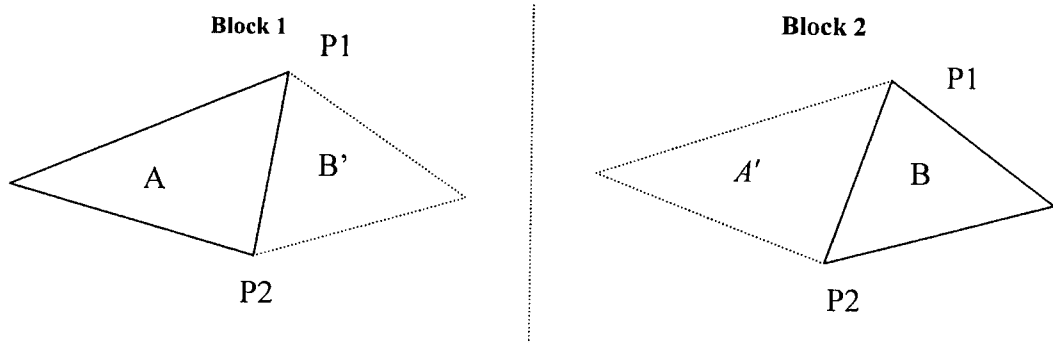
Figure 6.4 Interface treatment

In parallelised flow solvers, these block boundaries are treated as internal faces, i.e. using the same way to evaluate fluxes across these faces. At the pre-processing stage, the contribution (used for the weighted averaging procedure) of each block to boundary nodes is calculated and sent to corresponding blocks. At each synchronization point, there are two phases. First, the flow variables of the nodes that lie on a block boundary are updated by sending and receiving messages. As shown in Figure 6.4, flow variables of nodes P1 and P2 are updated by message passing at this stage. Then, the flow variables near the boundary face are evaluated using formulae described in 3.4.5 and sent to the other block. In Figure 6.4, the flow variables near P1-P2 in element A is used to update the values of A' in block 2.

A synchronization point is placed immediately after the boundary condition treatment, so that the information of either side of the block could be updated with the aid of PVM message passing API.

6.5.2 Message Passing Pattern

Message passing is a programming model for interactions between processors within a parallel system. In general, a message is constructed by a program on one processor and is sent through an interconnection network to another processor, which then must accept and act upon the message contents. Thus, message passing can yield high efficiency making it a very effective way to transmit a large block of data from one

processor to another. However, the overhead in handling each message including latency could be high, which could lead to a major dropping in performance. In order to minimize the need for expensive message passing operations, data structures within a parallel program must be spread across the processors so that most data referenced by each processor is in its local memory. This task is known as mesh partitioning in CFD terms.

The actual cost of transmission of a message can often be modelled by a linear relationship,

$$T_C = T_L^s + T_M + \alpha M + T_L^d \tag{6-5}$$

where T_L^s and T_L^d are the communication latency of the sending/receiving system, T_M is the cost for preparing the message (including copying, packing and unpacking) that is linearly related to the length of the message, α is the rate of bandwidth for data transfer between two processors and M is the message length.

In a parallel system, the total cost of N message transmission on every synchronization points will be $N \cdot T_c$. Because T_L and α can be considered as constants and T_M is only related to the message length, one can either reduce the total message numbers or the message length to reduce the total communication cost. For a given partitioned mesh, the total communication volume is constant. Reducing the number of messages to be transmitted becomes an obvious choice to reduce the overall communication cost.

In the present 3D flow solvers, a 3D block (sub-grid) may consist of a number of triangular or rectangular faces. Each element of these boundaries contains three (triangle) or four (rectangle) nodes, two neighbouring element IDs (one in the current block and one in the other block) and the neighbouring block ID. At each synchronization point, the flow variables of the boundary elements in the current block should be sent to its neighbouring blocks to overwrite values in ghost elements. For a typical 3D block, it may consist of several thousands of such boundary faces.

That means equal amount of messages are required to be sent at each synchronization point. As we mentioned previously, the latency of a PC cluster system is very high, one can expect a very low efficiency outcome with such a scheme. To reduce the number of messages to be transmitted, all the boundary faces sharing the same neighbouring block are collected and compiled to several sending buffers and receiving buffers prior to any communication. Each sending buffer contains the destination block ID and IDs of the elements in the current block to be sent. Each receiving buffer contains the block ID that the incoming message is expecting and the IDs of ghost elements that the message data should be unpacked for.

At each synchronization point, the flow variables of these in the send buffer are sent to their destination process according to their neighbouring block ID. When a message is received, the destination ID is checked and unpacked according to the receiving buffer. In this manner, each block only sends and receives the number of messages equal to the number of their neighbours. Thus, the overhead due to latency of the network is reduced.

6.5.3 Parallel Programming in a Cluster System

In parallel computing of a CFD problem on a cluster system, the partitioned sub-domains (blocks) are distributed across processors in the cluster system. During the parallel execution, inter-communication between blocks to determine block boundary conditions at each time integration step is required. In a system with shared memory, one can simply "copy" the memory from a block boundary array to its corresponding zone boundary array. This procedure is very simple and efficient because of the high bandwidth and low latency of the system memory. Unfortunately, this method cannot apply to a cluster system with distributed memory. For a cluster system, the communication is implemented using the PVM message passing library and the inter-processor communication pattern is pre-determined at run time.

MIMD is a flexible programming model. It allows different programs working together. In the present parallel implementation, a master/slave programming model

is adopted. The master process is a control process with a range of the responsibilities, including:

1. Pre-processing. The master process reads the computing mesh and configurations for parallel computing, adds computing nodes that are not currently in VM.

2. Partitioning the computing mesh. After reading the computing grid and partitioning instructions, the master process coverts the mesh to its compatible graph, partitions the graph and then maps the partitioned graphs to the computing meshes.

3. Initialising computing processes across the network and detecting faulty nodes. After computing processes are spawned, partitioned computing meshes, initial and boundary conditions are sent to corresponding processes in the network by the PVM message passing library.

4. Sending the initial data to computing processes and receiving the final result from them. When the solution is converged, computing processes are terminated by the control process.

There are two types of flow solvers serving as slave computing process: Tetra3D and Prism3D. The Tetra3D is a tetrahedron based flow solver and the Prism3D is prismatic element based. Each of the solvers can be run in the parallel mode or single process mode. After a flow solver is launched, it will try to get its VM parent process id. If the parent id is -1, it indicates no VM parent is present and it turns into the single process mode. In single process mode, it will turn off any parallel subroutines. When in parallel mode, it will receive the partitioned computing mesh from the control process and send/receive any messages at each synchronization point. The following flow chart (Figure 6.5) represents a simple parallel computing job with one tetrahedron block and one prismatic block, in which the dash lines represent communication among processors.

Master3D

Tetra3D   Prism3D

```
                              ┌──────────────────────┐
                              │ Read VM and partition│
                              │    configuration     │
                              └──────────┬───────────┘
                                         ↓
                         ┌───────────────────────────────┐
                         │ Add hosts to VM, partition the mesh │
                         └───────────────┬───────────────┘
                                         ↓
┌──────────────┐         ┌───────────────────────────────┐         ┌──────────────┐
│ Initialising │         │  Spawn computing processes    │         │ Initialising │
└──────┬───────┘         └───────────────┬───────────────┘         └──────┬───────┘
       ↓                                 ↓                                 ↓
┌──────────────────┐      ┌───────────────────────────────┐      ┌──────────────────┐
│ Receiving the mesh│ ◄····│   Send computing meshes and   │····► │ Receiving the mesh│
│ and configurations│      │   computing configuration     │      │ and configurations│
└──────┬───────────┘      └───────────────┬───────────────┘      └──────┬───────────┘
       ↓                                                                 ↓
┌──────────────────┐ ◄·······························► ┌──────────────────┐
│ Pre-processing and│                                  │ Pre-processing and│
│validating block-boundary│                            │validating block-boundary│
└──────┬───────────┘                                   └──────┬───────────┘
       ↓                  ┌───────────────────────────┐       ↓
┌──────────────┐ ◄········│   Start parallel computing │·······► ┌──────────────┐
│Start computing│         └───────────────┬───────────┘         │Start computing│
└──────┬───────┘                          ↓                     └──────┬───────┘
       ↓                                                                ↓
┌──────────────────┐ ◄·······························► ┌──────────────────────┐
│Boundary conditions &│                               │Boundary conditions & Block│
│block interface treatment│                           │  interface treatment │
└──────┬───────────┘                                  └──────┬───────────────┘
       ↓                  ┌───────────────────┐               ↓
┌──────────────┐    ────► │  Wait for messages │         ┌──────────────┐
│Flux evaluation│         └───────────────────┘         │Flux evaluation│
└──────┬───────┘                  ↓                     └──────┬───────┘
       ↓                                                        ↓
┌──────────────┐                                         ┌──────────────┐
│Time marching │                                         │Time marching │
└──────┬───────┘                                         └──────┬───────┘
       ↓                  ┌───────────────────┐                 ↓
┌──────────────┐ ·······► │  Receive residual  │ ◄······· ┌──────────────┐
│Send residual │          └───────────────────┘          │Send residual │
└──────┬───────┘                  ↓                       └──────┬───────┘
       ↓                      ◇ End ◇                            ↓
     ◇ End? ◇ ◄········ ┌───────────────────┐ ········► ◇ End? ◇
                        │ Send stop to computing│
                        │     processes     │
                        └───────────────────┘
┌──────────────┐ ·····► ┌───────────────────────────┐ ◄····· ┌──────────────┐
│Send final results│    │Receive and output final results│  │Send final results│
└──────┬───────┘        └───────────────┬───────────┘        └──────┬───────┘
       ↓                                ↓                            ↓
    ( END )                    ┌───────────────┐                 ( END )
                               │     Stop      │
                               └───────────────┘
```

Figure 6.5 Flow chart of parallel computing

Chapter 7

2D Validation and Discussion

In this chapter, Euler/Navier-Stokes algorithms based on unstructured-grids developed in the previous chapters are examined on a range of 2D flow cases. These cases are served to validate the solution algorithms for inviscid, laminar and turbulent flows, and to assess accuracy and efficiency of these methods.

The basic algorithms developed in Chapter 3, including the upwind scheme, explicit multi-stage Runge-Kutta method, spatial second order scheme and cell-centred finite volume method, are examined and validated on several selected inviscid flow cases. The results are compared to analytical and experimental data to assess overall accuracy of the algorithm. Several laminar and turbulent flow cases are presented to validate the viscous term treatment and to confirm the correct implementation of the turbulent model. The inflation mesh generation technique developed in Chapter 4 is demonstrated in several turbulent flow cases.

The adaptive mesh refinement technique is demonstrated in both inviscid and viscous turbulent flow cases to assess accuracy of this method. The efficiency of the multigrid method developed in Chapter 5 is examined in both inviscid and turbulent flow simulations. The results of turbulent flow simulations also serve to demonstrate the effectiveness of the present aspect-ratio adaptive multigrid method.

7.1 Results for Euler Algorithm

In this section, the Euler equations are solved on a cell-centred finite volume setting using an upwind scheme on several inviscid flow test cases. The first case is subsonic flow in a 2D channel with a bump. This case is designed to demonstrate the overall accuracy of the second spatial order scheme. The second test case is the simulation of transonic flow over a NACA 0012 airfoil to examine the efficiency and accuracy of the mesh adaptation technique. The third case is transonic flow around a RAE 2822 airfoil. This case is to demonstrate the effectiveness of the multigrid in inviscid flow simulations.

All numerical results presented in this section are obtained using the spatial second order scheme as described in Chapter 3. Roe upwind scheme (3.4.3) is used to evaluate fluxes across internal edges. The solutions are advanced to steady states by using a four-stage explicit Runge-Kutta time integration with local time stepping.

7.1.1 Subsonic flow in a 2D Channel

The first test case is a 10% thick circular arc bump in a channel. The flow approaches at a zero incident with an inlet Mach number of $M_{inlet} = 0.5$. This case has been widely used by many researchers due to simplicity of the geometry (Allmaras 1989).
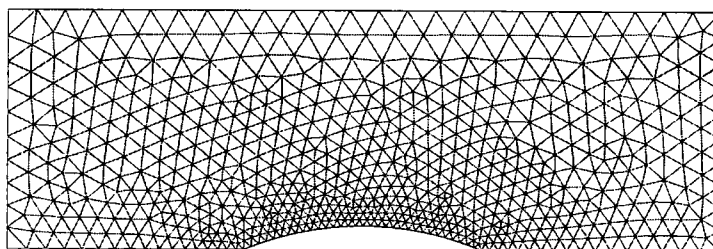
Figure 7.1 Unstructured-grid in a 2D channel

Figure 7.1 shows the computational unstructured-grid used in this case, which is generated by the mesh generator developed by the author (Zheng 1995). The mesh contains 699 points and 1,285 triangles. The flowfield is completely subsonic with stagnation points at the leading and trailing edge of the bump as shown by Mach

number contours (Figure 7.2). Figure 7.3 is the distributions of Mach number and pressure (non-dimensionlised) on the upper and lower walls (Figure 7.3).
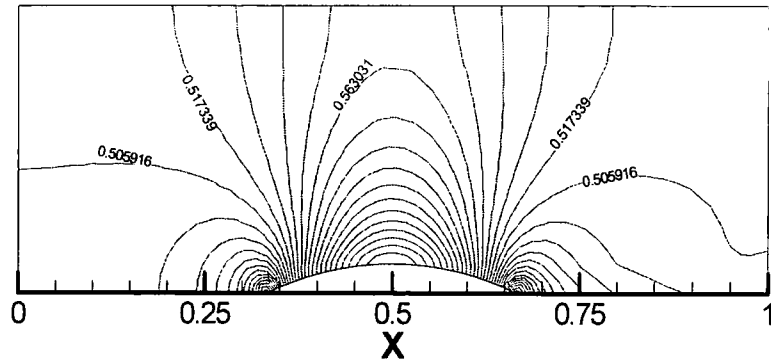
Figure 7.2 Mach number contours
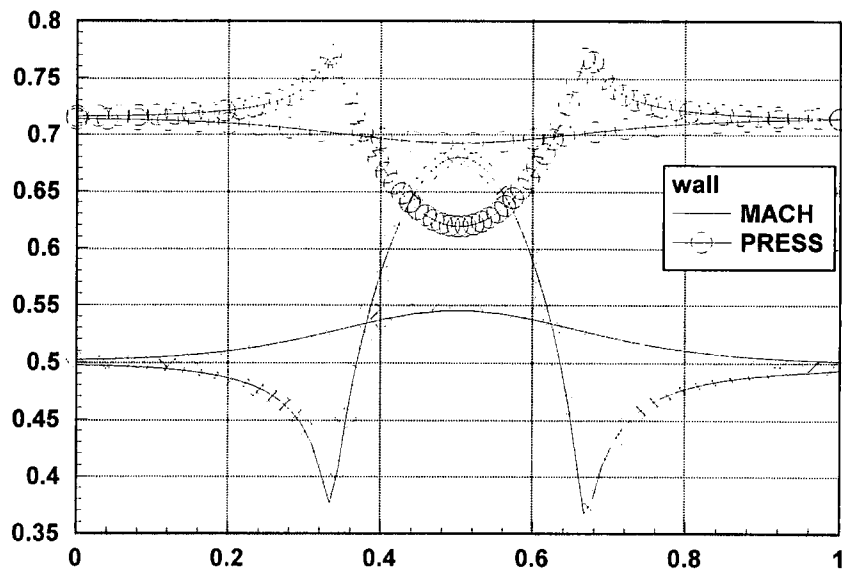
Figure 7.3 Mach number and Pressure distributions on the wall

In this case, the overall quality of the solution is indicated by symmetry distribution of the flowfield about the bump. Reasonably good symmetry of Mach number distribution is observed in Figure 7.2 and 7.3, except from the bump trailing edge to exit plane due to the propagation of the numerical dissipation. It is also noticed that the simple reflected exit boundary condition treatment (3.8) also contributes to the non-symmetric distribution near the exit plane.

7.1.2 Transonic Flow over a NACA 0012 Airfoil

The second test case is the transonic flow over a NACA 0012 airfoil. This case is designed to examine the shock wave capturing ability of the present 2D unstructured flow solver and to assess the accuracy of the solution with the mesh adaptation technique.

The flow over the NACA 0012 airfoil approaches at a free stream Mach number of 0.8 with an incidence angle of $1.25°$. The far field boundary is approximately circular and placed at a distance of five chords away from the airfoil to minimise the disturbance. Figure 7.4 shows the initial coarse unstructured-grid, which contains 581 nodes and 1,097 triangles.
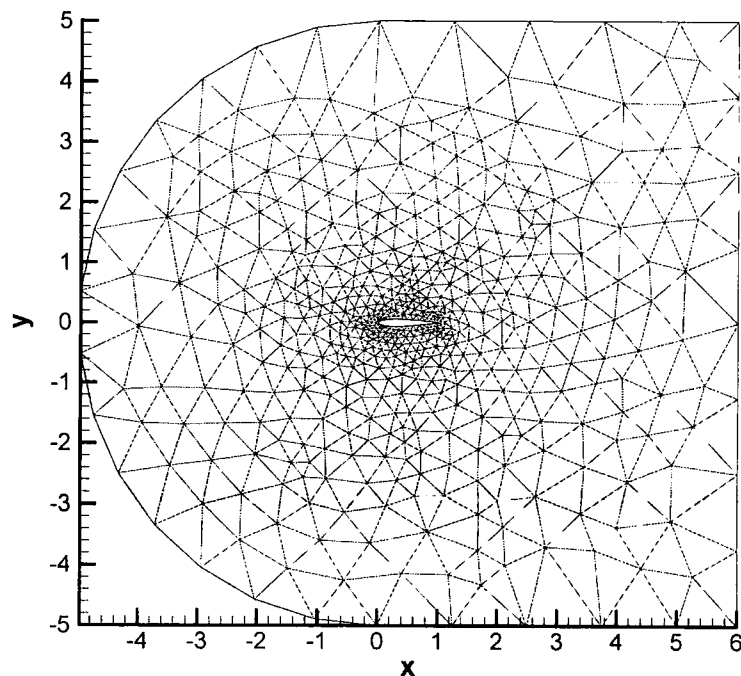
Figure 7.4 The initial computational mesh around NACA 0012

The computing begins on the coarse mesh (Figure 7.4). The mesh adaptation is enabled and the refinement criterion is set to static pressure because of the inviscid nature of the flow. The Direct Connected Multigrid (DCMG) is used to accelerate the solution. After three times of successive mesh adaptations, the solution reaches a

steady state. The sequence of the meshes used in the different phases of the computation is plotted in Figure 7.5. The mesh near the airfoil is refined repeatedly due to the high pressure gradients in this region. High density of the mesh on the upper surface regions indicates the presence of a strong discontinuity. Most of the triangles are well formed due to the mesh relaxation procedure after each refinement.

(a) Initial mesh around the airfoil   (b) Refined mesh 1

(c) Refined mesh 2   (d) Final mesh
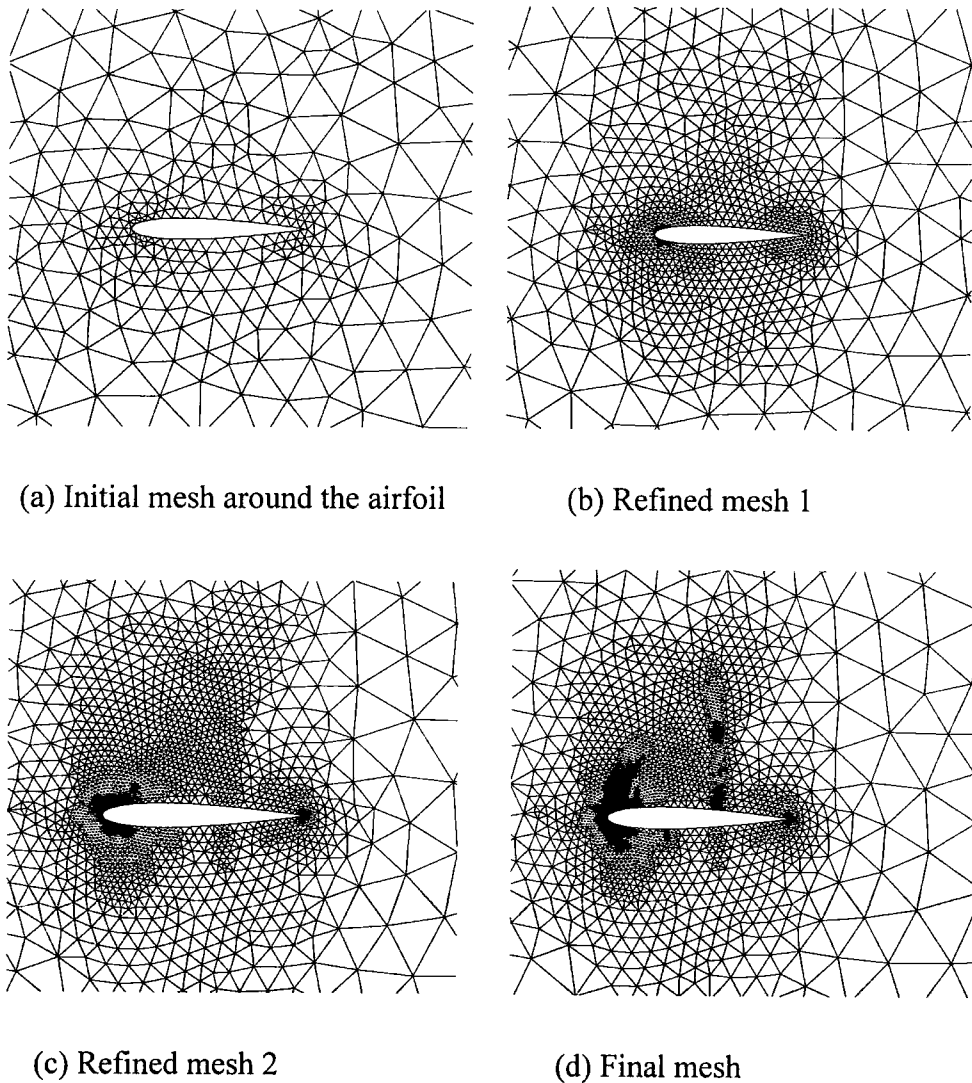
Figure 7.5 Sequence of the unstructured meshes

Figure 7.6 is the plot of Mach number contours of the flowfield. Both the strong shock on the upper surface and the weak shock on the lower surface are well resolved. This is due to the mesh refinement in these regions, as depicted in Figure

7.5. It is often very important to investigate the changes of entropy ($\varepsilon = c_v \ln \dfrac{p}{\rho^\gamma}$) of a solution to determine the accuracy of a scheme. For convenience, the entropy function ($S = \dfrac{p}{\rho^\gamma}$) is used to model the entropy increase ($\dfrac{S}{S_{inlet}} - 1$). Figure 7.7 shows the entropy increase contours. As we know, entropy increase should be zero for an inviscid flow upstream of the shock wave. As can be seen in Figure 7.7, the computed values near the leading edge are below 0.5%, which is a good indication of local accuracy of this solution. Adaptive mesh refinement is evident in the region of the leading edge, and in the vicinity of both the upper shock and the lower weak shock waves. The distribution of the surface pressure coefficient ($\dfrac{p - p_\infty}{\frac{1}{2}\rho v_\infty^2}$) shows very good agreement with the structured grid calculation of Ref. (Anderson et al. 1986) in Figure 7.8.
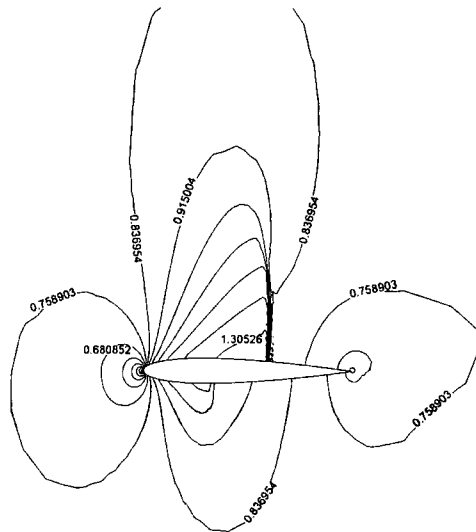
Figure 7.6 Mach number contours
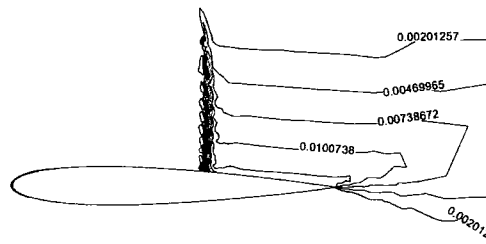
Figure 7.7 Entropy increase contours

The convergence history of the entire calculation is shown in Figure 7.9. After each adaptive refinement, flow variables are interpolated to the new mesh and a series of multigrid levels is built up based on this new mesh. With 3 levels of multiple grids (DCMG) and 3 stages of adaptive mesh refinement, the solution procedure converges very rapidly. This case takes about 3 minutes on a Pentium II 450 PC. It is evident that the present multigrid is effective for the inviscid flow simulation.
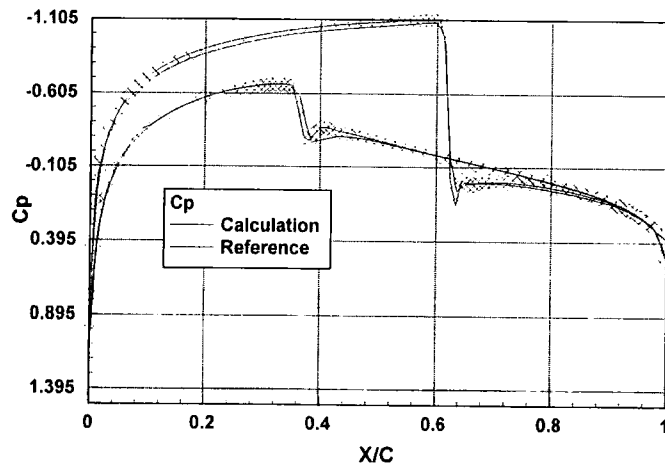
Figure 7.8 Pressure coefficient distribution on the airfoil
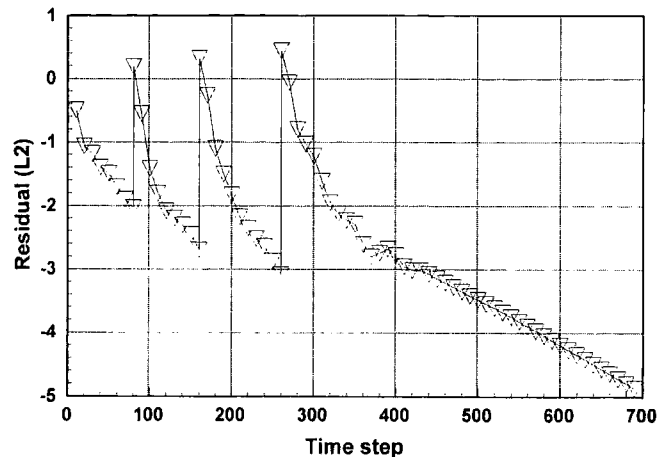
Figure 7.9 Convergence history

7.1.3 Transonic Flow over RAE 2822 Airfoil

The final test case for the Euler algorithm is transonic flows over a RAE 2822 airfoil, which is developed by Cook et al (Cook et al. 1979). A range of experiments has been carried out at the Royal Aircraft Establishment, UK. In the present study, a transonic flow case corresponding to the experiment case 10 is simulated.

In this case, the flow approaches the airfoil at $M_\infty = 0.75$ with an incidence of $\alpha = 3.19°$. In the experiment, a strong shock wave located at about 53% chord of the upper surface is observed. The main objective of this case study is to assess the effectiveness of the multigrid method for inviscid flow simulations.

The computational mesh (Figure 7.10) for this case consists of 5,217 points and 10,320 triangles. It is generated by the 2D mesh generator in the GRUMMP (Ollivier Gooch 1998). The mesh adaptation function is disabled in this case because the main objective of this case is to examine the effectiveness of the present multigrid method. Three levels of multigrid, as plotted in Figure 7.11, are used to accelerate the inviscid solution. These coarse meshes are generated with the DCMG method based on the connectivity of triangles.
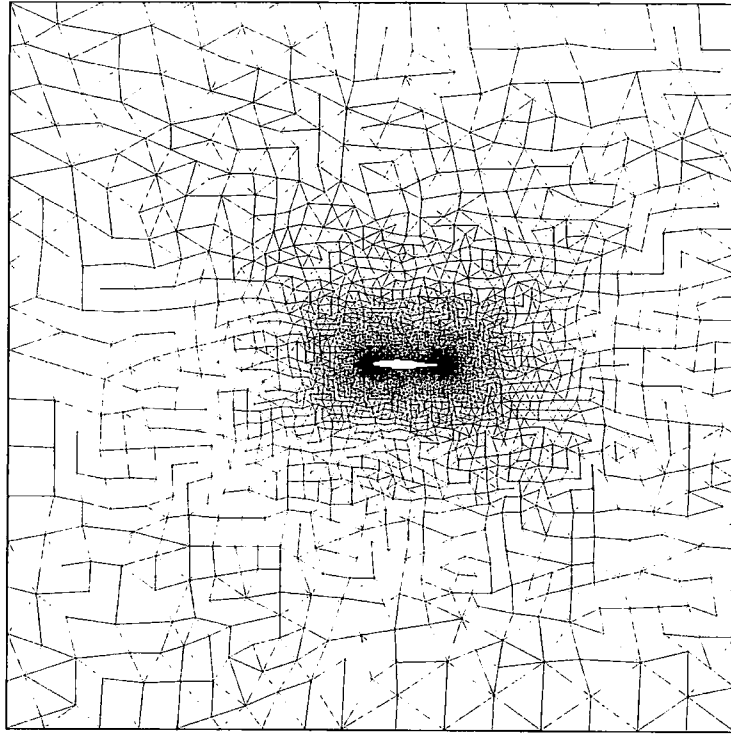
(a) Full field view of mesh
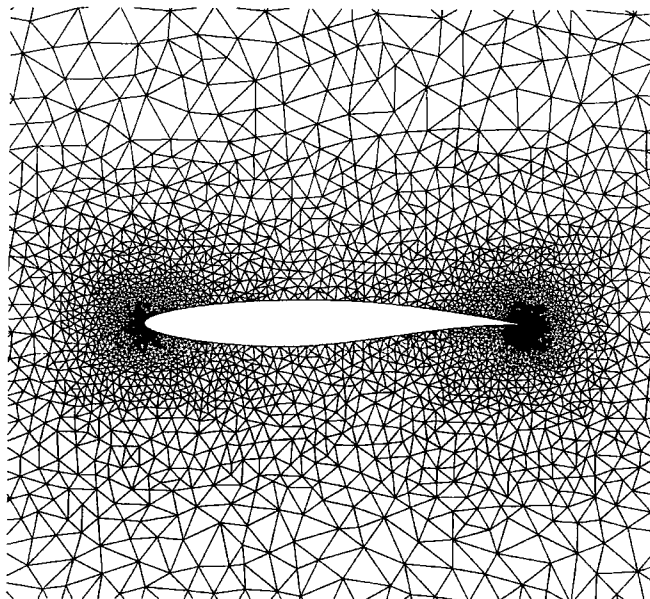
(b) Unstructured-grid near the airfoil

Figure 7.10 Computational mesh around RAE 2822 airfoil
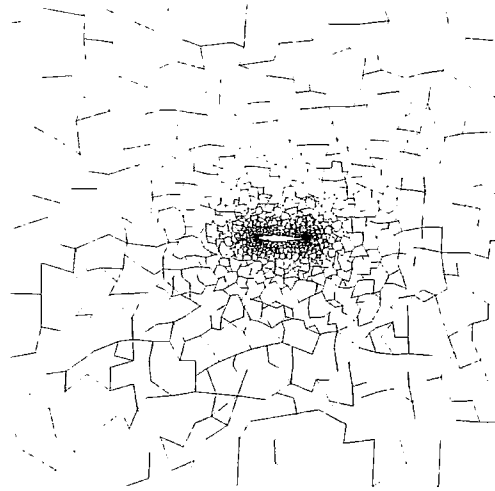
(A) First level of coarse meshes

(B) Second level of coarse meshes   (C) Third level of coarse meshes

Figure 7.11 Sequence of coarser levels

Two computations with the same flow conditions are performed: a single grid and a multigrid. The unstructured mesh used in both computations is plotted in Figure 7.10. The final flowfield for both computations are identical. Figure 7.12 is the plot of Mach number contours. The incoming flow is accelerated to supersonic on the upper surface of the airfoil and a strong shock is formed at around 70% of the chord. Figure 7.13 shows the comparison of the pressure distribution on the airfoil with experimental data (Cook et al. 1979). The pressure coefficient shown in the plot is defined as $C_p = \dfrac{p - p_\infty}{\frac{1}{2}\rho v_\infty^2}$. With this definition, C_p could exceed 1.0 in the leading

edge area. The pressure on the upper surface is in reasonable agree with experimental data before the shock wave, but it is clear that the inviscid solver fails to predict the location of the shock wave correctly. This highlights the importance of viscous effects in transonic flow simulations. Figure 7.14 is the comparison of the convergence histories of the single grid and multigrid computations. With three levels of coarse meshes, the multigrid converges within 1000 steps and it takes a much longer time, around 8500 steps, for the single grid to reach a steady solution.
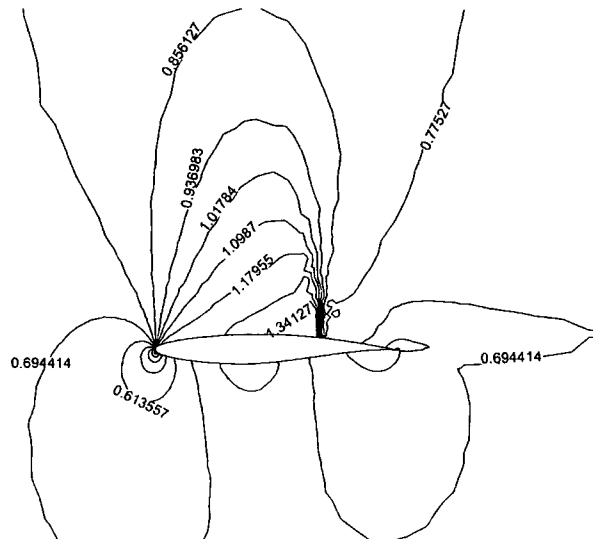
Figure 7.12 Mach number contours
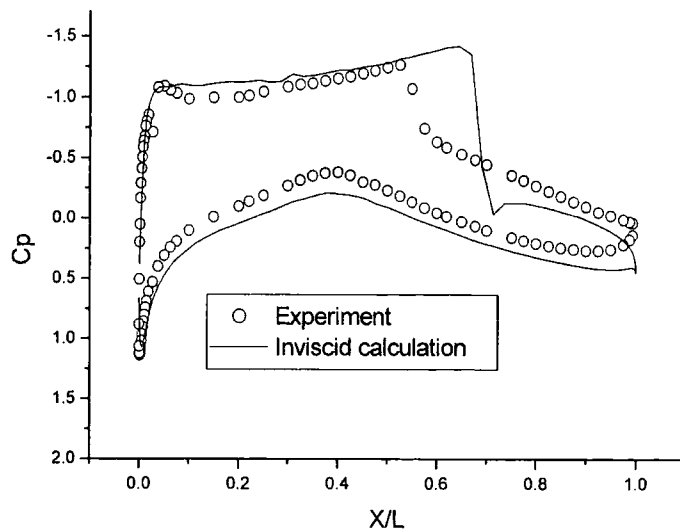
Figure 7.13 Comparison of the surface pressure distribution
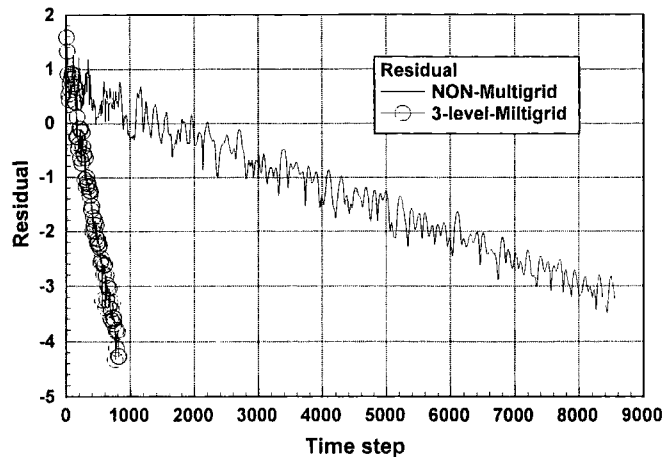
Figure 7.14 Comparison of Convergence histories

Although both computations fail to predict the location of the shock wave as expected, very good acceleration is observed in the multigrid computation. With three levels of coarse meshes, the solution converges in 20 minutes on a PII 450 PC. It is evident that the direct connectivity based multigrid can deliver up to 7 times of speedup in inviscid computations.

7.2 Laminar Flows over a Flat Plate

In this section, the simulation of the flat plate boundary layer is carried out to demonstrate the accuracy of the viscous term treatment and to examine the influence of the mesh with a high aspect ratio on the present 2D unstructured solver. The computations are carried out on a triangular based unstructured mesh (Figure 7.15a) for $M_\infty = 0.2$ at various Reynolds numbers. The accuracy is assessed by comparing the flat plate flow solution with the Blasuis similarity solution, which can be regarded as the exact solution for incompressible laminar boundary layers on a flat plate.
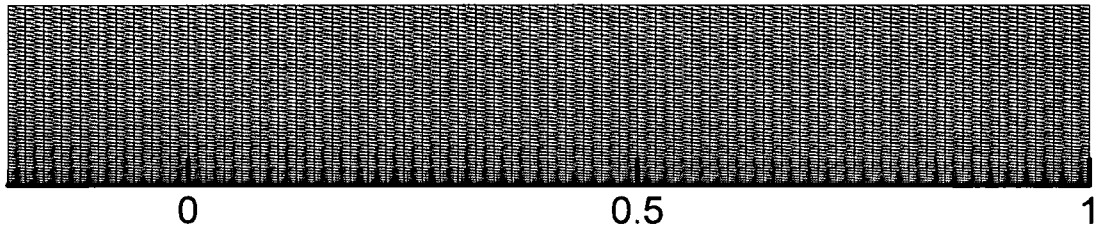
(a) Full field view of the unstructured-grid
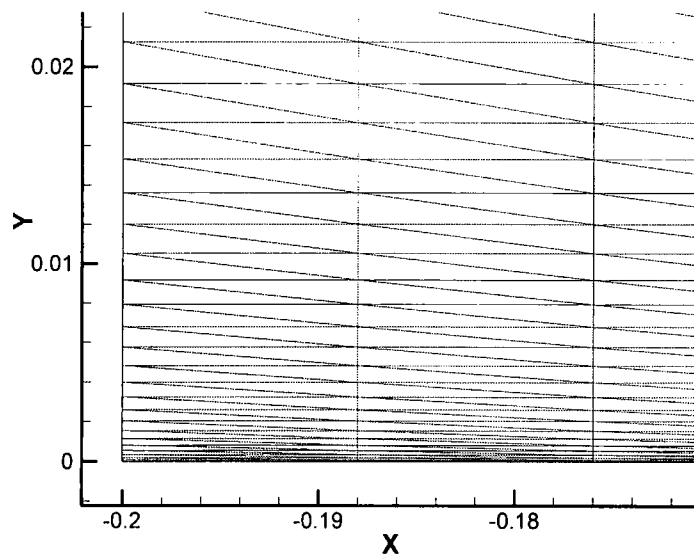
(b) Unstructured-grid near the wall

Figure 7.15 Computational grid for the flow over a flat plate

The unstructured mesh (Figure 7.15) used in the various laminar flow simulations is generated by subdividing a 61x31 H-topology structured-grid. The mesh is uniform in streamwise spacing and highly stretched near the solid wall (Figure 7.15b). The maximum aspect ratio near the solid wall is around 30. In all flat plate boundary layer calculations, the Mach number is kept at 0.2 to minimise the compressibility effect.

The first computation is performed at a Reynolds number of 5×10^3. Figure 7.16a shows the comparison of the boundary layer velocity profile, where η is

$$\eta = y \sqrt{\frac{u_e}{vx}} \qquad\qquad (7-1)$$

and u_e is the velocity of farfield. With 12 grid lines (excluding the first grid line which lies on the wall), the velocity profile is well modelled against the Blasius exact solution. Figure 7.16b is the comparison of the skin friction coefficient c_f,

$$c_f = \frac{\tau}{\frac{1}{2}\rho u^2} * \text{Re}_x \qquad (7\text{-}2)$$

with c_f from the Blasius analytical solution,

$$C_f = \frac{0.6641}{\sqrt{\text{Re}_x}} \qquad (7\text{-}3)$$

and the lcoal Reynolds number based on x is obtained from the reference Reynolds number,

$$\text{Re}_x = \frac{u_e x}{\nu} * \text{Re} \qquad (7\text{-}4)$$

In the skin friction coefficient plot (Figure 7.5b), some disagreements are shown near the leading edge of the plate and the trailing edge. The disagreement at the leading edge suggests some numerical errors or compressible effects, while at the trailing edge, the simple boundary condition treatment is likely to be the reason. This issue can be addressed by adding a buffer zone at the trailing edge to smooth the pressure changes from the inner boundary layer to a free flow condition.

To further investigate the accuracy of the present solution method, numerical simulations are carried out at different Reynolds numbers, Figure 7.17a at $\text{Re} = 1 \times 10^4$ and Figure 7.17b at $\text{Re} = 4 \times 10^4$, respectively. By using various Reynolds numbers, various boundary layer thicknesses can be obtained on one unstructured-grid. The velocity profile agrees well with the Blasius solution when the Reynolds number is doubled from 5×10^3. When the Reynolds number increases to 4×10^4, some disagreement can be observed in Figure 7.17b, which suggests too few grid lines in the boundary layer. It is noted the upwind procedure requires only 9

128

points to resolve the boundary layer well (Figure 7.17a), whilst our experience suggests typically 20 points or more would be needed for central difference schemes with the artificial viscosity.
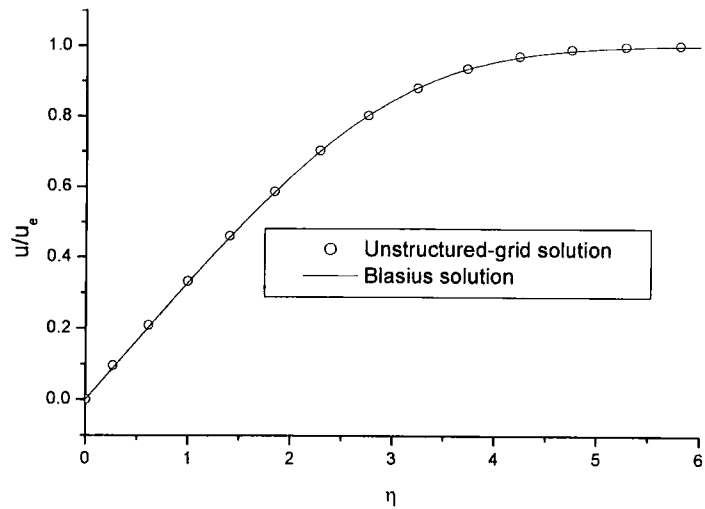
(a) Velocity profile at 50% of the plate (Re=5×10^3)
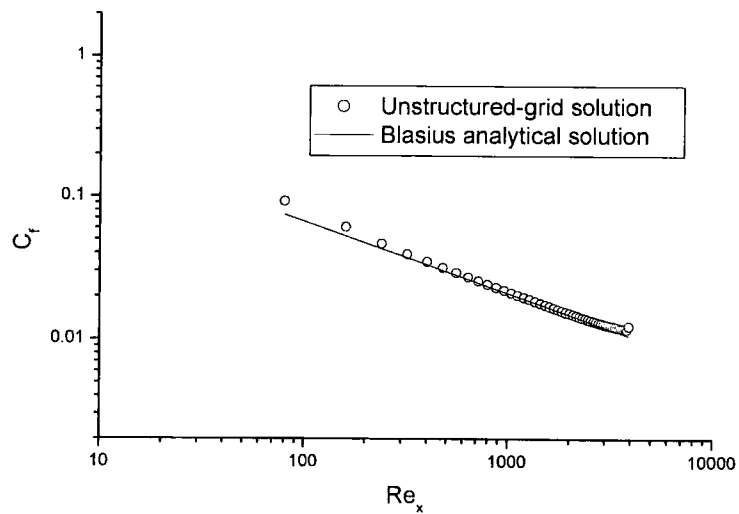
(b) Local wall friction coefficient

Figure 7.16 Laminar flow over a flat plate
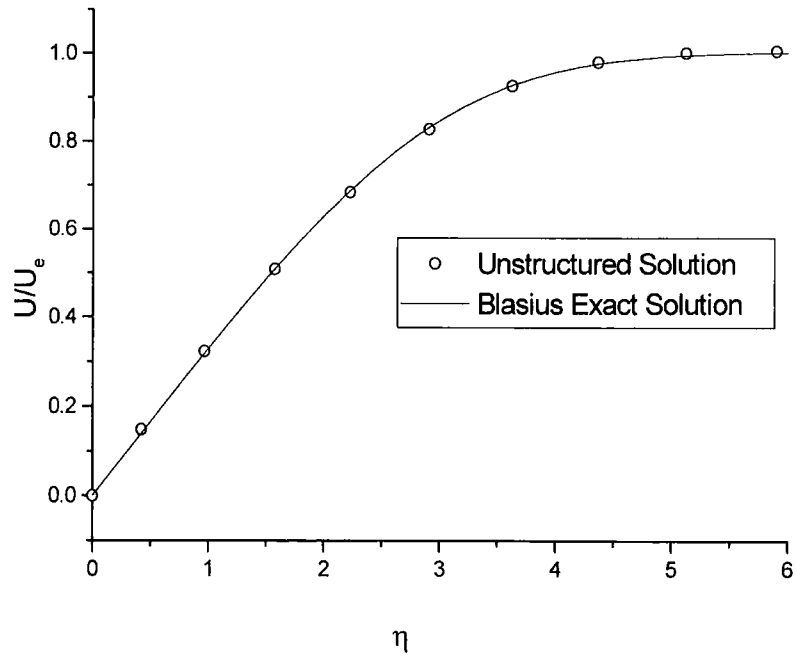
(a) $Re = 1 \times 10^4$
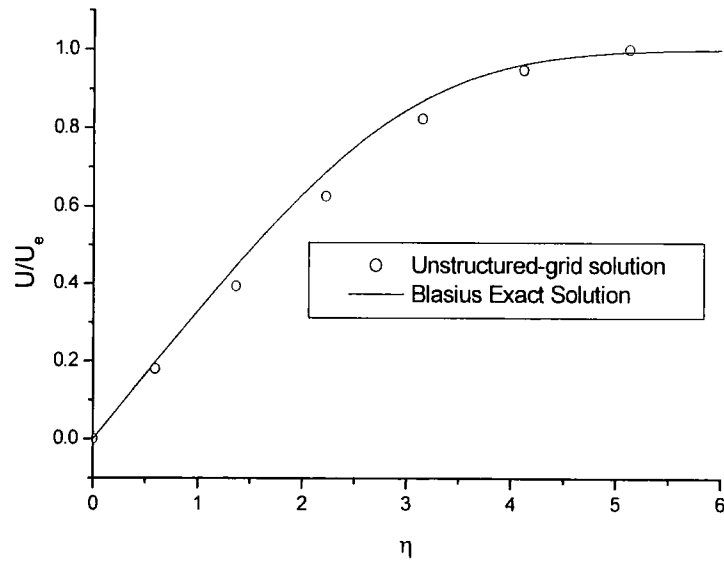
(b) $Re = 4 \times 10^4$

Figure 7.17 Velocity profiles at various Reynolds numbers

7.3 Results for Turbulent flows

In this section, turbulent flows are simulated to validate the correct implementation of the turbulence model and to assess the accuracy and efficiency of the present 2D unstructured flow solver. These include low speed flow over a flat plate, transonic flows around the RAE 2822 airfoil and low speed flows in a linear turbine cascade.

7.3.1 Turbulent Flows over a Flat Plate

The flat plate boundary layer solution serves to confirm the correct implementation of the Spalart-Allmaras turbulence model. Numerical results are compared with an empirical formula: the law of the wall.

In this case, the undisturbed incoming flow is approaching the flat plate at $M_\infty = 0.2$ and $Re_L = 2 \times 10^6$. Because natural transition cannot be predicted by the present model, a trip point is placed at 10% of the plate length to produce transition to turbulent flow. This is done by the transition terms in the turbulence model (3.24). The computation is carried out on the same unstructured mesh used in the previous laminar flow computation. At the farfield, the turbulence dependent variable $\tilde{v}$ is set to 0.001 for numerical reasons. A no-slip condition is applied on the solid wall.
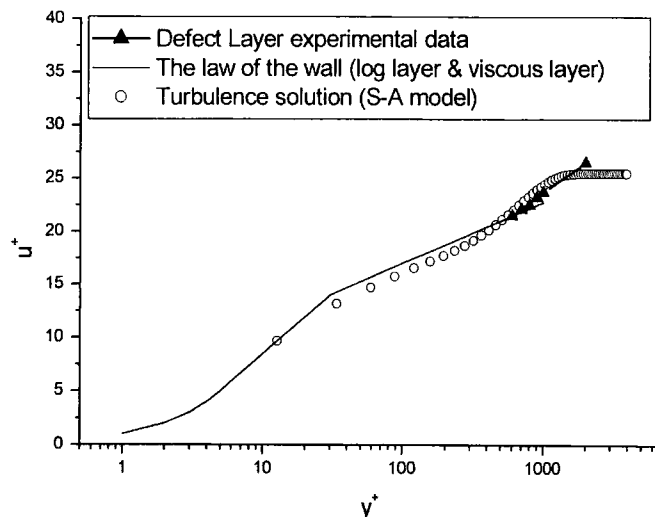
Figure 7.18 Velocity profile against the law of wall

Figure 7.18 shows the result in terms of the velocity profile using the Spalart-Allmaras one-equation model at 50% length of the flat plate. The calculated velocity profile agrees well with the law of the wall. Figure 7.19 is the convergence history of the computation. The residual goes down rapidly before 1000 time step, but slows down after 1000 time step. This indicates that the explicit single grid solver is effective in damping the high order frequency errors and less effective against the low order frequency ones.
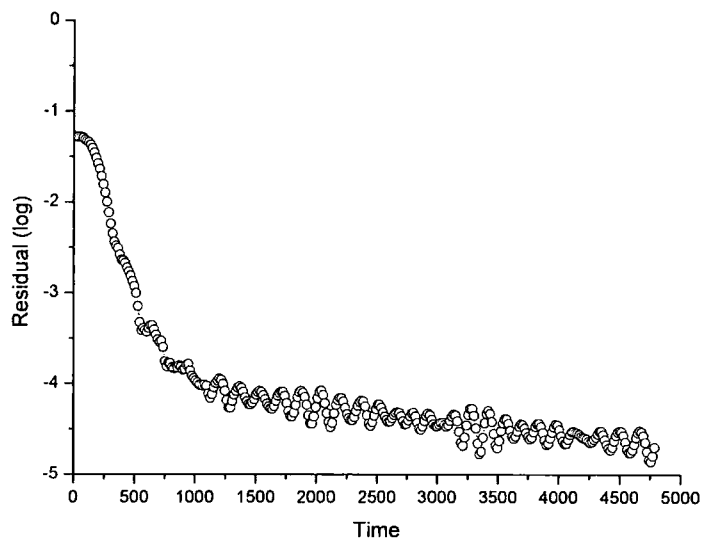
Figure 7.19 Convergence history

In this simplified boundary layer problem, the 2D unstructured-grid based flow solver accurately predicts the velocity profile. This indicates the correct implementation of viscous terms and the turbulence model. The convergence history reveals the need for more effective convergence acceleration means for viscous flow computations.

7.3.2 Turbulent Flow over RAE2822 Airfoil

The second test case is the turbulent flow around the RAE 2822 airfoil at $M_\infty = 0.75$, $Re_\infty = 6.2 \times 10^6$, and $\alpha = 3.19°$. This corresponds to the condition of test case 10 in Ref. (Cook et al. 1979). The similar inviscid computation can be found in 7.1.3, in which the shock wave position has not been predicted correctly. Here the same case is

computed including viscous effects to examine the capacity of the present flow solver to simulate the boundary-layer and shock wave interaction problem as well as to assess the accuracy of the adaptive mesh refinement technique and effectiveness of the present multigrid method in viscous flow simulations.
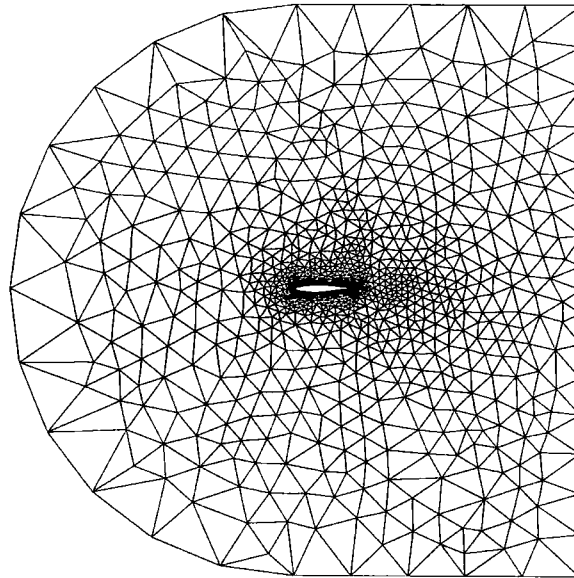
(a) Initial mesh
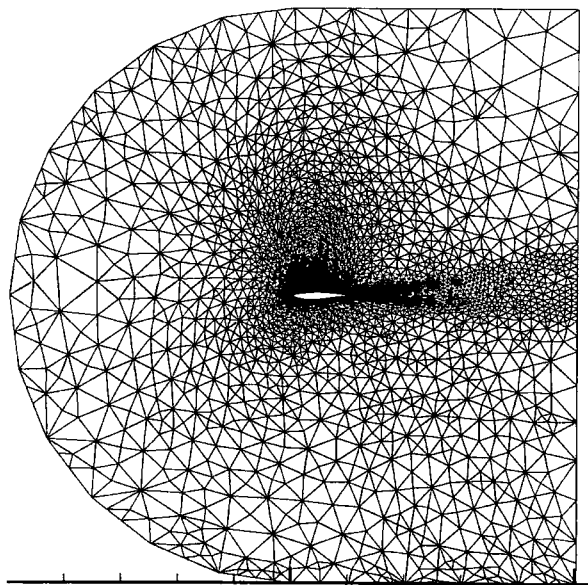
(b) Final mesh

Figure 7.20 Unstructured-grid around the RAE 2822 airfoil

With the mesh adaptation technique, a relatively coarse unstructured grid generated with the Advance Front method is used as an initial mesh, plotted in Figure 7.20a. It consists of 2,329 nodes and 4,491 triangles. The solution is obtained on a fine mesh (Figure 7.20b) with 5,846 nodes and 11,457 triangles after four successive adaptive mesh refinements. It is clear that most element subdividing occurs in the wake region and near the airfoil. The high density of the grid on the upper surface indicates the capture of a strong discontinuity. Figure 7.21 shows a close view of the computational grid near the airfoil. The computational grid in the leading edge region is well formed as shown in Figure 7.21. The mesh near the solid surface is highly stretched (the maximum aspect ratio is about 25) to resolve the boundary layer. The typical y^+ of the first point off the wall is 20 and a wall function is employed.
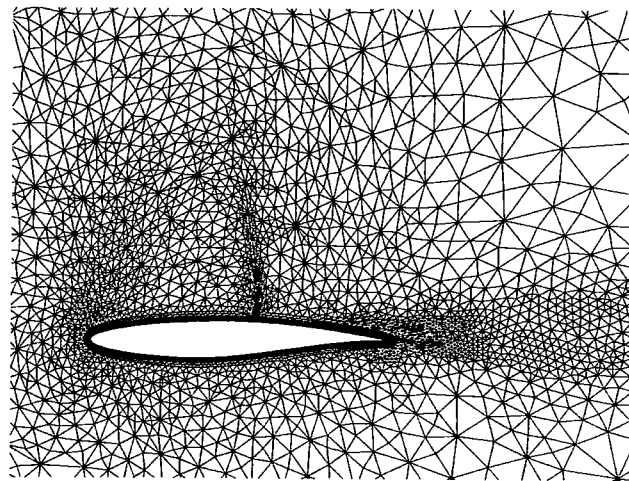
(a) Around the airfoil
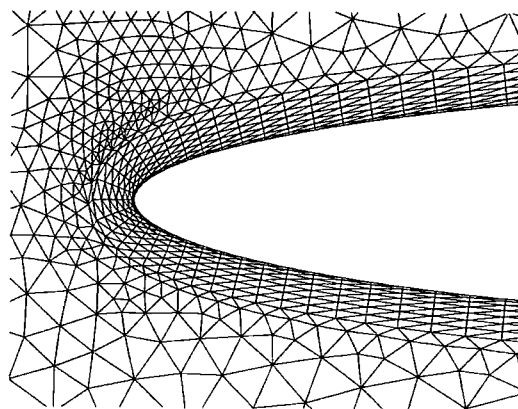
(b) Near the leading edge

Figure 7.21 Computational grid near the airfoil

Figure 7.22 shows the computed result in terms of Mach number contours, demonstrating that the shock wave is well resolved. Figure 7.23 is the plot of eddy viscosity contours near the airfoil. The comparison of the computed surface pressure and experimental data is shown in Figure 7.24. The shock wave location and pressure distribution near the leading edge obtained using the Spalart-Allmaras model agrees well with the experimental results. However, it seems the pressure coefficient is over-estimated after the shockwave. This indicates the separation induced by the shock wave is not well predicted.
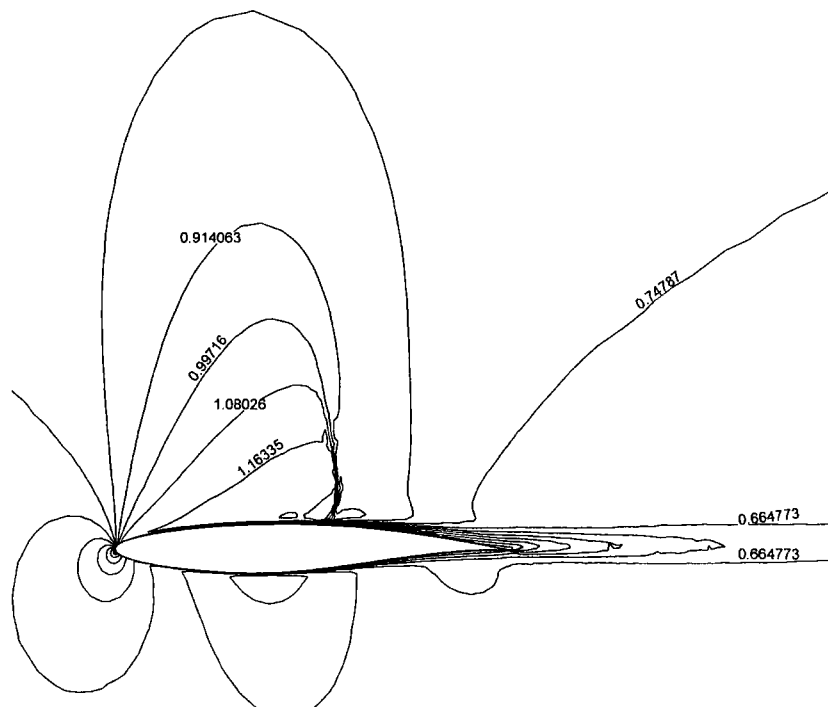
Figure 7.22 Mach number contours
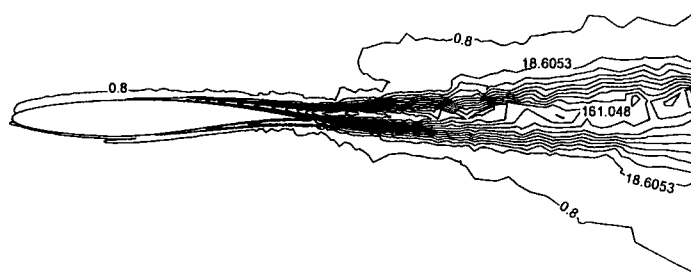
Figure 7.23 Eddy viscosity contours
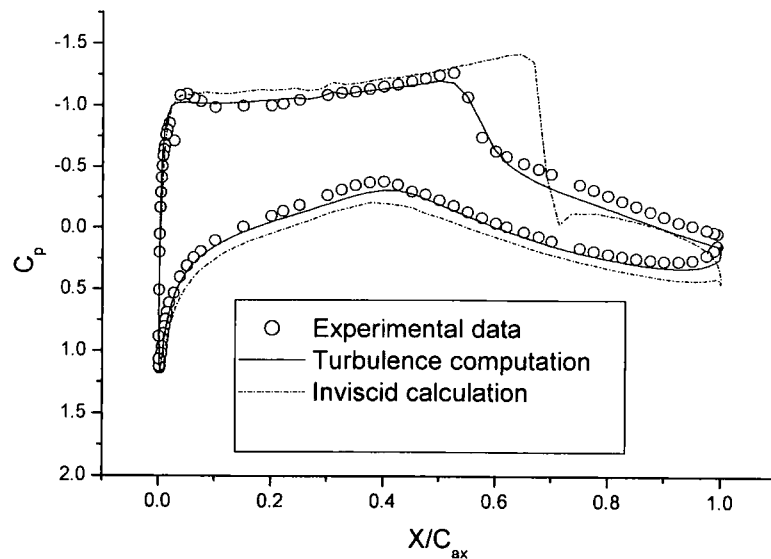
Figure 7.24 Comparison of surface pressure distribution

The effectiveness of the aspect-ratio adaptive multigrid for the high Reynolds number flows is evaluated in this case. The mesh used here is the final refined mesh of the last simulation, and the maximum aspect ratio is about 25. The solutions are obtained in three different ways: Single grid (SG), the Direct Connected Multi-Grid (DCMG) and the Aspect-ratio Adaptive Multi-grid (AAMG). Figure 7.25 shows the meshes of the sequence of AAMG. A four-stage Runge-Kutta scheme is used in all calculations with a fixed CFL number around 1.2. Figure 7.26 shows the convergence histories for the three calculations. It can be seen that the solver with the aspect-ratio adaptive multigrid gives about 6 times speedup compared to the single grid solver and two times speedup compared to the direct connected multigrid solver. The direct connected multigrid can give about 3 times speedup. It should be noted that DCMG and AAMG methods increase by about 12% the computing time per step compared to the single grid because of extra calculation of the fluxes and residuals on coarse levels.

136
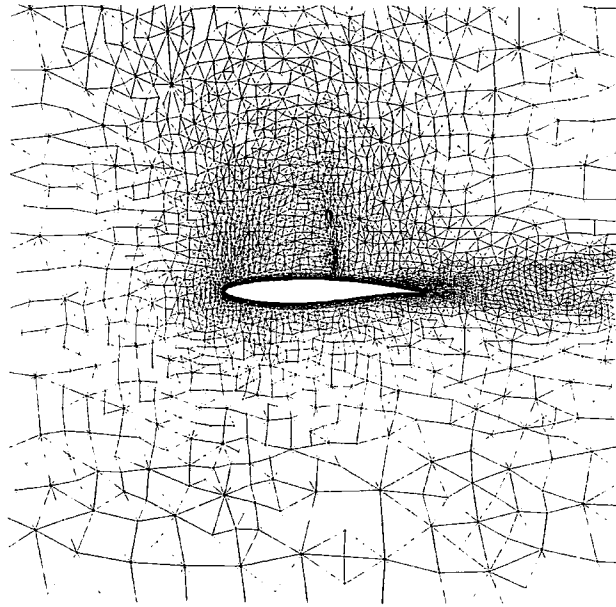
(a) Fine mesh
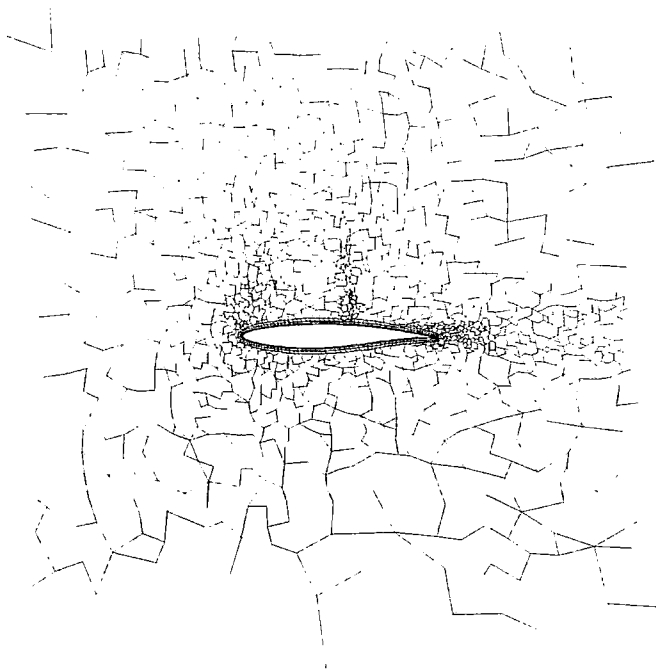
(b) The first coarse level
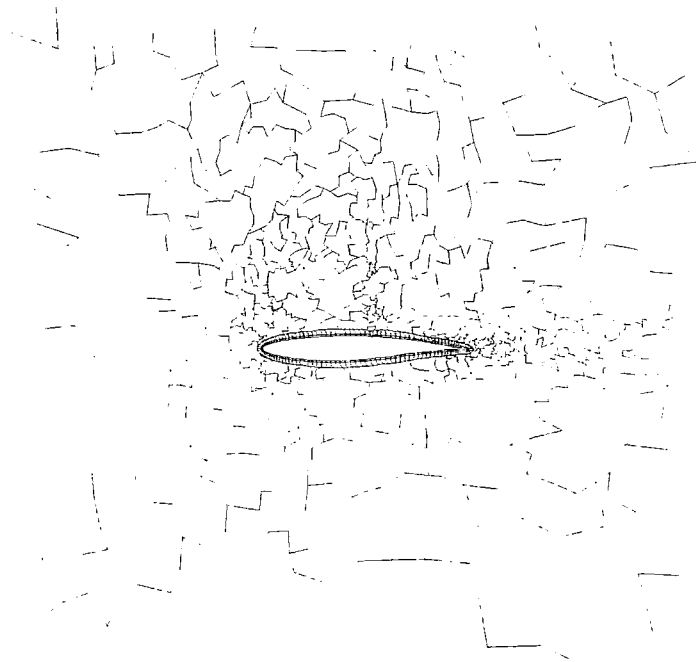
(c) The second coarse level

(d) The third coarse level

Figure 7.25 Multigrid meshes used in AAMG near the airfoil
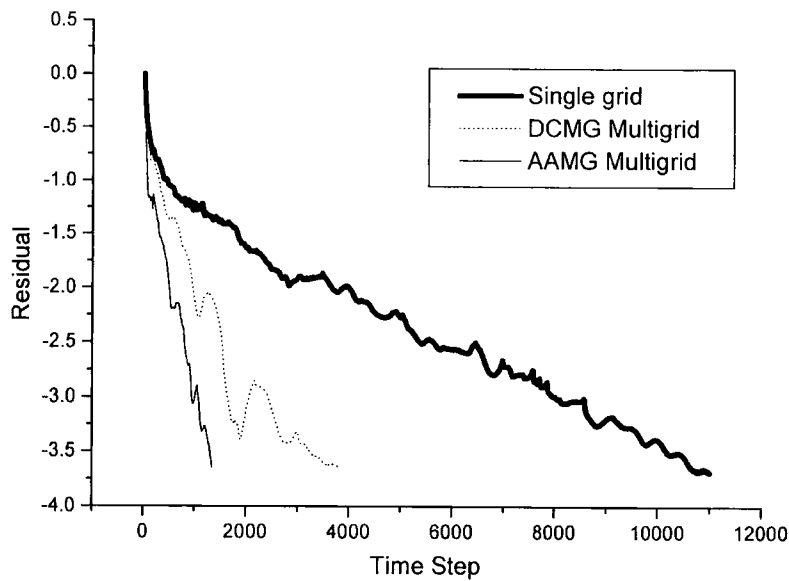
Figure 7.26 Convergence histories

7.3.3 Turbulent Flows in a Turbine Cascade

The simulations of turbulent flows in a turbine cascade have been selected as a test case for the performance and accuracy of the present 2D code when large scale separation is present. He (1998) conducted exhaustive studies using both numerical and experimental methods of this case. In this case, flows in a low-pressure turbine cascade with inlet flow angles of $20°$ and $40°$ are simulated at $Re = 2.2 \times 10^5$. In both cases, a large turbulent separation bubble appears on the pressure surface near the leading edge, and a small laminar separation bubble with transition and reattachment appears on the suction side, as observed in the corresponding experiment (He 1998). In the present calculations, the flow is assumed to be fully turbulent from the leading edge, and no attempt is made to resolve the small laminar bubble on the suction surface.
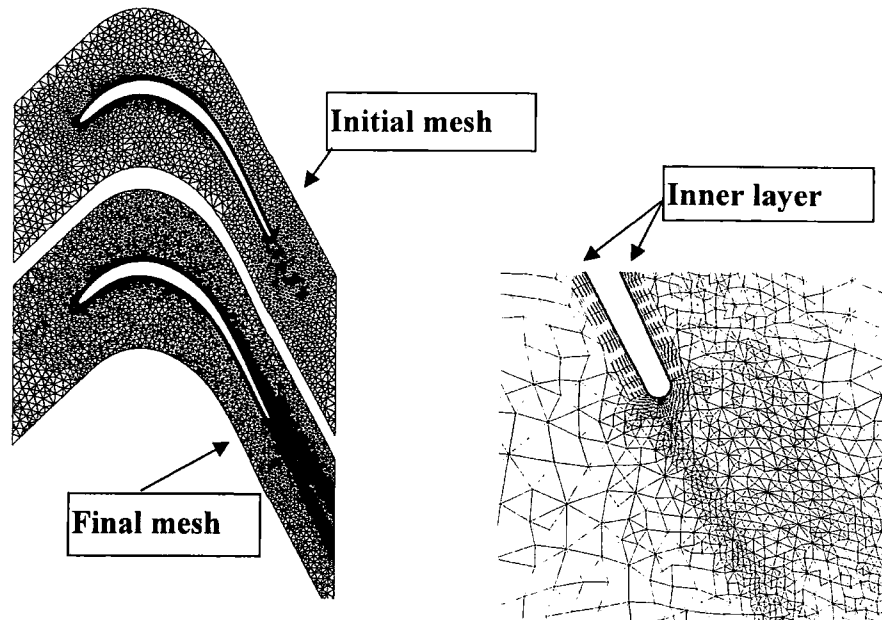
Figure 7.27 Computational meshes around a turbine blade

The initial mesh used in both calculations is generated by the Advancing Front method, as plotted in Figure 7.27. The initial mesh consists of 4,347 nodes and 8,295 triangles. The solution for the case with an 40° incoming flow angle is obtained on a mesh with 8,500 nodes and 16,441 triangles after three successive adaptive mesh refinements. The velocity criteria are used to decide whether to subdivide elements because strong viscous effects are likely to dominate the flow. Figure 7.27 illustrates the initial mesh (upper part on the left), the final mesh (bottom part on the left) and the local refinements near the blade and the wake region.
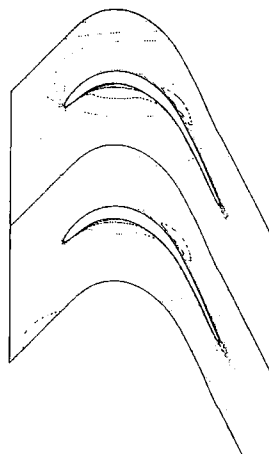
Figure 7.28 Mach number contours
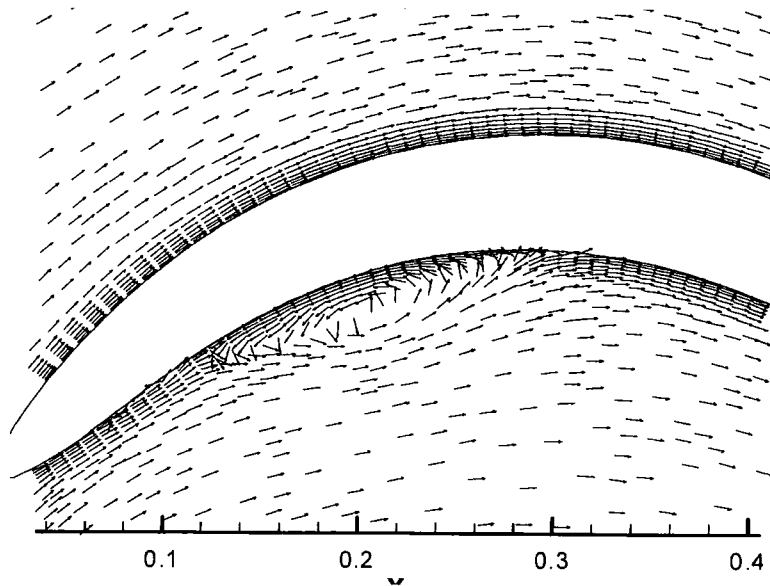
Figure 7.29 Flow vectors near the separation
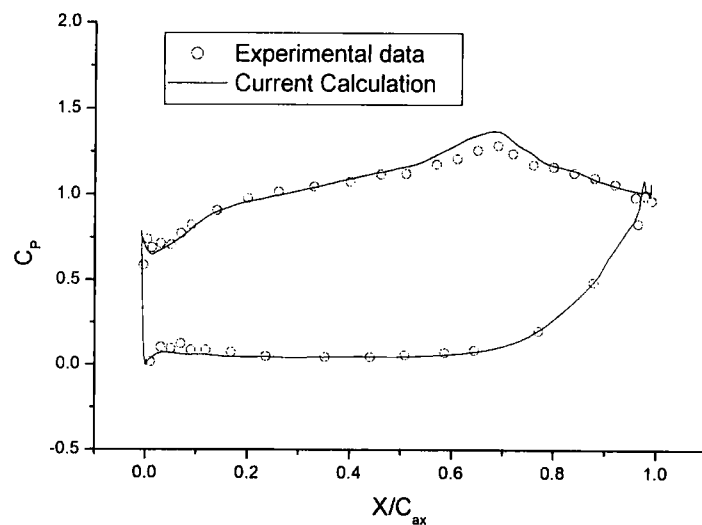
Figure 7.30 Pressure coefficient comparison

Figure 7.28 shows the computed result in terms of Mach number contours in the whole flowfield. Figure 7.29 illustrates the flow vectors in the separation region. The calculated separation bubble starts from about 10% chord and reattaches at about 25-30% of the chord. This agrees with the experiment very well. The comparison of the

pressure coefficient ($\dfrac{p_0 - p}{p_0 - p_2}$), which p_0 is total pressure at the inlet, p_2 is the static

pressure at the outlet) distribution with experiment (He 1998) is plotted in Figure 7.30. Figure 7.31 is the convergence history for the turbulence calculation with three adaptive mesh refinements and three levels of multiple grids (AAMG).
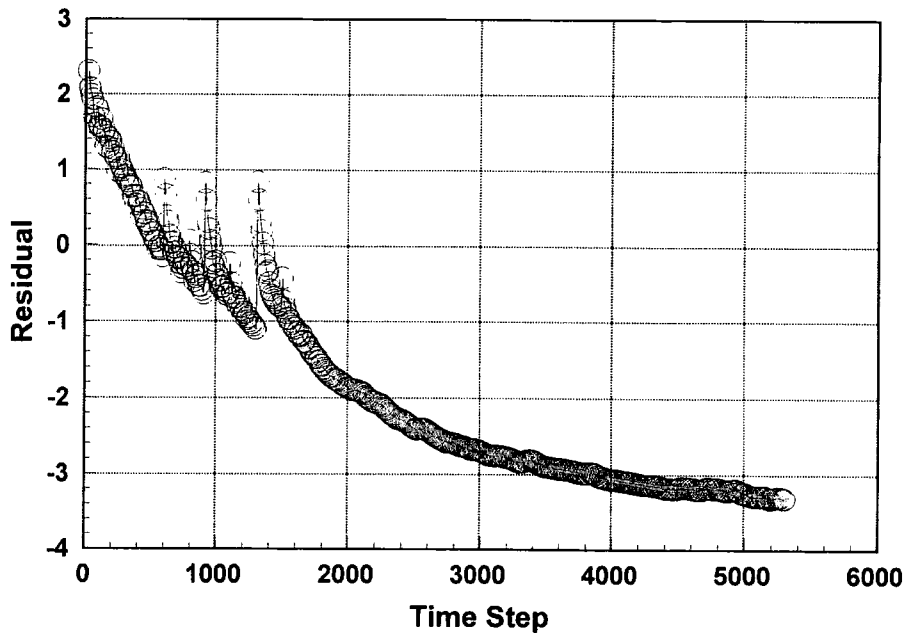
Figure 7.31 Convergence history

An examination of two different multigrid methods, AAMG and DCMG was carried out on the turbine cascade with an incoming flow angle of 40° when high grid aspect ratio is present. The computational mesh used in this case is the final mesh of the previous computation, which consists of 11,309 nodes and 22,038 triangles. Figure 7.32 shows the meshes near the leading and trailing edge, the mesh near the blade is highly stretched to resolve the boundary layer. The maximum aspect ratio of the grid is about 60.

The calculations using two distinct multigrid methods: AAMG and DCMG are performed with the same CFL number, boundary conditions and initial condition. Figure 7.32 shows the convergence histories of the solutions with AAMG and DCMG method. The AAMG solver converges rapidly in 4,100 steps. The DCMG solver converges 2.5 times slower than AAMG solver and also indicates some oscillatory

behaviour. It is evident that in the high aspect ratio case the Aspect-ratio Adaptive Multi-Grid method is more effective.
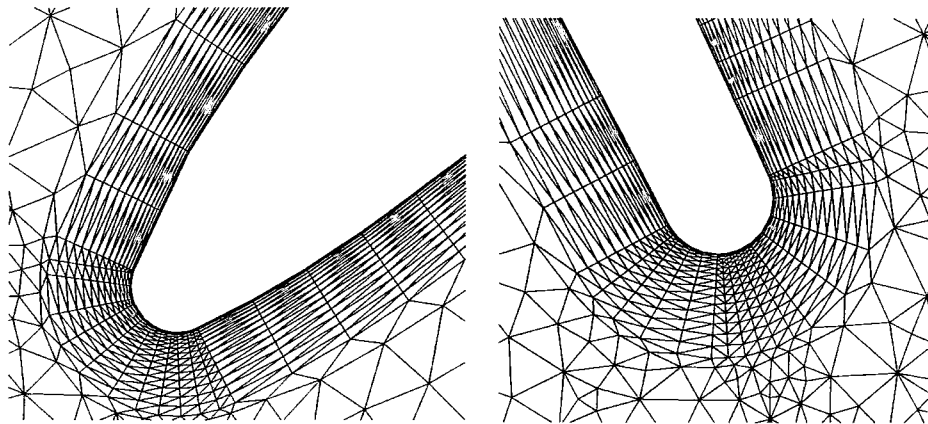
Figure 7.32 Close view of the mesh near the leading and trailing edge
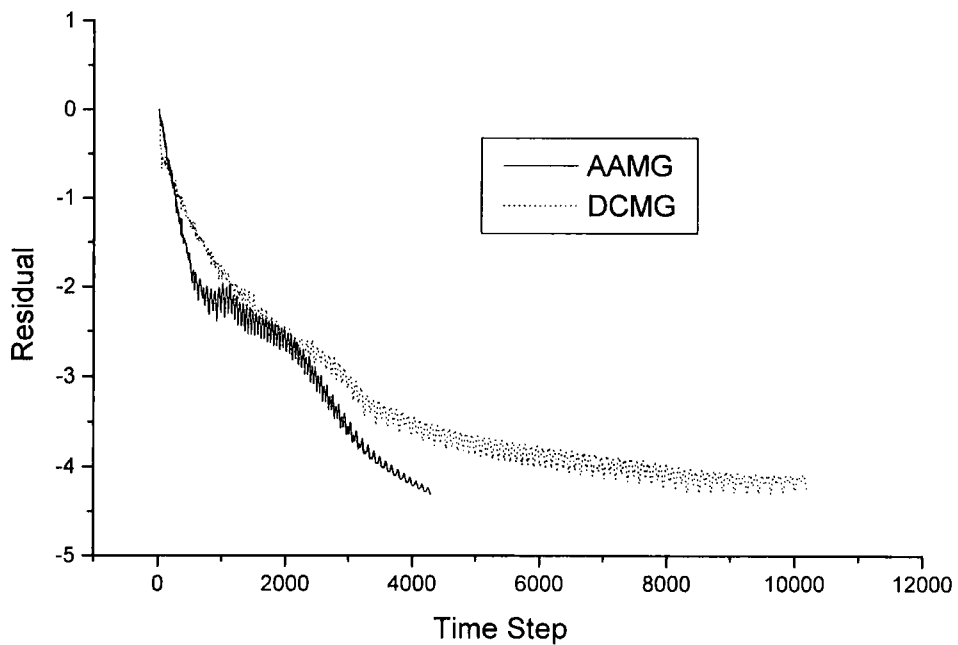

Figure 7.33 Convergence histories

It has been proven that the present 2D flow solver is capable of capturing the small separation bubble and flow reattachment present this case. In the following test case, the incoming flow angle is decreased to $20°$ to examine the effectiveness of the

present multigrid and mesh adaptation techniques in the presence of massive separation of the flow field.

When the incoming flow angle decreased to 20° in the same cascade, a massive separation occurs in the pressure side of the blade, as described by He (1998). The initial computational grid is the same as the 40° flow angle case. After 3 times of mesh adaptations, the flowfield reach a steady state. Figure 7.34 shows the closeup view of the final mesh. It is clear that the mesh is adaptively refined in the separation region (Figure. 7.34a) and the wake region due to high gradients of velocity.
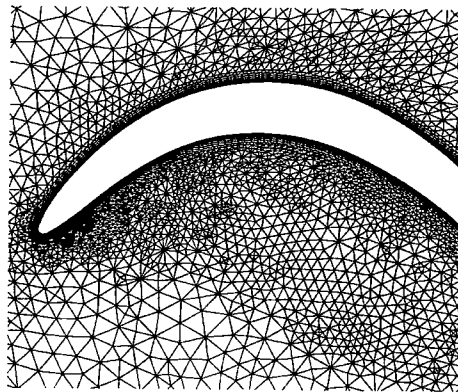
(a) Computational mesh in the seperation region
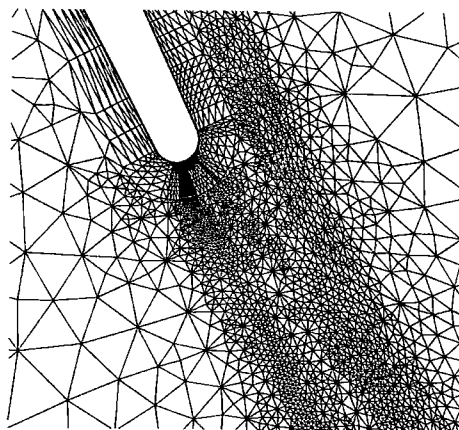
(b) computational grid near the trailing edge

Figure 7.34 Close view the final mesh

144

The result in terms of Mach number is plotted in Figure 7.35. It is clear that the wake is resolved by the mesh adaptation. Flow vectors and eddy viscosity contours are presented in Figure 7.36. The flow separates near the leading edge and reattaches at about 38% of the chord. This agrees well with the experiment.
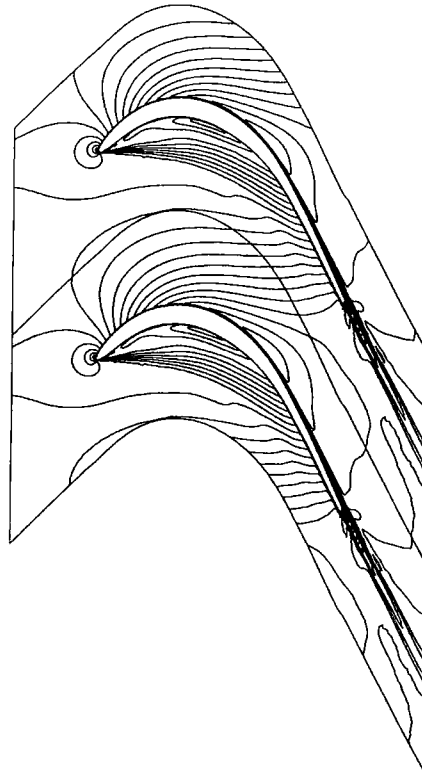
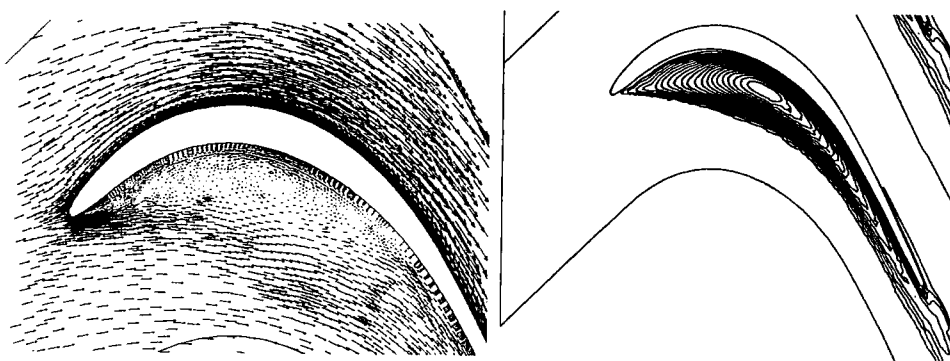
Figure 7.35 Mach number contours

Figure 7.36 Flow vectors and eddy viscosity in the separation region

Figure 7.37 shows the comparison of the pressure distribution with the experimental data; it shows a good overall agreement. Figure 7.38 is the convergence history for the turbulence simulation with three adaptive mesh refinements and three multiple

grids (AAMG). In this case, the convergence is clearly affected by the separation and reattachment of the flow.
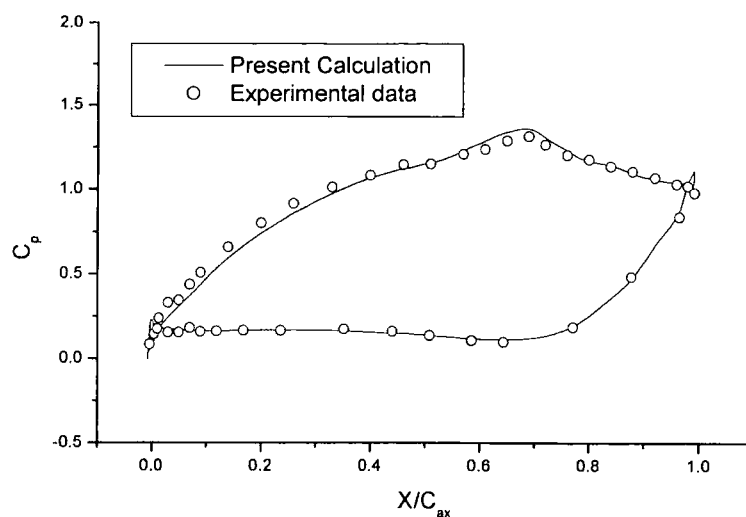
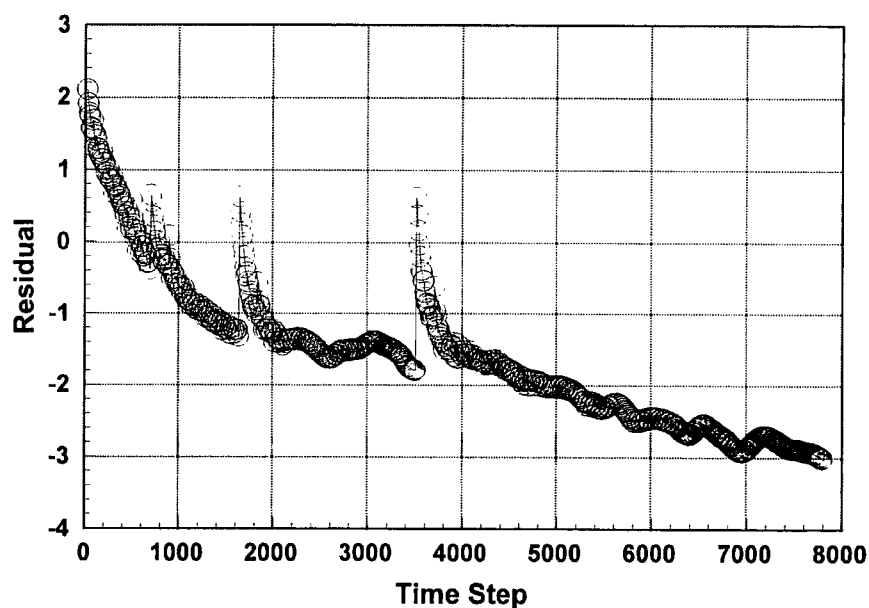**Figure 7.37 Blade pressure distribution**

Figure 7.38 Convergence history

7.4 Concluding Remarks

A 2D flow solver based on a cell-centred finite volume scheme using unstructured-grids has been developed for solving Euler/Navier-stokes equations and has been

146

presented. Several inviscid/viscous flow simulations were performed on the 2D unstructured grid based flow solver. The accuracy and the efficiency of this method were investigated. The following conclusions have been drawn:

- The spatial and temporal discretisation scheme used in the present 2D flow solver has shown to be accurate and efficient in solving both inviscid and viscous steady flow problems. The Roe upwind scheme is accurate in simulating shockwave and boundary layer interaction problems. The turbulence model used in the present solver is accurate in simulating boundary layer problems.

- The inflation method for solving viscous flow problems, such as turbulent flows around the RAE 2822 airfoil and in the turbine cascade, has been shown to be effective and efficient in solving viscous problems. The difficulty in generating the highly stretched grids near the solid wall is reduced by using this method.

- Adaptive mesh refinement exhibits great potential to improve the accuracy of the solution as is shown in simulations of inviscid flow around the NACA 0012 airfoil, turbulent flows around the RAE 2822 airfoil and turbulent flows in the turbine cascade. This method is able to concentrate the computing resource where it is most needed. The over-resolved problems can be countered by a carefully designed error indicator and the two-phase refinement procedure.

- The multigrid method developed in the present research has been shown to be efficient in simulations of both inviscid and viscous flows. The Adaptive Aspect-ratio Multi-Grid method is particularly effective in solving viscous flow problems. In inviscid flow problems, the method becomes a semi-coarsening method. Reasonably good speed-ups compared to a single grid solution are observed.

Chapter 8

3D Validation and Discussion

Several test cases are presented in this chapter to validate the Euler/Navier-Stokes algorithms on 3D unstructured-grids as described in previous chapters. Two distinctive 3D cases are the flow around a transonic airfoil (the ONERA M6 wing) and that around a low speed wind turbine blade (the NREL Phase II Wind turbine).

Accuracy and speed of the 3D flow solvers are studied on various configurations. The accuracy is achieved with the high order spatial discretisation method and an upwind scheme. Multigrid schemes and parallel computing techniques are used to accelerate solutions to steady state and to reduce overall computing time. Furthermore, an alternative discretisation scheme is applied to viscous flow simulation of boundary layer problems to improve the overall accuracy and efficiency of the solutions.

Applications of two distinct three dimensional flow solvers (a prismatic mesh based and a tetrahedron mesh based) developed in the current research are presented in this section. Both are completely standalone CFD codes capable of solving various flow problems. Furthermore, both of them are capable of serving as a slave computing process in the parallel computing mode. As described in previous chapters, the prismatic based solver is aimed at viscous effect dominated regions for efficiently resolving the boundary layer characteristics of the flow, and the tetrahedron based solver is targeted at the outer regions where the viscous effect is relatively small or no boundary layer characteristic is present.

Results of parallel computing on a small Linux clustered PC system are also presented in this chapter. The capability of the cluster is assessed by the speedup on computing a 3D inviscid flow problem with different partition schemes.

8.1 Inviscid Flows around ONERA M6 Wing

The first 3D case considered in this section is the simulation of the transonic flow past an ONERA M6 wing. The configuration has been widely used as a benchmark to validate three dimensional solution algorithms and to evaluate the performance of solution methods. The wing has a symmetrical airfoil section, a leading edge sweep angle of 30 degrees, and an aspect ratio of 3.8. The root chord of the airfoil is 0.67m and the semi-span of the wing is 1.0m with a rounded tip. The geometry of the M6 wing is provided by NPARC Alliance Validation Archive (NASA 1999).

The test case presented here has an incoming flow at $M_\infty = 0.8395$ and a flow angle of $3.06°$, which corresponds to the test 2308 from the report by Schmitt and Charpin in the AGARD Report AR-138 (Schmitt and Charpin 1979). At this condition, two strong shockwaves are developed on the upper wing surface. Correctly resolving the locations of these two shockwaves is the key indicator of solution accuracy. This computation is used to validate the basic 3D algorithm as well as to examine the performance of the multigrid method for inviscid computations.

The computing mesh (Figure 8.1) for the inviscid computation is generated by GMSH developed by Geuzaine and Remacle (1999) in a semi-sphere domain with a radius of 5 times of the main chord of the wing. The 3D unstructured mesh consists of 29,432 nodes, 168,432 tetrahedral elements with 8,642 triangular cells on the wing surface. The overview of the surface mesh on the wing and the symmetric plane is plotted in Figure 8.1a, where the centre of the symmetric plane located at the trailing edge of the root of the wing. The farfield, which is almost a half sphere, is plotted in Figure 8.1b. Figure 8.1c is the close view of the surface meshes of the wing.

The inviscid computation is performed using the tetrahedral based flow solver with an explicit 4-stage Runge-Kutta time stepping method and the multigrid technique.

Three levels of multigrid are employed in the calculation: the original mesh and two coarse levels generated by the DCMG method described in Chapter 5. The computed result is presented in terms of Mach number contours on the wing surface in Figure 8.2. Two shock waves and their crossing are clearly visible on the wing surface. To investigate the shockwave position, comparisons of pressure distributions

($c_p = \dfrac{p_\infty - p}{\frac{1}{2}\rho v_\infty^2}$) at sections located at 20%, 44%, 90% and 95% of the span are plotted

in Figure 8.3. It should be noted that the current computation is performed without viscous effects. The shockwave positions are generally well predicted, better than one would expect from an inviscid solution. This might be due to numerical viscosity and dissipation introduced by the coarse mesh on the wing surface (Figure 8.1c).

To assess the performance of the direct connectivity based multigrid method, a single grid computation is performed on the same grid with the same CFL number starting from free stream conditions. The convergence histories for the single grid and the previous multigrid solution are plotted in Figure 8.4. A 4-5 times improvement in convergence rate is observed. This indicates the effectiveness of the direct connectivity based multigrid. The speedup is slightly lower than that of this method used in two dimensional computations.

8.2 Parallel Computing Performance

As one of main interests in this work is to explore the parallel computing on a cluster system, some corresponding investigations are carried out.

8.2.1 The Cluster System

The parallel computing platform used for the 3D computation is based on a PC clustered system, which consists of 4 Intel Pentium II PCs (which could be directly accessed by the author during the time of the work) running Linux systems. Table 8.1 shows the specifications of these PCs and network hardware.

Table 8.1 Specification of the PC cluster system

	PC01	PC02	PC03	PC04
CPU	Pentium II 450MHZ	Pentium II 500 MHZ	Pentium II 500 MHZ	Pentium III 600 MHZ
Memory	128M	128M	128M	512M
Operation System	Linux (Kernel 2.2, i586)			
Network Hardware	10M Ethernet adaptor, Netgear 10M hub			

8.2.2 Test Case and Computational Mesh

The inviscid flow around M6 Wing case is selected as a test case for parallel computing performance, since the flow feature has been studied in the previous case and the size of computing mesh is suitable for parallel computing on these 4 PCs with relatively small amounts of memory.

The computing mesh used is the same inviscid mesh as shown in Figure 8.1. It consists of 168,432 tetrahedral elements and 29,432 nodes. METIS (Karypis and Kumar 1998) partitioning library is used for domain decomposition. The calculations performed on the clustered system are identical to the one performed on PC01 except that the computational mesh is pre-partitioned to 2, 3 and 4 sub-grids, which are almost equal in size (the difference of the element number is no more than 1). Figure 8.6, 8.7 and 8.8 are the partitioning results for the two-, three- and four-zone runs. The minimum edge method cut is used as the partition criterion.

8.2.3 Timing and Results

Performance tests have been conducted on the cluster PC system described previously. All the tests cases presented here are about solving the inviscid flow over the M6 wing with same boundary and initial conditions. To assess the speedup, all the runs execute 1000 timesteps and the running times are recorded by the internal timer of a PC. Then the overall running time is averaged to each step and listed below.

Table 8.2 Summary of the single processor run

Host	Tetra	Node	Boundary	Computing time (sec)	Time/step (sec)
PC01	168,432	29,432	0	6.50	6.50

Speedup: 1.0   Communication: 0

Table 8.3 Summary of the two-zone run

Host	Tetra	Node	Boundary	Computing time (sec)	Time/step (sec)
PC01	84216	14943	6226	3.25	3.80
PC02	84216	15201	6226	3.25	

Speedup: 1.71   Communication: 0.55 sec/step

Table 8.4 Summary of the three-zone run

Host	Tetra	Node	Boundary	Computing time (sec)	Time/step (sec)
PC01	56138	9909	2094	2.17	2.66
PC02	56138	10253	1138	2.17	
PC03	56138	10484	1172	1.95	

Speedup: 2.44   Communication: 0.49 sec/step

Table 8.5 Summary of the four-zone run

Host	Tetra	Node	Boundary	Computing time (sec)	Time/Step (sec)
PC01	42108	7674	1716	1.62	2.02
PC02	42108	7727	1186	1.62	
PC03	42108	7881	1618	1.46	
PC04	42108	7911	1986	1.23	

Speedup: 3.21   Communication costs: 0.40 sec/step

The final speedup performance has been plotted along with the ideal speedup on this cluster system in Figure 8.9. The ideal speedup is defined by the number of processors used in a parallel computing job. Reasonably good speedup of the PC cluster system has been achieved.

8.3 Turbulent Flows over a Flat Plate

To examine and validate the implementation of the present turbulence model and the viscous term treatment, turbulent flows over a flat plate are simulated with the present prism mesh and tetrahedron mesh based flow solvers.

Both the tetrahedron and prism based mesh for the flat plate boundary layer flow has been generated by subdividing a $61 \times 41 \times 3$ H-topology structured grid. The structured mesh is highly stretched near the solid wall and has uniform spacing in the other two directions. Each hexahedron in the structured-grid is divided into 6 tetrahedral elements in the tetrahedron based mesh and 2 prismatic elements in the prism based mesh. This results in 7,503 points and 14,400 tetrahedral elements for the tetrahedron mesh (Figure 8.10) and 9,600 prismatic elements for the prism mesh. The maximum aspect ratio near the wall is about 30. The computational domain is from x=-0.1 to x=1.0 in the streamwise direction, and z=0.0 to 0.02 in the wall normal direction, and from y=0.0 to y=0.02 in the spanwise direction. The free stream Mach

153

number is maintained at $M_\infty = 0.2$ to minimise the compressible effect and the Reynolds number is $Re_\infty = 3 \times 10^6$ for the tetrahedron flow solver and $Re_\infty = 2 \times 10^6$ for the prism flow solver, respectively.

Both computations are carried out at CFL=1.2 with 2 levels of multigrid. Results using the Spalart-Allmaras one-equation turbulence model with the tetrahedral and prismatic flow solvers are shown in Figure 8.11 and Figure 8.12. In both cases, the velocity profiles agree well with the law of the wall. The residual history for both computations is plotted in Figure 8.13. Both solutions converge rapidly with multigrid acceleration. From the convergence rate point of view, it is evident that the prismamtic solver with the aspect ratio adaptive multigrid delivers better performance than the direct connectivity based method for viscous flow simulations with stretched grid near solid wall surfaces. Furthermore, the computing time of the tetrahedron based solver is far greater than that of the prism based solver, because there are 28,800 elements with the tetrahedral discretisation and 9,600 with the prismatic discretisation. It is clear that the prismatic discretisation is more effective in resolving this boundary layer flow problem.

8.4 Turbulent Flows over ONERA M6 Wing

In this section, numerical results of the turbulent flows over the ONERA M6 wing are presented to validate the present 3D Navier-Stokes algorithms. This case is the same as the case in section 8.1, except that the flow is turbulent. The Reynolds number is 1.172×10^7. When viscous effects are taken into account, the locations of shockwaves on the upper surface of the wing should move forward compared to the inviscid solution due to the presence of boundary layer. The correct prediction of positions and amplitudes of the shock waves is the key indicator of the accuracy of this computation.

The viscous mesh generation follows the method described in Chapter 4. The wing surface profile (Figure 8.14) is inflated by 15% of the main chord. To resolve the wake and avoid the sharp end of the trailing edge of the wing, a 'C' style inflation

scheme is adopted. Due to the fact that the wing is symmetric, the unstructured grid generation is carried out on half of the domain to reduce the memory overhead of the mesh generator. After generating the tetrahedral grid in the outer region, a new interface grid (Figure 8.15) is extracted from the tetrahedral grid. This surface grid is then mapped onto the surface of the wing. The resultant surface grid on the wing is plotted in Figure 8.16. The mapped surface grid and interface grid are used as the baseline grid for the generation of the prismatic grid. Figure 8.17 is the 3D view of the surface mesh on the wing surface and the symmetric plane. The outer region mesh consists of 43,230 nodes, 234,372 tetrahedral elements. There are 16 layers of prismatic elements around the wing and the wake region. Figure 8.18 shows a closeup view of the surface mesh near the wing root regions (The outer regions are plotted with an offset from the inner layers).

The computations are carried out on the cluster system described previously. The computing domain is decomposed to four sub-blocks as shown in Figure 8.18, 2 tetrahedral and 2 prismatic blocks. It is clear that the load on each processor is not balanced because of the presence of 2 different types of blocks. Two tetrahedral based and two prismatic based solution processes are used in the parallel computing.

Figure 8.19 is the pressure contours on the upper wing surface, in which two strong shockwaves can be observed. Comparisons of pressure distributions with experimental data (Schmitt and Charpin 1979; NASA 1999) are presented for locations at 20%, 44%, 65%, 80%, 90% and 95% of the span in Figure 8.20. At 20% span, the position of the first shockwave is well resolved, but the solution has failed to predict the position of the second one. It should be noted that in the viscous computation, a finer unstructured grid is employed, thus leads to a sharper shock wave than the previous inviscid solution. At 44% of the span, reasonably good agreement is displayed by the prediction of both shock waves and the overall good match of the pressure coefficient on both surfaces of the wing. At 65% and 80% of the span, the second shockwave is well resolved, but the first shockwave, which is present near the leading edge, is smeared. This may be caused by insufficient grid resolution in this region. At 90% and 95% of span, good overall agreement is

observed. The strong shockwave is well predicted. Generally speaking, the first shockwave is smeared due to the limitation of the current mesh generation package. However, some minor disagreement at the bottom wing surface is shown at 90% of span. Very good agreement of pressure distribution is observed for the bottom wing surface except at 90% of the span and the second shockwave is well modelled. This indicates high accuracy of both prism and tetrahedron based flow solvers.

In order to evaluate the ability of the multigrid method to solve turbulent flows, a multigrid simulation is carried out with the same CFL number and initial conditions. Three mesh levels are employed in the multigrid calculation: the original mesh and two coarse meshes generated by our AAMG method. In the regions with semi-structured mesh (consisting of prismatic elements), coarser levels are built by stacking certain numbers of layers in the finer mesh level depending on the maximum aspect ratio of that layer. In this case, each of the two coarser levels consists of 2 and 4 layers of the original fine grid. In the full unstructured-mesh regions (consisting of tetrahedron elements), a semi-coarsening procedure is used to generate a coarser level.

The efficiency of the AAMG multigrid method is assessed by the convergence comparison with a single grid computation. Figure 8.21 is the comparison of convergence history. It is evident that the present multigrid scheme achieves about 5 times speedup compared with the single grid solution. This speedup is less than the same method in 2D unstructured-grid cases (around 7 for viscous flows). The main reason behind this could be the actual 3D flow effects and the assumption that the flow is strongly 1D in the near solid wall regions. In the present 3D AAMG method, the method of stacking finer layers results in fast information propagation in the direction normal to the wall, but is less effective when the flow along the wall changes rapidly, as in this case when shockwaves are present on the wing surface.

8.5 Inviscid/Turbulent Flows over a Wind Turbine Blade

This test case concerns a wind turbine tested at the National Wind Technology Centre of the National Renewable Energy Laboratory (NREL), Colorado, USA. It is a 10.06 m diameter, three-bladed, downwind, free-yaw turbine (Figure 8.22).

This case is chosen because the complex nature of the flow around the blade provides us a good chance to demonstrate the state-of-the-art unstructured flow solver for rotary aerodynamics.

8.5.1 Geometry of the wind turbine blade

The wind turbine blade is non-twisted and non-tapered. The blades consist of an S809 airfoil, developed by Airfoils Inc. for NREL (Duque et al. 2000; Duque et al. 1999). The size of blade span is 4.52 m and the root is at radius 0.51 m. The blade pitch angle is 12 degrees. For the present test cases, where there is no twist, the pitch angle corresponds to the local blade angle, i.e. the angle between the chordline of the blade element and the rotor plane, and is positive when it points opposite to the wind direction. Figure 8.23 shows the definition of the pitch angle of the non-twisted blade.

8.5.2 Test cases and flow conditions

There are three test cases available in the public domain (FLOWNET 2001), for which experimental data is available. They correspond to different incoming flow speeds, as shown in Table 8.6. In all cases the incoming flow is assumed to be fully axial.

Table 8.6 Definitions of the test cases

	Test Case 1	Test Case 2	Test Case 3
Incoming flow speed	7 m/s	13 m/s	19 m/s
Rotational speed	71.68 rpm	71.19 rpm	71.54 rpm
Static temperature	283.1 K	292.1 K	291.1 K
Static pressure	81055 Pa	80138 Pa	80138 Pa
Re (based on diameter)	3.99×10^5	7.237×10^5	1.06×10^6

For all these test cases, the pressure distributions are available at spanwise positions of 30%, 47%, 63% and 80%.

Table 8.7 Angle of attack at different spanwise positions

Spanwise Position	Test Case 1 7 m/s	Test Case 2 13 m/s	Test Case 3 19 m/s
30%	19.87°	37.3°	47.4°
47%	9.64°	24.58°	35.18°
63%	4.49°	16.97°	26.84°
80%	1.2°	11.55°	20.38°

The simulations have been carried out at slightly different flow conditions to the original test cases to avoid the calculation of the virtually incompressible flow with a compressible flow solver. The incoming flow speed has to be increased to Mach number 0.1 to maintain good convergence of the solver.

According to the velocity triangle in Figure 8.24, to keep the same flow angle with original test cases, the rotation speed has to be increased by the scale of the increase of the absolute velocity. Table 8.7 shows calculated flow angles at four spanwise locations where experimental data are available. The Reynolds numbers are kept the same with those of the test cases to maintain solution similarities with the actual flows. The definitions of the flow conditions for the numerical simulations are given in Table 8.8.

158

Table 8.8 Flow conditions of simulations

	Test Case 1	Test Case 2	Test case 3
Incoming flow speed	33.721 m/s	34.24 m/s	34.20 m/s
Rotational speed	345.307 rpm	187.574 rpm	128.772 rpm
Static temperature	283.1 K	292.1 K	291.1 K
Static pressure	81055 Pa	80138 Pa	80138 Pa
Re (based on diameter)	3.99×10^5	7.237×10^5	1.06×10^6

8.5.3 Computational Mesh Generation

There are only three non-twisted blades in the wind turbine, as plotted in Figure 8.22. Thus, the blade-blade interaction is unlikely to be very strong except near the shaft regions and computational costs can be reduced by using a small part of the flow domain. In this case, a computational domain less than 1/5 of the whole domain is adopted. The inlet and outlet plane are placed at 2 chords away from the blade and the top surface is placed at 5 chords away from the blade tip. The flows on these "periodic" and far field boundaries are assumed to be undisturbed.

The computational mesh for the inviscid simulation is generated by GMSH (Geuzaine and Remacle 1999) using an advancing layer method. The mesh consists of 82,176 tetrahedral elements and 40,212 nodes. There are 32,760 triangle elements and 16,470 nodes on the surface of the turbine blade. Figure 8.25 shows the computational mesh on the wind turbine and the surface grid of the computational domain.

The mesh for the viscous simulation is generated following the way described in Chapter 4. First, the surface triangulation is achieved by dividing a structured surface mesh. Then, a similar surface mesh is generated at 1/10*chord away from the surface with the same connectivity of the surface mesh by moving the surface mesh normal to the surface. Next, this new blade surface triangulation along with the surface triangulation of the other boundaries of the domain is used for the volume mesh generation with GMSH. The resultant volume mesh of the outer domain consists of 236,558 tetrahedral elements and 150,636 nodes. Around the turbine blade, there are

24 prismatic element layers, which are built from a surface mesh of 129,280 triangles and 82,480 nodes. The maximum mesh aspect ratio near the blade is about 60.

8.5.4 Numerical Results and Discussions

Numerical simulations of flows around the wind turbine blade are performed on the PC cluster system described previously. The computing mesh is partitioned equally to 4 domains with minimum communication restraint in each case using the METIS graph partition library.

Inviscid simulations are carried out for all three flow conditions listed in the Table 8.8. Farefield boundary conditions are applied to the inlet/outlet and the top surface. A slip wall condition is applied to the hub surface. Since the computational mesh contains only tetrahedral elements, four tetrahedral slave computing processes are employed in all three inviscid computations.

In Figure 8.26 predicted pressure coefficient from the test case 1 (*7m/s*) is plotted along with the experimental data from Flownet (Flownet 2001). Excellent agreement with experimental data at all four spanwise locations is observed. At 7m/s, the flow angle is relatively small (Table 8.2) from the hub to tip. Therefore, the flow is largely attached. The agreement demonstrated that the tetrahedron flow solver is accurate.

Figure 8.27 shows the comparison of predicted pressure coefficient for the test case 2 (13*m/s*) and the experimental data. Good agreement is displayed for 63% and 80% spanwise positions. At 30% span position, the predicted pressure distribution is inaccurate on the suction surface, because the flow is separated in these regions when the wind speed is increased to 13m/s (34.24 m/s in the simulation) and the present inviscid solution is incapable of predicting separation.

When the incoming flow speed increased to 19 m/s (test case 3), massive separation appears near the hub regions where the flow angle is very high (Table 8.7). The present inviscid solution failed to predict the pressure correctly in these regions, as shown in Figure 8.28. Good agreement is observed at spanwise positions of 44%, 63% and 80%, where the flow is largely attached.

In Figure 8.29, the computational grid and flow vector & streamlines on the pressure surface at 22%-28% span at 7m/s of the blade are plotted. It should be noted that the large separation on the surface is non-physical phenomena because the inviscid solution is incapable of predicting separation. However, the real flow in this region should show similar pattern, such as strong radial flow and large separation.

Further efforts were made to use the Navier-Stokes solver to compute the turbulent flows for this wind turbine case. However, it appears that the present 3D viscous flow solver has some convergence problems for low Mach flow conditions. Because for low Mach number flows, the stiffness of the governing equations is very high and the numerical methods developed for compressible flows could break down or not function properly (Wesseling 1999). The reason behind this break down is that the solution of the Navier-Stokes equations contains pressure fluctuations of the order of Mach number while the continuous pressure scales with the square of Mach number (Guillard and Viozat 1999). The use of preconditioning (Godfrey and Leer 1993; Wesseling 1999) should help in addressing this problem.

8.6 Concluding and Remarks

- A cell-centred finite volume scheme has been presented for the solution of the Euler/Navier-Stokes equations on 3D unstructured meshes. An upwind scheme has been adopted in both prismatic mesh and tetrahedron mesh based flow solvers to compute the inviscid flux contributions. The flow solvers employ an efficient and accurate wall function procedure and use a face stencil to construct interface gradients. A parallel computing technique is adopted to reduce computing times in 3D. The flow solvers exploited a multi-block scheme.

- The accuracy of the present 3D spatial discretisation is displayed by a number of test cases. The simulations of inviscid flows around the M6 wing and a wind turbine blade show good agreement with experimental results. Excellent agreements with experimental data have been observed in the turbulent flow simulations over a flat plate and the M6 wing. This indicates that the present 3D flow solvers are accurate in simulating turbulent flows. However, the present 3D

flow solver is not functioned properly for simulations of low Mach number flows as shown in the wind turbine flow case. A preconditioning method is needed to remove the convergence difficulty associated with high stiffness.

- The proposed "inflation" unstructured mesh generation method is tested for 3D cases. In the viscous flow around the M6 wing, a "C" type inflation scheme is used to generate highly stretched elements in viscous effects dominated regions. In the outer domain, tetrahedral elements can be easily generated using an isotropic unstructured mesh generator. This method shows good potential in reducing the difficulty of generating stretched viscous mesh near solid walls and improving overall solution accuracy. Furthermore, this mesh generation scheme enables the exploitation of an efficient and robust multigrid method.

- The proposed multigrid method is effective both for inviscid and turbulent flow simulations as shown in the cases of inviscid and viscous flow around the M6 wing. This indicates that computational methods employed in the current research are computationally efficient. However, the efficiency of the aspect-ratio adaptive multigrid in 3D viscous flow computations is less satisfactory than in 2D. This is mainly due to the 2D nature of the method. An aspect-ratio adaptive multigrid capable of building coarser levels both normal to the wall and along the wall direction is urgently needed.

- The parallel computing on the PC cluster system is successful. In all tests conducted on the M6 wing, very good speedup is observed. This indicates that the present implementation of parallel computing with PVM message passing is efficient on a low bandwidth and high latency cluster system. This cluster system is also used in simulating turbulent flows around the M6 wing and flows over the wind turbine blade. The conclusion is that clustering currently available desktop PCs or workstations to build a middle level parallel system is possible. However, due to lack of resource (only 4 PCs are available) the full strength of this clustering idea has not been investigated.
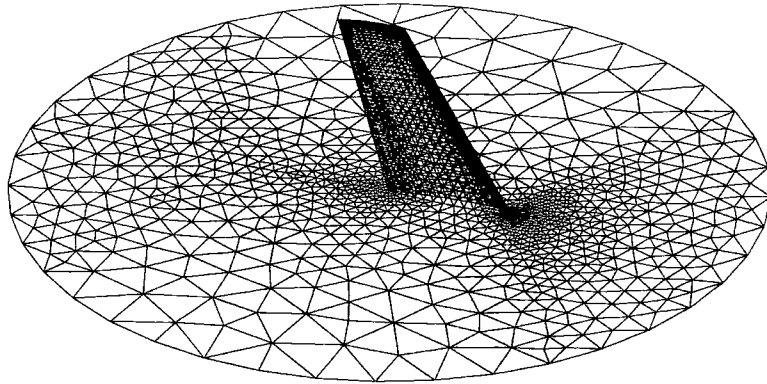
(a) Overview of the surface mesh on wing and the symmetric plane
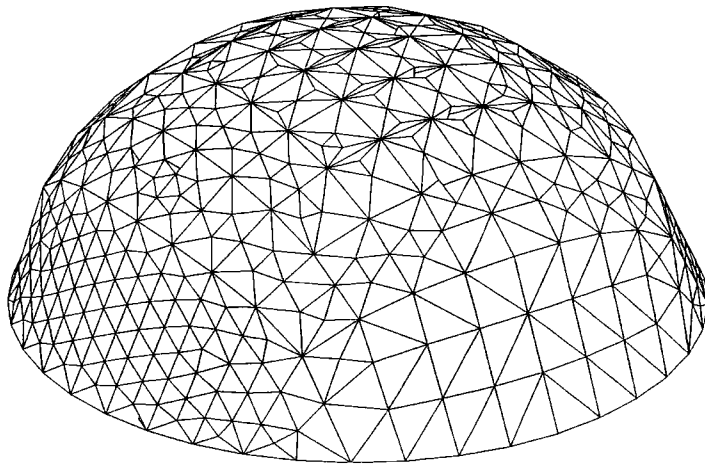
(b) Surface mesh of the farfield
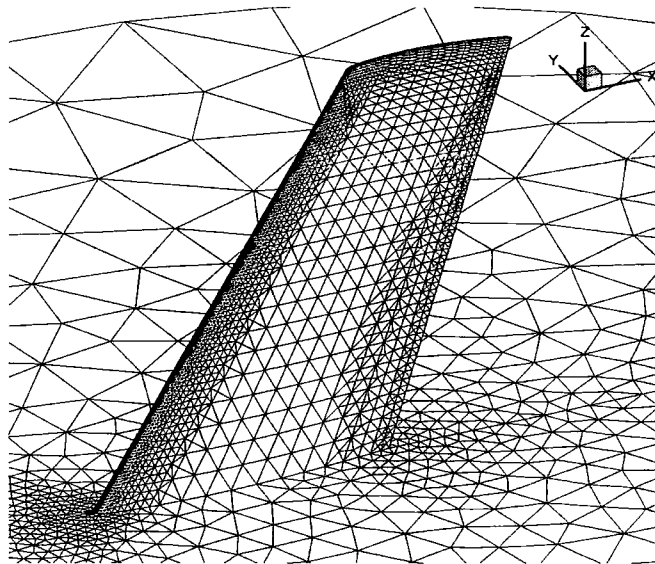
(c) Surface grid on the wing surface

Figure 8.1 Computational grid
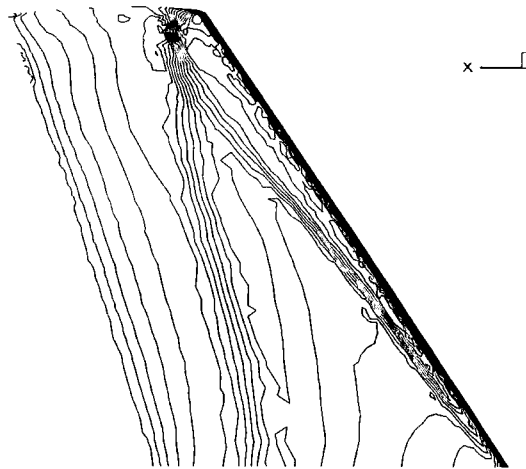
Figure 8.2 Mach number contours
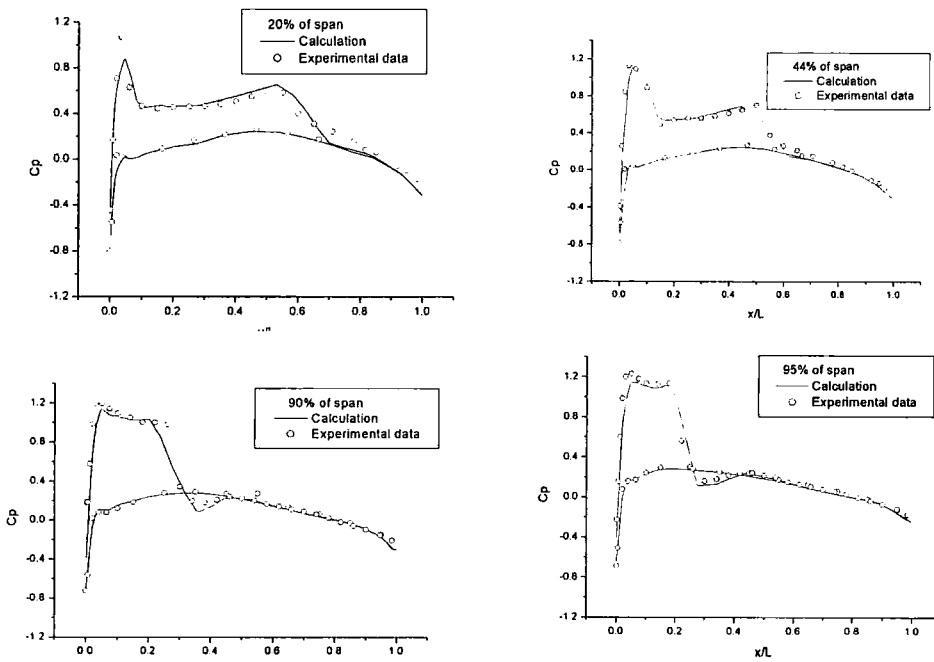
Figure 8.3 Surface pressure distributions
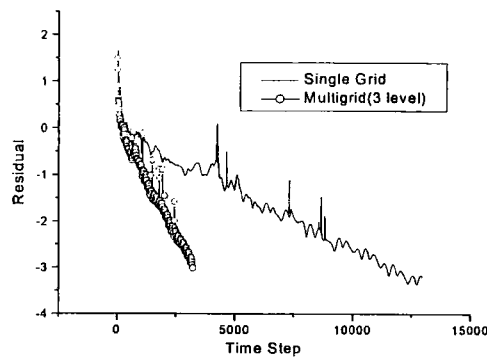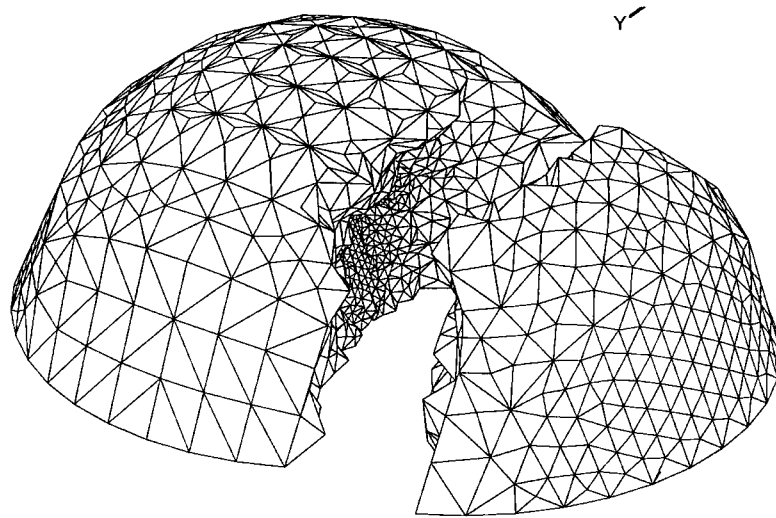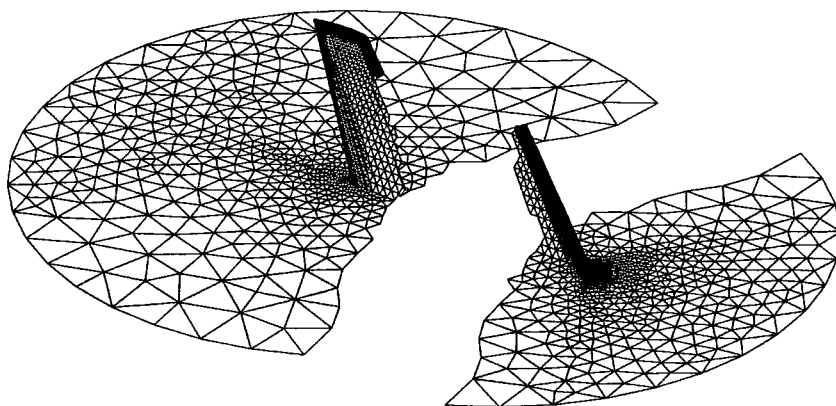

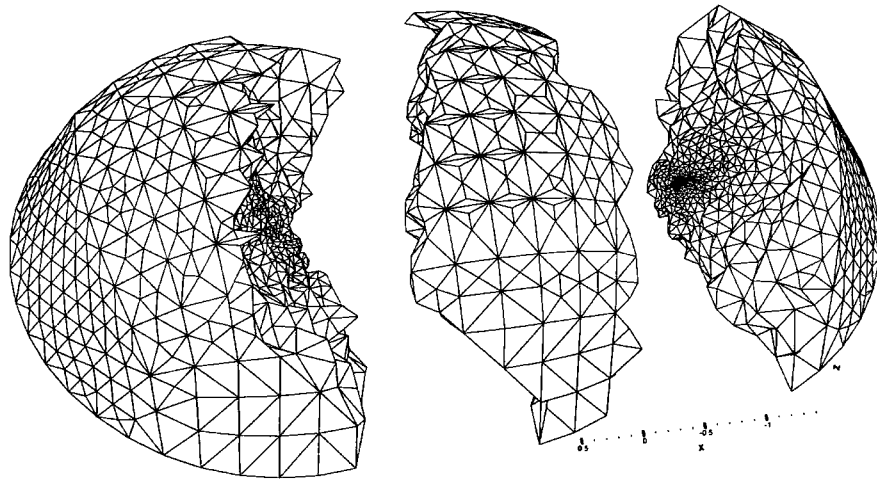**Figure 8.4 Convergence histories**
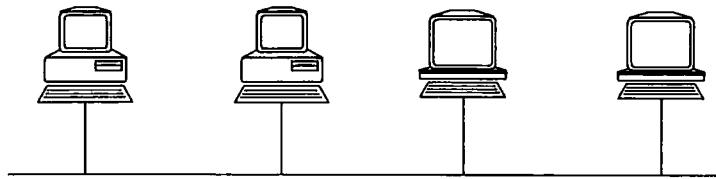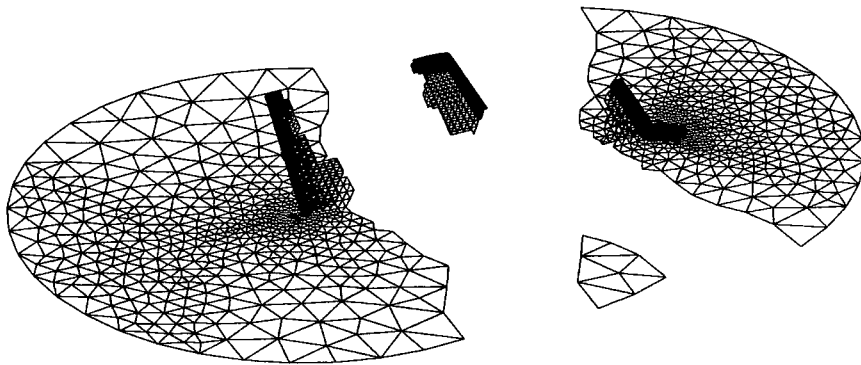
Figure 8.5 A Linux PC cluster system

(a) Full flow field view

(b) The wing surface and the symmetric plane
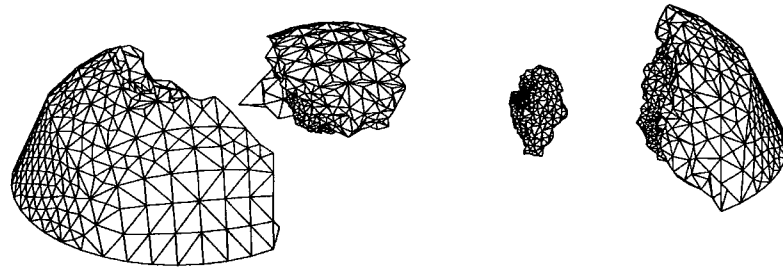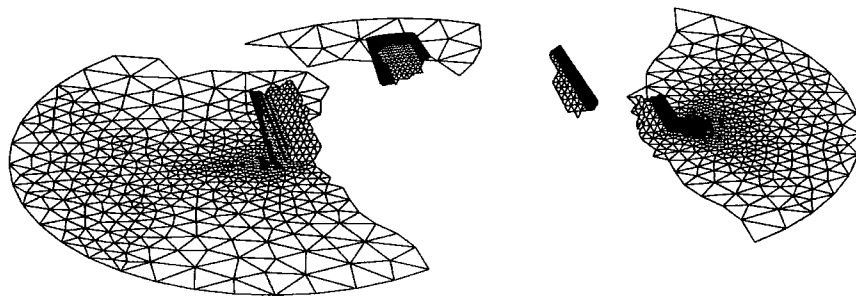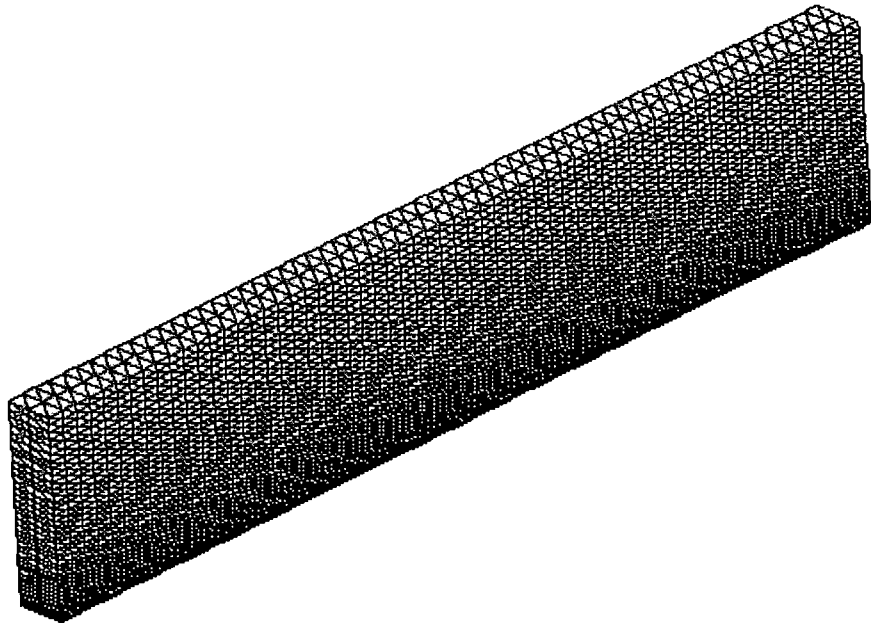**Figure 8.6 Two zones of the computational grid**

165

(a) Full flow field view

(b) The wing surface and the symmetric plane
Figure 8.7 Three zones of the computational grid

(a) Full flow field view

(b) The wing surface and the symmetric plane

Figure 8.8 Four zones of the computational grid
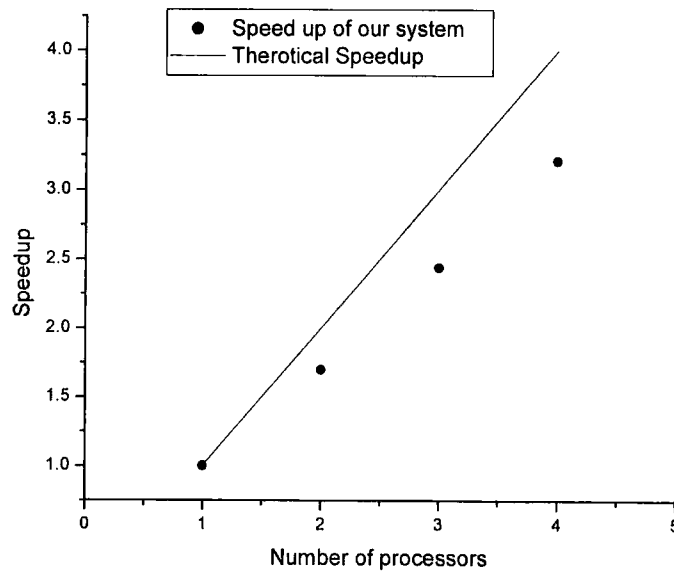
Figure 8.9 Observed speedup

Figure 8.10   Tetrahedral grid for the flat plate
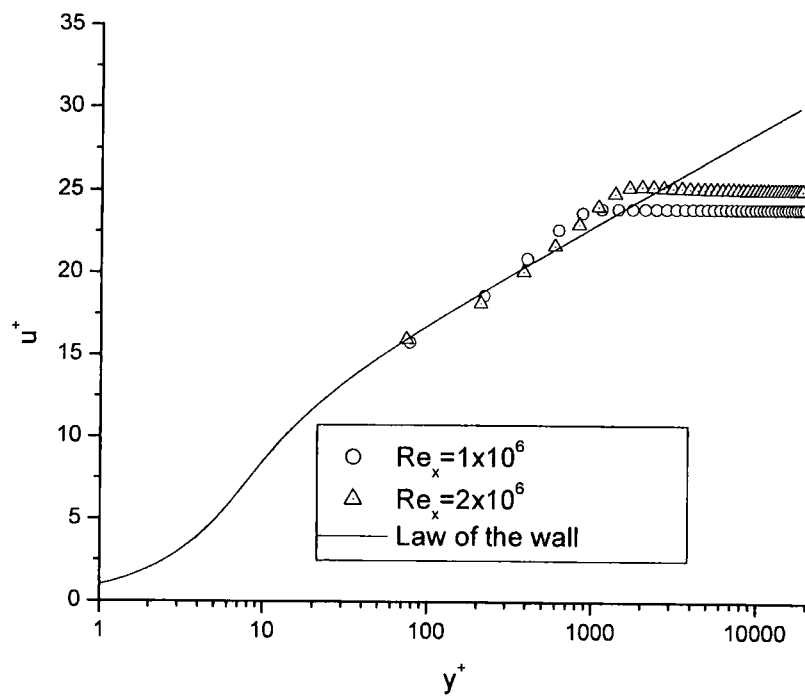
Figure 8.11 Velocity profile on the tetrahedron mesh
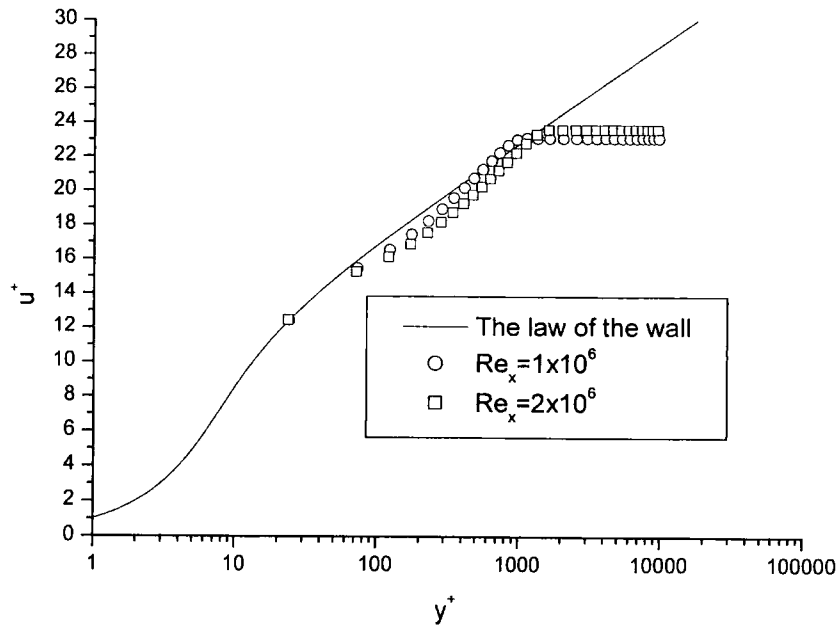
Figure 8.12 Velocity profile on the prismatic mesh
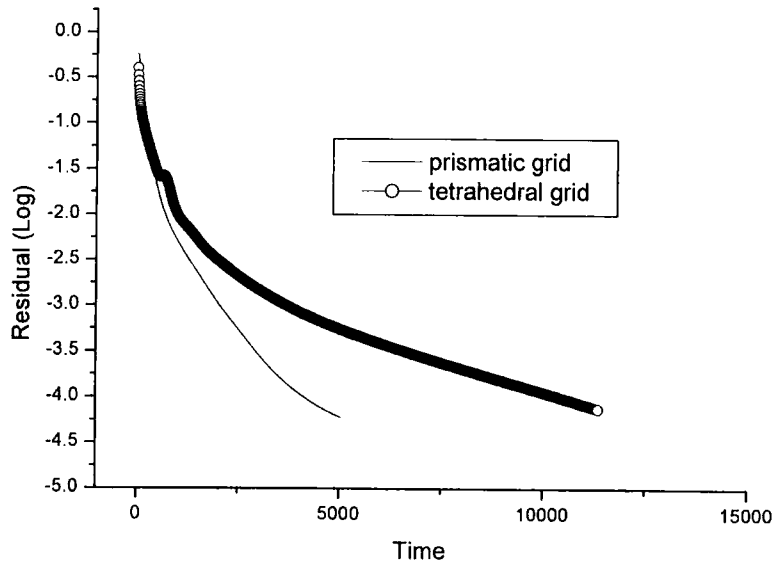
Figure 8.13 Convergence history
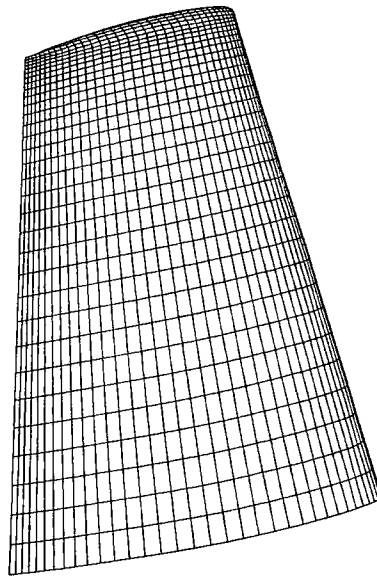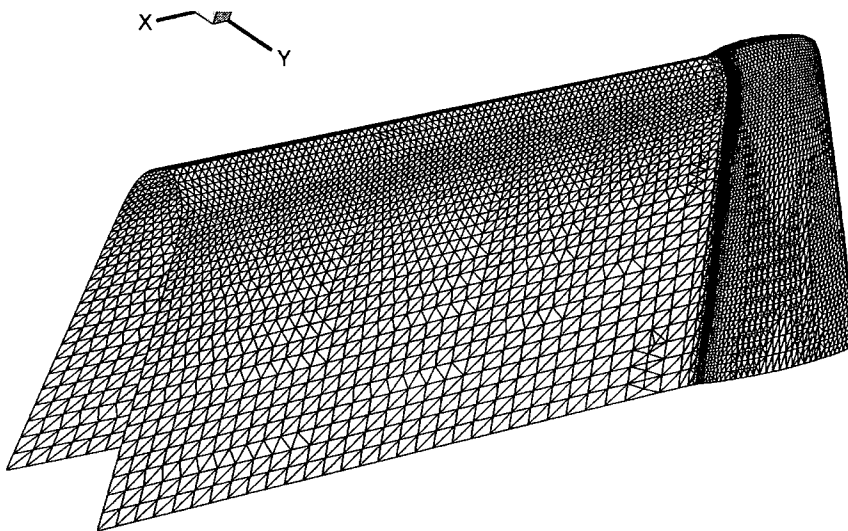
Figure 8.14 Surface profile of the M6 wing
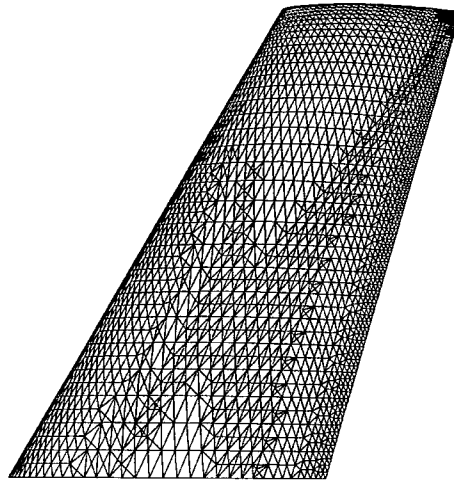

Figure 8.15 New interface grid

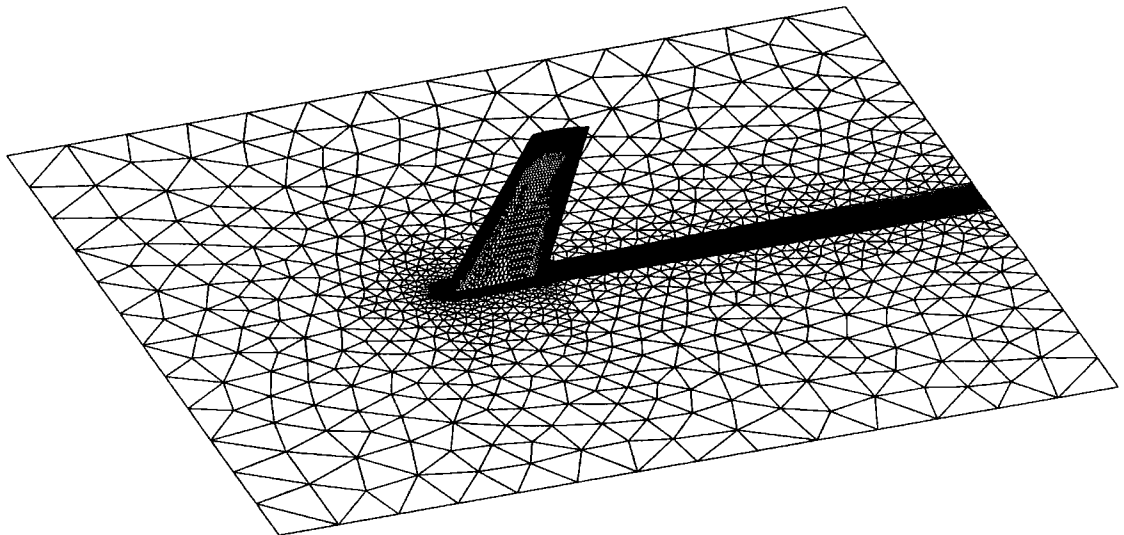**Figure 8.16 Mesh on the wing surface**

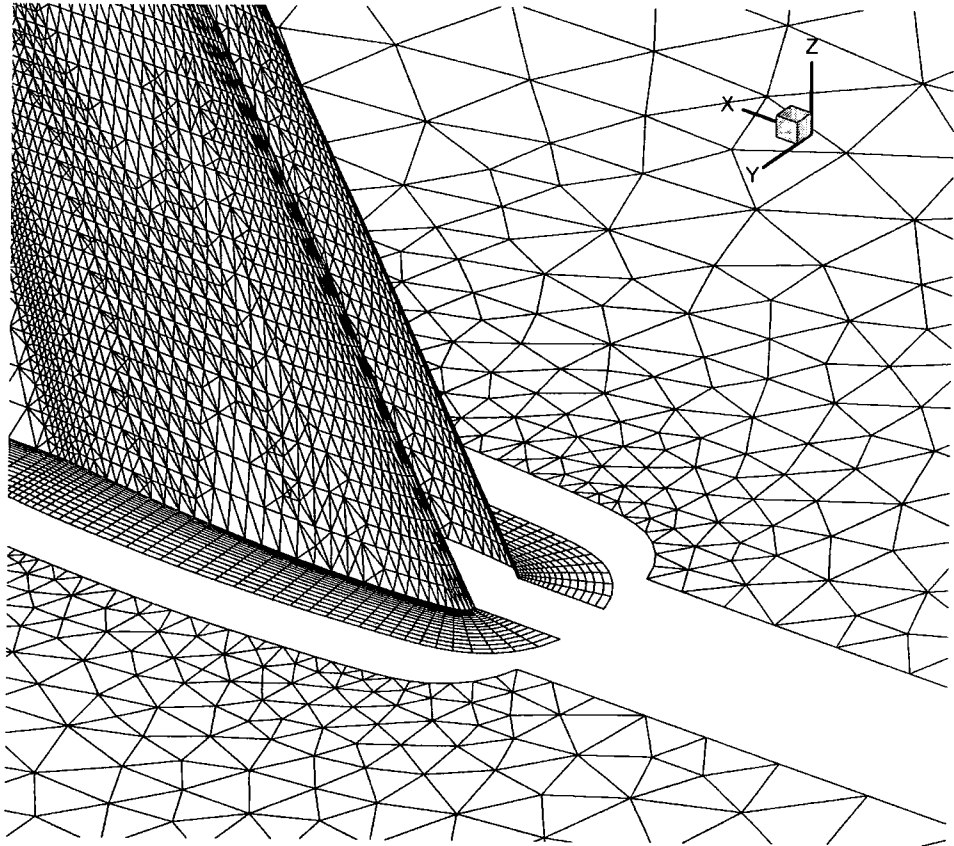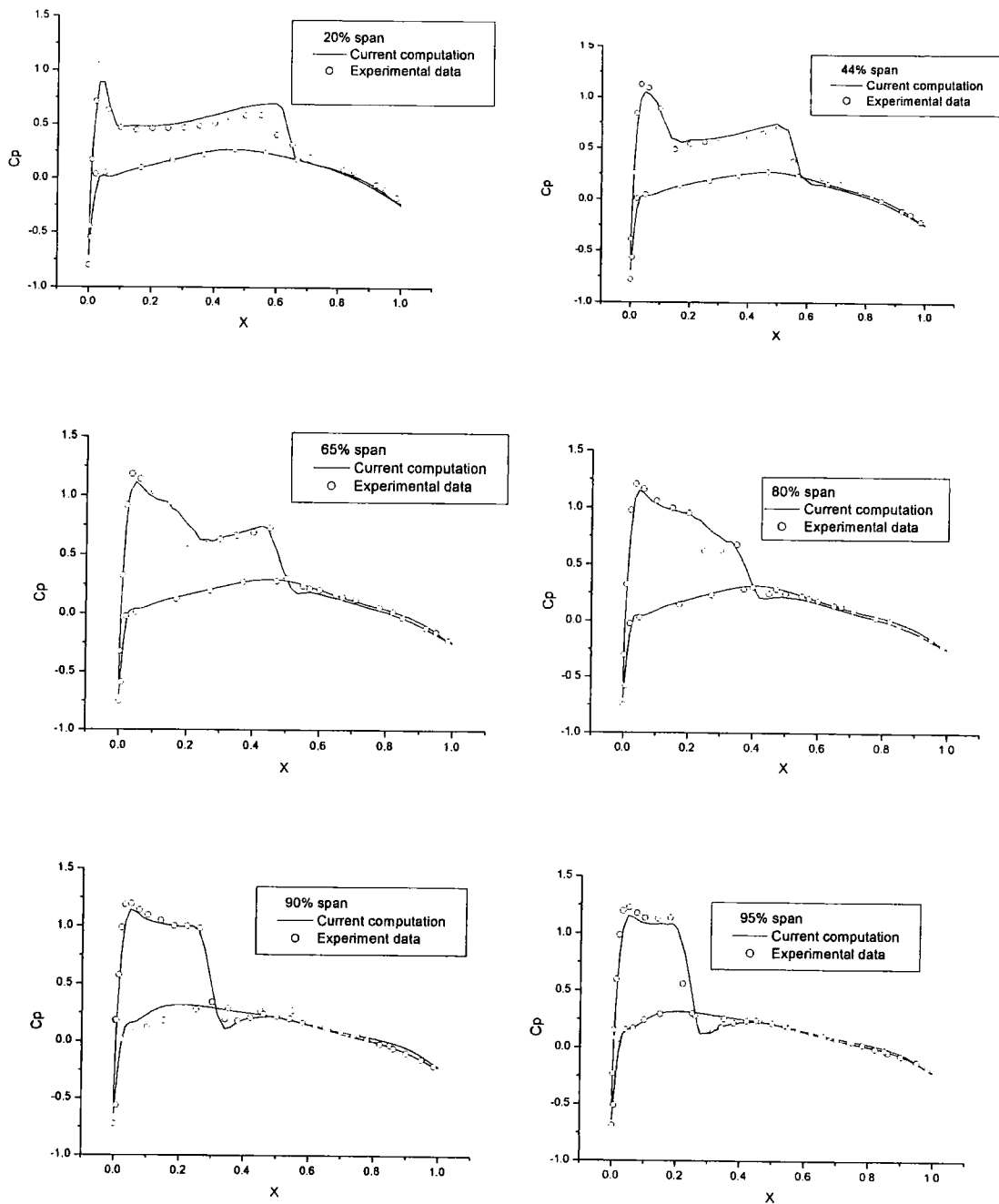**Figure 8.17 3D view of the surface mesh**

Figure 8.18 Four zones of the computational grid

Figure 8.19   Mach number contours on the wing surface

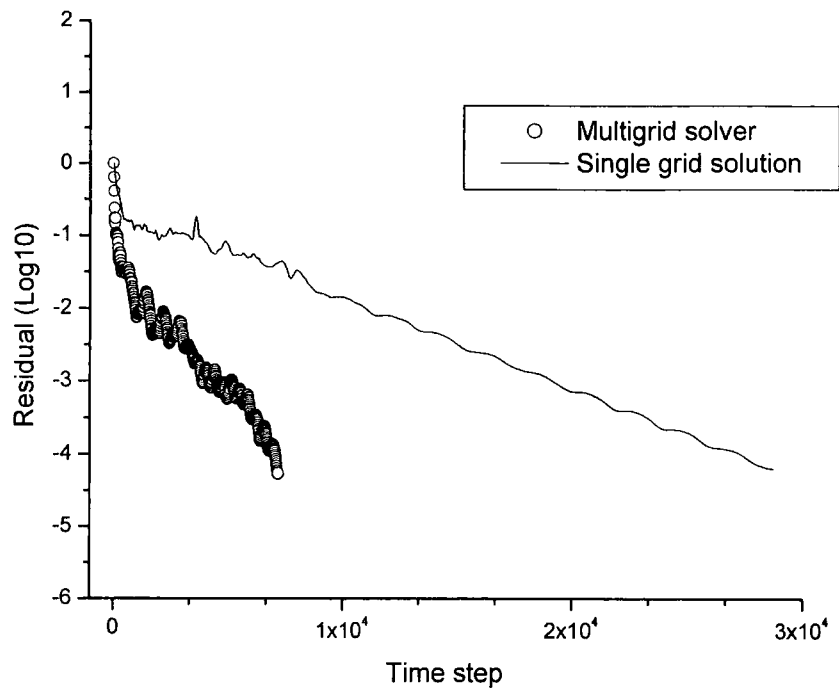**Figure 8.20 Pressure distributions**

Figure 8.21 Convergence comparison of single grid and multigrid solutions

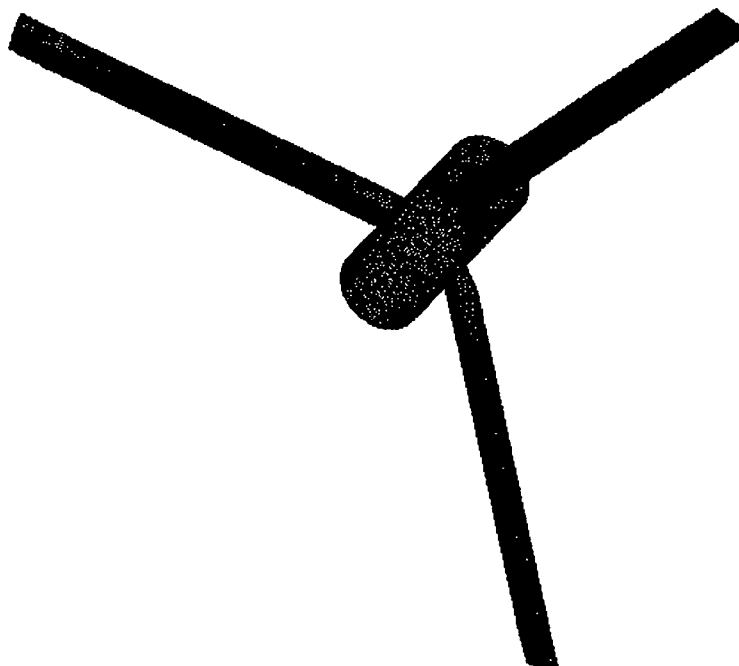**Figure 8.22 NREL wind turbine**

174
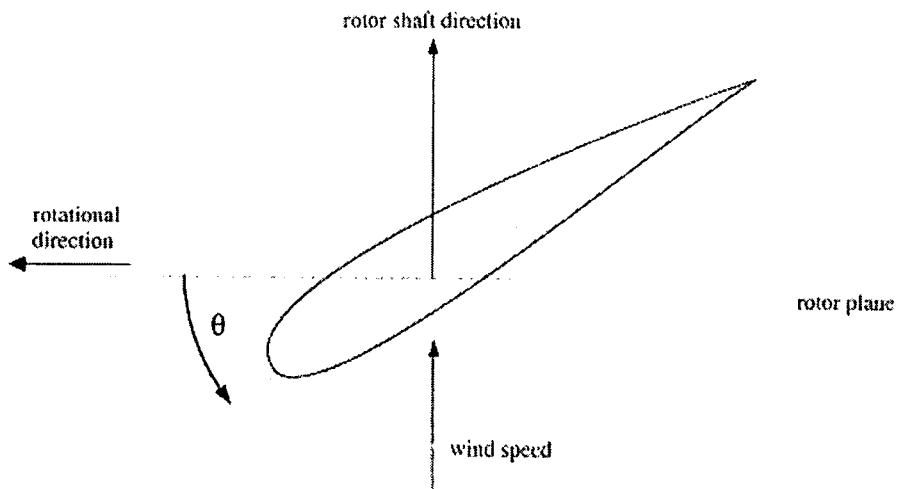
Figure 8.23 Pitch angle of the non-twisted blade
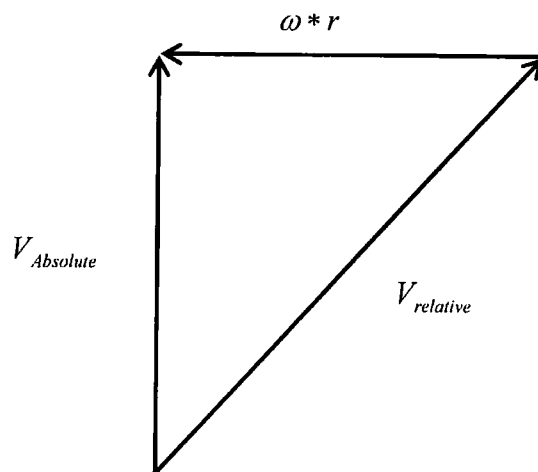
Figure 8.24 Velocity triangle
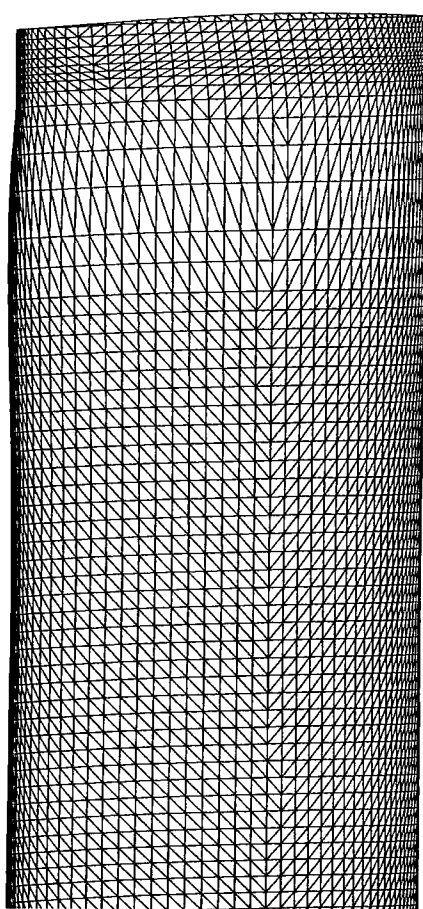
(a) near the tip region   (b) near the hub region
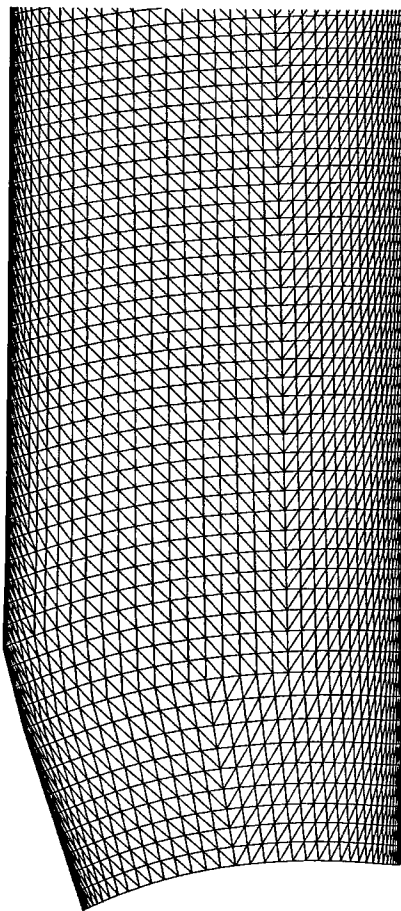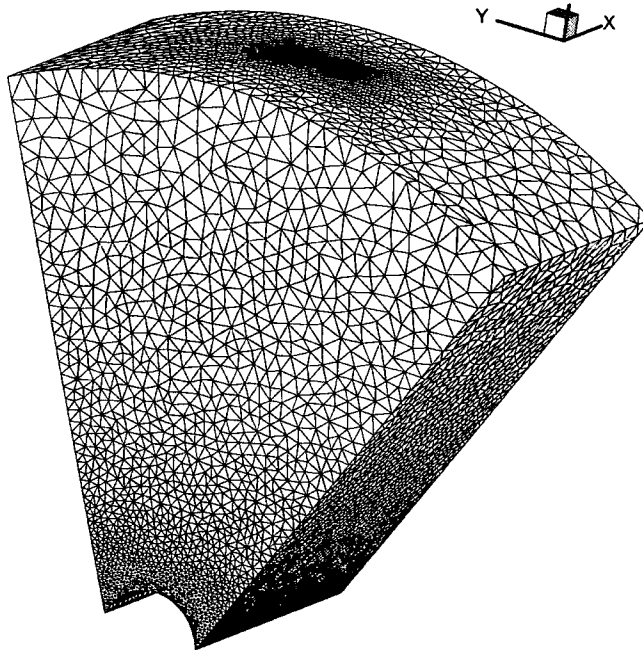
(b) farfield

Figure 8.25 Computational grid for single passage
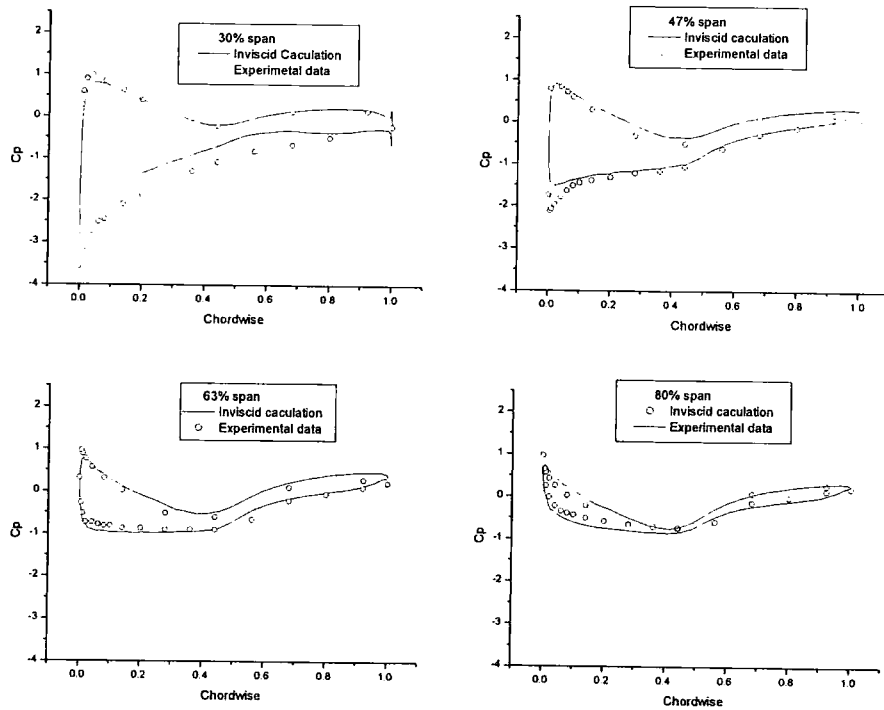
Figure 8.26 Pressure distributions on the wind turbine (7 m/s)
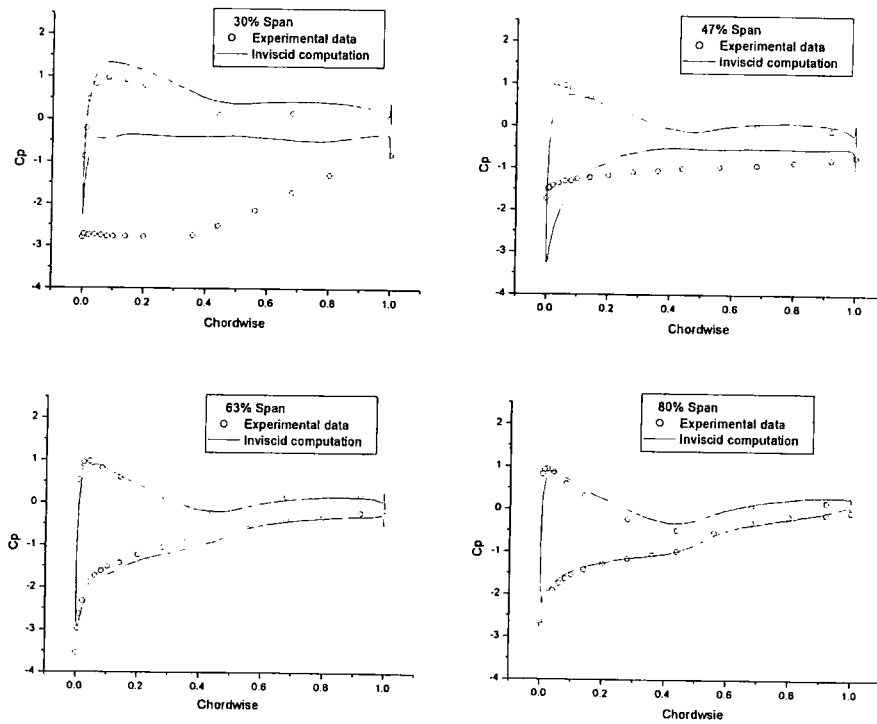
Figure 8.27 Pressure distributions on the wind turbine (13 *m/s*)
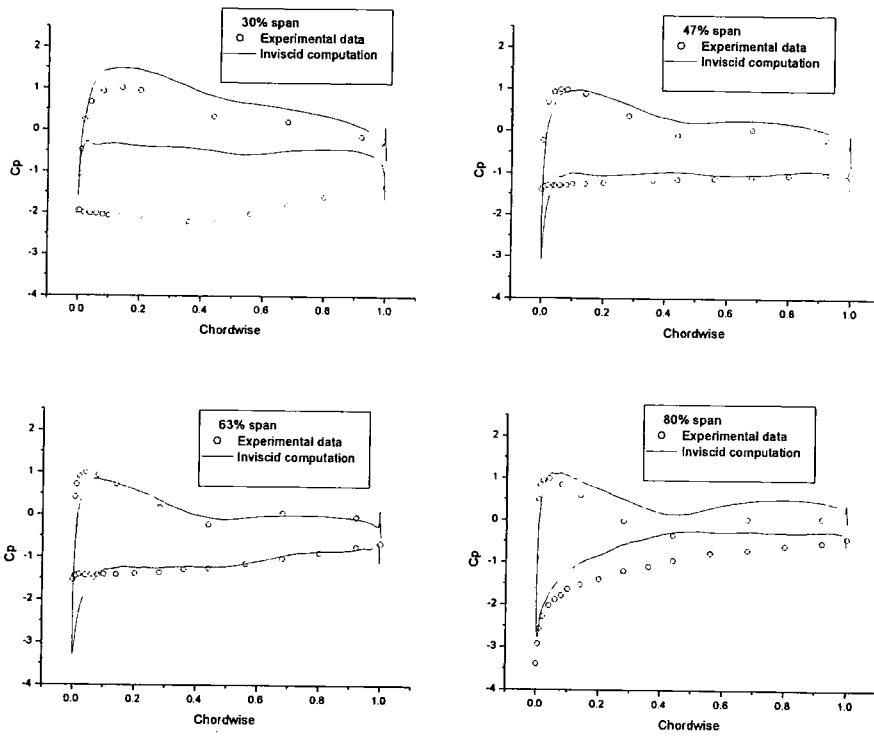
Figure 8.28 Pressure distributions on the wind turbine (19 *m/s*)

(a) Computational grid   (b) Flow vector & stream line
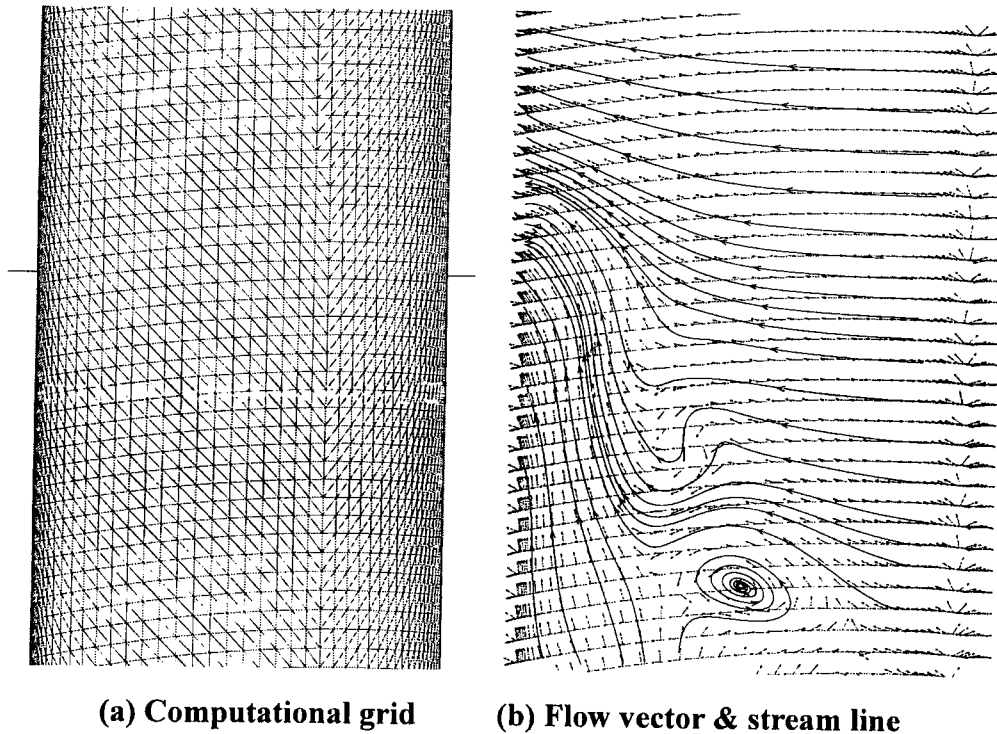
Figure 8.29 3D plot of the flow at 22%-28% of the span (7 *m/s*)

178

Chapter 9

Conclusions and Recommendations

In this chapter, conclusions and findings are drawn from the development of efficient and accurate solution algorithms for Euler/Navier-Stokes equations on 2D/3D unstructured meshes. This thesis has presented four contributions all aimed at improve the accuracy and efficiency of the unstructured-grid method. These contributions are the 2D/3D spatial discretisation and inflation mesh generation scheme developed in Chapter 3 and 4, the solution mesh adaptation scheme discussed in Chapter 4, the aspect-ratio related Multigrid approach described in Chapter 5 and the parallel computing technique on a cluster system discussed in Chapter 6. This chapter ends with recommendations and suggestions for further research.

9.1 Conclusions and Highlights

The primary aim of the present work is to develop efficient and accurate solution algorithms for the Euler/Navier-Stokes equations on 2D/3D unstructured meshes. In respect to the overall objective, 2D/3D flow solvers based on unstructured-grids are developed for aerodynamics applications. Extensive studies have been carried out to validate the algorithms and assess the accuracy and efficiency of the solution methods. The following conclusions are drawn from the present research:

1. The 2D and 3D inviscid/viscous flow solvers based on unstructured-grids for steady compressible flows are capable of solving the Euler/Navier-Stokes equations for 2D and 3D aerodynamics applications. The spatial discretisation with a cell-centred finite volume scheme on 2D and 3D unstructured meshes is accurate in simulating inviscid and viscous turbulence flows. The multistage time integration scheme along with a local time stepping technique is effective in marching flows to a steady state solution.

2. The "inflation" mesh generation technique coupled with traditional isotropic mesh generators to generate 2D/3D computational meshes is effective for all the present viscous computations. This method also improves standard unstructured-grid schemes in terms of accuracy, speed and storage. Accuracy and efficiency are improved by using prismatic elements in the regions where highly stretched cells are necessary to resolve the disparity in directional gradients. Furthermore, this inflation method enables an efficient and robust multigrid method and a solution mesh adaptation procedure to overcome the over-resolved problems.

3. The mesh adaptation technique developed for the present 2D flow solver can effectively improve the accuracy of solution with reasonable costs. The method is based on an adaptive mesh refinement procedure. Various strategies are used to capture shockwave / boundary layer problems. By utilising the structure of the viscous grids, the meshes can be better refined near solid wall

regions. Thus the over-resolving problem is overcome with a two-phase refinement procedure in the viscous flow simulations.

4. The multigrid method developed in the present research is effective in accelerating the solution of the Euler/Navier-Stokes equations. The Aspect-ratio Adaptive multigrid is particularly effective when a high aspect ratio grid is used in viscous turbulence flow simulations.

5. The cluster system developed in the present research can be used for middle level high performance computing. The mesh partitioning and communication scheme developed is suitable for parallel computing on a PC cluster system.

9.2 Suggestion for Further Research

In the light of above conclusions it is felt that this exercise has established good confidence in the solving of aerodynamic problems with unstructured meshes. The unstructured-grid method has displayed an excellent ability to simulate complex flow problems in various geometric configurations. It has been foreseen that this method would play a more important role in solving flow problems over complex geometries with the aid of solution mesh adaptation and parallel computing techniques in the future. However, there are still major challenges with this kind of method and a number of possible improvements are suggested below.

9.3.1 Mesh Generation

The present "inflation" method has been very successful in 2D viscous flow simulations. With this method, highly stretched elements can be easily generated without the sacrificing flexibility of the unstructured-grid. However, dealing with corner points and sharp-ended objects remains a challenge in 3D. A more sophisticated and robust method is needed to generate more effective artificial boundaries that separate inner viscous layers and outer regions before this method can be used in more complex geometries.

9.3.2 Time Marching Solution

The explicit time marching scheme adopted in the present work represents a straightforward way of integrating the Euler/Navier-Stokes equations. It is ideal for damping high-frequency errors. However, it has been proved to be inefficient for low-frequency error damping. This leads to slow convergence in simulations of unsteady flows and viscous turbulent flows. An efficient implicit method might be beneficial for solving steady and unsteady viscous flow problems on unstructured-grids. However, this is not easy and there are several difficult issues in developing an implicit method on unstructured-grids: multigrid, parallel scalability, high Reynolds number flows.

9.3.3 Solution Mesh Adaptation

Solution mesh adaptation holds the key for the success of the unstructured-grid method. The adaptive mesh refinement method developed in this project has been proved to be capable of capturing complex phenomena such as shock waves, boundary layers, separation and wakes. However, the present remeshing scheme is based on "refining" a grid when the local error is high. A "coarsening" procedure is necessary when high grid density is no longer required in some previously refined regions in solving unsteady flow problems. Furthermore, given the state of the current viscous mesh generation, the idea of solution mesh adaptation for simulating 3D viscous flows becomes more important in improving the quality of grids and thus the accuracy of solutions.

9.3.4 Multigrid Techniques

The new multigrid method developed in the presented work has displayed excellent convergence rate for both 2D inviscid and viscous flow simulations. The Direct Connectivity based Multigrid shows moderate convergence rate in simulating 3D inviscid flows. However, the performance of the new multigrid method for 3D viscous flow simulations is less than satisfactory. A more general aspect-ratio sensitive multigrid, i.e. not only stacking layers of prismatic elements in the wall

normal direction but also grouping elements in the other two directions when necessary, should greatly improve the efficiency of this method.

9.3.5 Navier-Stokes Solvers

A preconditioning method is need for the present flow solvers in order to solve low Mach number viscous flows. The convergence problem revealed in the 3D viscous flow simulation around the NERL Phase II wind turbine blade highlights the urgency.

Turbulence modelling represents an important issue for solving complex aerodynamic problems. In this work, the one-equation model of Spalart and Allmaras model has been used. When dealing with more complex problems, extension to more sophisticated models might be required. However, more complex turbulence models may also increase the numerical stiffness of the solution.

9.3.6 Parallel Computing Suggestions

New trends in software and hardware technology are likely to make computing using clusters more promising. Clustered super-computers are seen everywhere. Further testing of the cluster system with more PCs is required to assess the strength of this kind of systems. The graph-partitioning scheme with METIS is well suitable for small and middle size applications. However, it is noticed that the increasing requirement of partitioning memory with METIS becomes higher as the number of points in the mesh increases. Therefore, a new efficient partition scheme is needed to partition a large 3D unstructured mesh domain. Furthermore, the present partitioning scheme is not optimised when both prismatic and tetrahedral blocks are present for viscous computations. A more sophisticated partitioning scheme is needed to produce more effective partitions. In the present parallelisation of 3D flow solvers, the error tolerance is not implemented. This means when one node in the cluster fails, the job has to be restarted manually from the last saved point. An error tolerant implementation is needed to automatically start the job on other nodes when one fails, in order to improve the overall reliability and operational effectiveness of the parallel computing.

References

Abdol-Hamid, K. S., B. Lakshmanan, et al. (1995). 'Application of Navier-Stokes code PAB3D k-e turbulence model to attached and separated flows'. NASA Tecnical paper 3480

Addison-Wesley Inc. (1998). **"Designing and Building Parallel Programs"**. *http://www.aw.com/, http://www.anl.gov/.*

Aftosmis, M., D. Gaitonde, et al. (1994). 'On the Accuracy, Atability and Monotonicity of Various Reconstruction for Unstrcutured Meshes'. AIAA paper 94-0415

Allmaras, S. R. 'A Coupled Euler/Navier-Stokes Algorithm for 2-D Unsteady Transonic Shock/Boundary-Layer Interaction' (MASSACHUSETTS INSTITUTE OF TECHNOLOGY, Ph. D Thesis, 1989)

Amaladas, J. R. and H. Kamath (1998). Accuracy assessment of upwind algorithms for steady-state computations. Computers & Fluids 27(8), pp. 941-962.

Anderson, W. K. and D. L. Bonhaus (1994). An implicit upwind algorithm for computing turbulence flows on unstructured grids. Computers & Fluids 21(1), pp. 1-21.

Anderson, W. K., J. L. Thomas, et al. (1986). Comparison of finite volume flux vector splittings for the Euler equations. AIAA Journal 24(9), pp. 1453-1460.

Baggag, A., H. Atkins, et al. (1999). 'Parallelization of an Object-Oriented Unstructured Aeroacoustics Solver'. NASA Report NASA/CR-1999-209098, ICASE Rep. No. 99-11

Baker, T. J. (1989). Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrianed Delaunay Triangulation. Engineering with Computers 5, pp. 161-175.

Balwin, B. S. and T. J. Barth (1991). 'A one-equation turbulence transport model for High Reynolds number wall bound flows'. NASA Technical Memorandum 102847

Balwin, B. S. and H. J. Lomax (1991). 'Thin layer approximation and algebraic model for searated turbulence flows'. AIAA paper 78-257

Barakos, G., M. Vahdati, et al. (2001). A fully distributed unstructured Navier-Stokes solver for large scale aeroelasticity computations. The Aeronautical Journal August(8), pp. 419-426.

Bardina, J. E., P. G. Huang, et al. (1997). 'Tubulence modeling validation, test, and development'. NASA Technical Memorandum NASA Technical Memorandum 110446

Barth, T. J. (1991). 'Numerical aspects of computing viscous high Reynolds flows on unstructured meshes'. AIAA paper AIAA-91-0721

Barth, T. J. (1995). 'Aspect of Unstructured Grids and Finite Volume Solver for the Euler and Navier-Stokes Equations'. von Karman Institute for Fluid Dynamics Lecture Series 1995-02

Barth, T. J. (1995). 'An unstructured mesh Newton solver for compressible fluid flow and it's parallel implementation'. AIAA paper AIAA-95-0221

Barth, T. J. and D. C. Jesperson (1989). 'The design and application of upwind schemes on unstructured mesh'. AIAA paper AIAA 89-0366

Bottasso, C. L., H. L. D. Cougny, et al. (1994). 'Compressible Aerodynamics Using a Parallel Adaptive Time-Discontinuous Galerkin Least-Square Finite Element Method'. AIAA paper 94-1888

Bowyer, A. (1981). Computing DirichletTessellations. Computer Journal 24(2), pp. 162-166.

Carre, G. (1997). An implicit multigrid method by agglomeration applied to turbulence flows. Computers & Fluids 26(3), pp. 299-320.

Connel, S. D. and M. E. Braaten (1994). 'Semi-structured mesh generation for 3-d Navier-Stokes calculations'. GE Research and Development Center Tech Rep. 94CRD154

Connell, S. D. and D. G. Holmes (1994). A 3d unstructured adaptive multigrid scheme for the Euler equations. AIAA Journal 32(2).

Cook, P., M. McDonald, et al. (1979). 'Airfoil RAE 2822 - pressure distributions and boundary layer wake measurement'. AGARD AR-138

Crumpton, P. I. and M. B. G. Gile (1997). 'Aircraft computations using multigrid and an unstructured parallel library'. AIAA paper 95-0210

Crumpton, P. I. and M. B. Giles (1993). 'OPlus programmer's manual'. Oxford University Computing Laboratory

Danish Wind Industry Association (2001). **"Wind energy and wind turbines: Danish wind industry Association"**. _http://www.windpower.org/core.htm._

Dawes, W. N. (1992). 'The Extension of a Solution Adaptive Three-Dimensional Navier-Stokes Solver Toward Arbitrary Complexity'. ASME paper 92-GT-363

Dawes, W. N. (1993). The Extension of a Solution Adaptive Three-Dimensional Navier-Stokes Solver Toward Arbitrary Complexity. <u>ASME Journal of Turbomachinery</u> 115(4).

Dawes, W. N. (1994). The Solution Adaptive Numerical Simulation of the Three-Dimensional Viscous Flows in the Serpentine Coolant Passage of a Radial in flow Turbine Blade. <u>ASME Journal of Turbomachinery</u> 116(1).

Denton, J. D. (1983). An Improved Time-Marching Method for Turbomachinery Flow Calculations. <u>ASME Journal of Engineering for Gas Turbines and Power</u> 105, pp. 514-524.

Department of Trade and Industry, United Kindom (2002). **"dti: Energy - Renewable"**. *http://www2.dti.gov.uk/energy/renewables/policy/overview.shtml*.

Desideri, J. A. and A. Dervieux (1988). 'Compressible flow solvers using unstructured grids'. von Karman Institute for Fluid Dynamics VKI lecture Series 1988-05

Diskin, B. (1999). 'Solving Upwind-biased Discretizations II: Multigrid Solver Using Semicoarsening'. NASA Langley Research Center NASA/CR-1999-209355

Duque, E. P. N., C. P. V. Dame, et al. (1999). <u>Navier-Stokes simulations of the NREL combined experimental Phase II rotor</u>. In European Wind Energy Conference, Nice, France.

Duque, E. P. N., W. Johnson, et al. (2000). 'Numerical Predictions of Wind Turbine Power and Aerodynamic Loads for NREL Phase II Combined Experiment Rotor'. AIAA paper 2000-0038

Fischer M. (1998). **"WINRSHD - A Remote Execution Facility to Harness and Control Remote Windows"**. *http://www.winrshd.com/*.

Fischer M. (1999). **"PVM as Message Passing Environment PVM 3.4.4 for Win32"**. *http://www.markus-fischer.de/getpvmwin32.htm*.

FLOWNET (2001). **"Flownet test case database: NREL wind turbine"**. *http://dataserv.inria.fr/flownet/public/index.php3*.

Freitag, L. A. and C. F. Ollivier-Gooch (1997). Tetrahedral Mesh Improvement Using Swapping and Smoothing. <u>International Journal for Numerical Methods in Engineering</u> 40(1), pp. 3979-4002.

Frink, N. T. (1992). Upwind scheme for solving Euler equations on unstructured tetrahedral meshes. <u>AIAA Journal</u> 30(1), pp. 70-71.

Frink, N. T. (1994). 'Recent progress toward a three-dimensional unstructured Navier-Stokes flow solver'. AIAA paper 94-0061

Frink, N. T. (1996). 'Assessment of an unstructured-grid method for predicting 3-D trubulent flows'. AIAA paper 96-0292

Frink, N. T., P. Arikh, et al. (1991). 'A fast upwind solver for the Euler equations on three-dimensional unstructred meshes'. AIAA paper 91-0102

Frink, N. T. and S. Z. Pirzadeh (1998). 'Tetrahedral Finite-Volume Solution to the Navier-Stokes Equations on Complex Configurations'. NASA NASA/TM-1998-208961

Geist, G. A., J. A. Kohl, et al. (1996). PVM and MPI: A Comparison of Features. Calculateurs Paralleles 8(2).

Geuzaine, C. and Remacle, J. (1999). **"Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities"**. *http://www.geuz.org/gmsh/*.

Giles, M. and R. Haimes (1993). Validation of a Numerical Method for Unsteady Flow Calculations. ASME Journal of Turbomachinery 115.

Godfrey, A. G. and B. V. Leer (1993). 'Preconditioning for the Navier-Stokes equations with finite chemistry'. AIAA paper 93-0535

Gopakaswamy, N., H. U. Akay, et al. (1997). Parallization and dynamic load balancing of NPARC codes. AIAA Journal 35(12).

Gropp, W., Lusk E. (1999). **"MPICH-A Portable Implementation of MPI"**. *http://www-unix.mcs.anl.gov/mpi/mpich/*.

Gropp, W. and B. Smith (1993). User manual for Chameleon parallel programming tools, Argonne National Laboratory.

Guillard, H. and C. Viozat (1999). On the behaviour of upwind schemes in the low Mach number limit. Computer & Fluids 28, pp. 63-86.

Hammond, S. W. and T. J. Barth (1992). Efficient Massively Parallel Euler Solver for Two-Dimensional Unstrucrured Grids. AIAA Journal 30(4), pp. 947-952.

Haselbacher, A. and J. Blazek (2000). Accurate and Efficient Discretization of Navier-Stokes Equations on Mixed Grids. AIAA Journal 38(11), pp. 2094-2102.

Haselbacher, A., J. J. McGuirk, et al. (1999). Finite Volume Discretization Aspects for Viscous Flows on Mixed Unstructured Grids. AIAA Journal 37(2), pp. 177-184.

Hawken, D. F. (1991). Review of some adaptive node-move techniques in finite element and finite difference solutions of partial differential equations. J. Comp. Phys. 95, pp. 254-302.

He, L. (1993). New two-grid acceleration method for unsteady Navier-Stokes calculations,. Journal of Propulsion and Power 9(2), pp. 272-280.

He, L. (1998). Unsteady Flow in Oscillating Turbine Cascades: Part 1 - Linear Cascade Experiment. ASME Journal of Turbomachinery 120(4), pp. 262-268.

He, L. (1998). Unsteady Flow in Oscillating Turbine Cascades: Part 2 - Computational Study. ASME Journal of Turbomachinery 120, pp. 269-275.

Hirsch, C. (1990). Numerical Computation of Internal and External Flows, John Wiley & Sons Ltd. Baffins Lane, Chichester, West Sussex PO19 1UD. England.

Holmes, D. G. (1994). 'Numerical methods for flow calculation in turbomachines'. von Karman Institute for Fluid Dynamics Lecture Series 1994-06

Holmes, D. G. (1994). 'Unstructured grids and mesh adaptivity for inviscid and viscous flows'. VKI Lecture Series 1994-06

Ilinca, F., D. Pelletier, et al. (1997). An adaptive finite element scheme for turbulent free shear flows. IJCFD 8, pp. 171-188.

Jameson, A. (1994). 'Analysis and design of numerical schemes for gas dynamics 1: Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence'. NASA Center for AeroSpace Information (CASI) NASA-CR-196477

Jameson, A., T. J. Baker, et al. (1986). 'Calculation of inviscid transonic flow over a complete aircraft'. AIAA paper AIAA-86-0103

Jameson, A. and D. J. Mavriplis (1986). Finite volume solution of two-dimensional Euler equations on a regular triangular mesh. AIAA Journal 24.

Jameson, A., W. Schmidt, et al. (1981). 'Numerical Solutions of Euler Equations by Finite Volume Methods Using Runge-Kutta Time-stepping Schemes'. AIAA paper 81-1259

Jameson, A., W. Schmidt, et al. (1985). 'Numerical Solutions of the Euler Equationsby Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes'. AIAA paper 81-1259

Jin, H. and R. Tanner (1993). Generation of unstructured tetrahedral meshes by the advancing front technique. International Journal for Numerical Methods in Engineering 36, pp. 1805-1823.

Kang, S. and C. Hirsch (2001). Numercial Investigation of The 3D Flow Around NREL Untwisted Wind Turbine Blades. In Turbomachinery Fluid Dynamics and Thermodynamics, Firenze Italy.

Karypis, G. and V. Kumar (1995). 'A fast and high quality multilevel scheme for partitioning irregular graphs'. Computer Science Department, University of Minneasota MN55455 Technical Report TR 95-035

Karypis, G. and V. Kumar (1998). METIS: A Software Package for Partitioning Unstructured Graphs, Partiiotning Meshes, and Computing Fill-Reduce Orderings of Sparse Matrices. University of Minnesota, Department of Computer Science / Army HPC Research Center, Minneapolis, MN 55455.

Knight, D. (1993). 'A fully implicit Navier-Stokes algorithm using unstructuted grid and flux diference spilitting'. AIAA paper 93-0875

Kunz, R. F. and B. Lakshminarayana (1992). Explicit Navier-Stokes Computation of Cascade Flows Using the k-e Turbulence Model. AIAA Journal 30(1).

Kunz, R. F. and B. Lakshminarayana (1992). Explicit Navier-Stokes Computation of Cascade Flows Using the k-ε Turbulence Model. AIAA Journal 30(1), pp. 13-22.

Ladkany, S. G. (1998). 'Proposed Wind Turbine Aeroelasticity Studies Using Helicopter Systems Analysis'. NASA Center for AeroSpace Information (CASI)

Lallemand, M.-H., H. Steve, et al. (1992). Unstructured Multigridding by Volume Agglomeration: current status. Computers & Fluids 21(3), pp. 397-433.

Leer, B. V. (1979). Toward the ultimate conservative difference scheme V, A second order squel to Godunov's method. Journal of Comput. Phys. 32, pp. 101-136.

Liu, C. Y. and C. J. Hwang (2001). New Strategy for Unstructured Mesh Generation. AIAA Journal 39(6), pp. 1078-1085.

Lo, S. H. (1985). A New Mesh Generation Scheme for Arbitrary Planar Domains. International Journal for Numerical Methods in Engineering 21(8), pp. 1403-1426.

Lohner, R. and J. Cebral (2000). Generation of non-isotropic unstructured grids via directional enrichment. International Journal for Numerical Methods in Engineering 49, pp. 219-232.

Lohner, R. and P. Parikh (1988). Three-dimensional grid generation by the advancing front method. International Journal for Numerical Methods in Fluids 8, pp. 1135-1149.

Marvriplis, D. J. (1990). Accurate Multigrid Solution of Euler Equations on Unstructured and Adaptive Meshes. AIAA Journal 28(2), pp. 213-221.

Marvriplis, D. J. (1991). Algebraic turbulence modelling for unstructured and adaptive meshes. AIAA Journal 29, pp. 2086-2093.

Marvriplis, D. J. (1992). Three-dimensional multigrid for Euler equations. AIAA Journal 30(2), pp. 1753-1761.

Marvriplis, D. J. and A. Jameson (1987). 'Multigrid Solution of Euler Equations on Unstructured and Adaptive Meshes'. ICASE Report 87-53

Mavriplis, D. J. (1996). 'Multigrid Solution Strategies for Adaptive Meshing Problems'. NASA 19480

Mavriplis, D. J. (1999). Directional Agglomeration Multigrid Techniques for High-Reynolds-Number Viscous Flows. AIAA Journal 37(10), pp. 1222-1230.

Mavriplis, D. J. (2000). 'Parallel performance investigations of an unstructured mesh Navier-Stokes solver'. NASA Langley Research Center ICASE Report 2000-13; NASA/CR-2000-210088

Mavriplis, D. J. (2000). Viscous Flow Analysis Using a Parallel Unstrictured Multigrid Solver. AIAA Journal 38(11), pp. 2067-2076.

Mavriplis, D. J. (2002). 'Transonic Drag Prediction using an Unstructured Multigrid Solver'. NASA Center for AeroSpace Information (CASI) NASA/CR-2002-211455

Mavriplis, D. J. and V. Venkatakrishnan (1995). Agglomeration multigrid for tow dimentional viscous flows. Journal of Comput. Phys. 24, pp. 553-570.

Merriam, M. L. (1991). 'An Efficient Advancing Front Algorithm for Delaunay triangulation'. AIAA Paper 91-0792

Mitchell, C. R. (1994). 'Improved Reconstruction Schemes for the Navir-Stokes Equations on Unstructured Meshes'. AIAA paper 94-0642

Muller, J.-D. 'On triangles and flow' (University of Michigan, PhD thesis, 1996)

NASA (1999). **"NPARC Alliance Validation Archive: ONERA M6 Wing Study #1"**. *Http://www.grc.nasa.gov/www/wind/valid/m6wing/m6wing01/m6wing01.html.*

Ollivier-Gooch C. (1998). **"GRUMMP--- Generation and Refinement of Unstructured, Mixed-Element Meshes in Parallel"**. *http://tetra.mech.ubc.ca/GRUMMP/index.html.*

Ollivier-Gooch, C. (2001). Coarsening Unstructured Meshes by Edge Contraction. Int. J. Numer. Mech. Engng.

Oxford university computing laboratory (2000). **"BSP: A New Industry Standard for Scalable Computing on Clusters, SMPs and MPPs"**. *http://web.comlab.ox.ac.uk/oucl/work/bill.mccoll/oparl.html.*

Owen, S. (1998). **"A Survey of Unstructured Mesh Generation Technology"**. *http://www.andrew.cmu.edu/user/sowen/survey/index.html.*

Peiro, J. and A. I. Sayma (1995). A 3-D unstructured multigrid Navier-Stokes solver. In ICFD Conference on Numerical Methods for Fluid Dynamics, Oxford University Press.

Pelletier, D. and F. Ilinca (1997). Adaptive remeshing for the k-e model of turbulence. AIAA Journal 35(4).

Pirzadeh, S. (1993). Structured Background Grids for Generation of Unstructured Grids by Advancing-Front Method. AIAA Journal 31(2), pp. 257-265.

Pothen, A., G. Intemann, et al. (1993). 'Wing Pylon Fillet Design Using Unstructured Mesh Euler Solver'. AIAA paper 93-3500

Pothen, A., H. D. Simon, et al. (1990). Partitioning sparse matrices with eigenvectors of graphs. SIAM J. Mat. Anal. Appl. 11(3), pp. 430-452.

PVM (1995). **"PVM: Parallel Virtual Machine"**. *http://www.csm.ornl.gov/pvm/pvm_home.html.*

Rizzi, A., P. Eliasson, et al. (1992). The engineering of multiblock/multigrid software for Navier-Stokes flows on structured meshes. Computers Fluids 22(2/3), pp. 341-367.

Roberts, T. W., D. Sidikover, et al. (1997). 'Textbook Multigrid Efficiency for the steady Euler equations'. AIAA paper 97-1949

Roe, P. L. (1981). Approximate Riemann, parameter vectors and difference schemes. Journal of Comp. Phys. 43, pp. 357-372.

Roehl, C. and H. Simon (1999). <u>Numerical simulation of compressible flows with adaptive unstructured grids</u>. In Proceedings of the 3rd ASME/JSME joint fluids engineering conference, San Francisco, California, USA.

Sayma, A. I., M. Vahdati, et al. (2000). Modeling of Three-Dimensional Viscous Compressible Turbomachinery Flows Using Unstructured Hybrid Grids. <u>AIAA Journal</u> 38(6), pp. 945-954.

Sbardella, L. and M. Imregun (2000). An Efficient Discretisation of Viscous Fluxes on Unstructured mixed-elements Grids. <u>Communications in Numerical Methods in Engineeering</u> 16, pp. 839-849.

Sbardella, L. and M. Imregun (2000). An efficient discretization of viscous fluxes on unstructured mixed-element grid. <u>Communications in Numerical Methods in Engineeering</u> 16, pp. 839-849.

Sbardella, L., A. I. Sayma, et al. (1997). <u>Semi-unstructured mesh generator for flow calculations in Axis turbomachinery blading</u>. In 8th International Symposium on Unsteady Aerodynamics and Aeroelasticity of Turbomachines, Stockolm.

Sbardella, L., A. I. Sayma, et al. (1998). Semi-structured Meshes for Axial Turbomachinery Blades. <u>International Journal for Numerical Methods in Fluids</u> 32, pp. 569-584.

Schepers, J. G., A. J. Brand, et al. (1997). 'Final report of IEA Annex XIV: Field rotor aeronautics'. ECN-C-97-027

Schmitt, V. and F. Charpin (1979). 'Pressure Distributions on the ONEAR M6-wing at Transonic Mach Number'. AGARD Advisory Report 138

Sheng, C., L. K. Tylor, et al. (1995). 'Multiblock Multigrid Solution of Three-Dimensional Compressible Turbulent Flows About Appended Submarine Configuration'. AIAA paper 95-0203

Sheng, C., D. L. Whitfield, et al. (1999). Multiblock Approach for Calculating Incompressible Fluid Flows on Unstructured Grids. <u>AIAA Journal</u> 37(2), pp. 169-176.

Siden, G. L. D., W. N. Dawes, et al. (1990). Numerical simulation of two-dimensional viscous compressible flow in blade cascades using a solution-adaptive unstructured mesh. <u>ASME Journal of Turbomachinery</u> 112.

Simon, H. D. (1991). 'Partitioning of unstructured problems for parallel processing'. NASA Ames Research Center, Numerical Aerodynamic Simulations System Division

Spalart, P. R. and S. R. Allmaras (1992). 'A one-equation turbulence model for aerodynamic flows'. AIAA paper 92-0439

Spalart, P. R. and S. R. Allmaras (1992). 'A one-equation turbulence model for aeronautic flows'. AIAA paper 92-0439

Steinthorsson, E., M. S. Liou, et al. (1993). 'Development of an explicit Multiblock/Multigrid flow solver for viscous flows in complex geometries'. AIAA paper 93-2380

Swanson, R. C. (2001). <u>Towards Optimal Multigrid Efficiency for the Navier-Stokes Equations</u>. In 15th AIAA Computational Fluid Dynamics Conference, Anaheim, California, AIAA 2001-2574.

Thomas, J. L. and M. D. Salas (1986). Far-field Boundary Conditions for Transonic Lifting Solutions to Euler Equations. <u>AIAA Journal</u> 24(2), pp. 1074-1080.

Venkatakrishnan, V. and T. J. Barth (1989). 'Application of direct solvers to unstructured meshes for the Euler and Navier-Stokes equations using upwind schemes'. AIAA paper 89-0364

Venkatakrishnan, V. and D. Mavriplis (1994). 'Agglomeration Multigrid for 3D Euler Equations'. AIAA paper 94-0069

Venkatakrishnan, V., H. D. Simon, et al. (1991). 'A MIMD implementation of a parallel Euler solver for unstructured grids'. NASA AMES Research Center Tech. Report RNR-91-024

Venkatakrishnan, V., H. D. Simon, et al. (1991). 'A MIMD Implementation of Parrallel Euler Solver for Unstructured Grids'. NASA Ames R. C. Tech. Report RNR-91-024

Vilsmeier, R. and D. Hanel (1993). Adaptive methods on unstructured grids for Euler and Navier-Stokes equations, Computers Fluids. <u>Computers & Fluids</u> 22(4/5), pp. 485-499.

Wang, Q., S. J. Massey, et al. (1999). 'Solving Navier-Stokes Equations with Advanced turbulence models on three-dimensional unstructured grids'. AIAA paper 99-0156

Warren, G. P., W. K. Anderson, et al. (1991). 'Grid Convergence for Adaptive Methods'. AIAA paper 91-1592CP

Waston, D. F. (1981). Computing the n-Dimensional Delaunay Tessallation with Application to Voronoi Polytopes. <u>Computer Journal</u> 24(2), pp. 167-172.

Wesseling, P. (1999). 'Unified methods for computing incompressible & compressible flows'. von Karman Institute for Fluid Dynamics Lecture Series 1999-03

Wiel, S. V., D. Nathanson, et al. (1996). 'Performance and Program Complexity in Contemporary Network-based Parallel Computing Systems'. University of Minnesota, Technical Report HPC-96-02

Wilcox, T. (1993). <u>Turbulence modeling for CFD</u>, Griffin, Glendale, CA.

Wood, W. A. and W. L. Kleb (1998). 'Diffusion Characteristics of Upwind Schemes on Unstructured Triangulations'. AIAA paper 98-2443

Zheng, Y. 'CFD Simulation of Transonic Flows in a Turbine Cascade' (Beijing University of Aeronautics and Astronautics, Master thesis, 1995)

Zheng, Y. and L. He (2001). Multigrid upwind Euler/Navier-Stokes computational on adaptive unstructured meshes. <u>The Aeronautical Journal</u> 105(1046), pp. 173-184.