A Cut-cell, Agglomerated-Multigrid Accelerated, Cartesian Mesh Method for Compressible and Incompressible Flow

John Pattinson

Department of Mechanical and Aeronatical Engineering

University of Pretoria

Supervised by Prof. A.G. Malan and Prof. J.P. Meyer

A thesis submitted for the degree of

MEng

August 2006

Acknowledgements

I would like to give recognition and thanks to Professor Arnaud G. Malan for his guidance and support as my primary supervisor. I would also like to thank my co-supervisor Prof. J.P. Meyer for his support. Further thanks must go to Dr. C. P. Crosby, for his foresight in being the original initiator of the work. Special recognition and thanks is also due to Angus Brown, Prevani Kistan, John O'Mahony and Jonathan Fouche at Denel for their interest and support. Finally I need to thank my mother for sorting out some of my hieroglyphics into readable text. This work was sponsored by Denel Aerospace Systems as well as the South African National Research Foundation (Grant number: 2053287) and THRIP (Grant number: 3257).

Abstract

This work details a multigrid-accelerated cut-cell Cartesian mesh methodology for the solution of a single partial differential equation set that describes incompressible as well as compressible flow. The latter includes sub-, trans- and supersonic flows. Cut-cell technology is developed which furnishes body-fitted meshes with an overlapping Cartesian mesh as starting point, and in a manner which is insensitive to surface definition inconsistencies. An edge-based vertex-centred finite volume method is employed for the purpose of spatial discretisation. Further, an alternative dual-mesh construction strategy is developed and the standard discretisation scheme suitably enhanced. Incompressibility is dealt with via a locally preconditioned artificial compressibility algorithm, and stabilisation is in all cases achieved with scalar-valued artificial dissipation. In transonic flows, shocks are captured via pressure switch-activated upwinding. The solution process is accelerated by the use of a full approximation scheme (FAS) multigrid method where coarse meshes are generated automatically via a volume agglomeration methodology. The developed modelling technology is validated by application to the solution of a number of benchmark problems. The standard discretisation as well as the alternative method are found to be equivalent in terms of both accuracy and computational cost. Finally, the multigrid implementation is shown to achieve decreases in CPU time of between a factor two to one order of magnitude. In the context of cut-cell Cartesian meshes, the above work has resulted in the following novel contributions: the development of an alternative vertex-centred discretisation method; the use of volume agglomerated multigrid solution technology and the use of a single equation set for both incompressible and compressible flows.

Contents

List of Figures

Nomenclature

Roman Symbols

$\mathbf{A}$	Coefficient matrix
$\mathcal{A}$	Surface (m^2)
A_p	Artificial compressibility pressure sensor
a	Least squares plane coefficient
a_r	Artificial compressibility related parameter
a_u	Generalised artificial compressibility preconditioning coefficient
$\mathbf{B}$	Boundary edge coefficient
$\mathbf{C}$	Edge coefficient
C	Artificial dissipation approximation coefficient
C_p	Specific heat at constant pressure (J/kgK)
C_v	Specific heat at constant volume (J/kgK)
CFL	Courant-Friedrichs-Lewy number
c	Acoustic velocity (m/s)
c_τ	Pseudo acoustic velocity
$\mathbf{D}$	Stabilisation term

d	Distance between points (m)
E	Total specific energy (J/kg)
$\mathbf{e}$	Error between values approximated on a coarse mesh and those restricted up from the finer mesh
$\mathbf{F}$	Flux vector
$\mathcal{F}$	Safety factor for GCI calculation
f	Characteristic parameter of solution e.g. lift
f_{exact}	Estimate of exact solution at zero grid spacing
G	Riemann invariant
GCI	Grid Convergence Index
H	Total specific enthalpy (J/kg)
I	Grid transfer operator. I_h^{2h} is known as restriction and I_{2h}^h is known as prolongation. Grid transfer occurs from subscript to superscript.
L	Characteristic / non-dimensional length (m)
M	Number of points used in a least squares approximation of a plane
m	Equation order
$\mathbf{n}$	Unit normal vector
$\mathcal{P}$	Function value at a point
P	Pressure switch
$\mathtt{p}$	Order of convergence
p	Pressure (Pa)
$\mathbf{Q}$	Conservative variables vector

RHS	Right hand side of algebraic equation
R	Gas constant (J/kgK). Taken as 287 J/KgK for air
$\mathtt{r}$	Grid refinement ratio
r	Ratio of the distance from node 0 on an edge to the intersection of the edge with its edge coefficient over the total length of the edge.
r_{asym}	Asymptotic convergence ratio
T	Temperature (K)
t	Time (s)
U_∞	Free stream velocity (m/s)
$\mathbf{u}$	The solution on a coarse mesh as restricted from the finer mesh
u	Velocity (m/s)
$\mathcal{V}$	Computational volume (m^3)
V	V-cycle operator
$\mathbf{v}$	Approximate solution to problem
v_1	Number of relaxation sweeps on the downward traverse of the V-cycle
v_2	Number of relaxation sweeps on the upward traverse of the V-cycle
$\mathbf{W}$	Density dependant conservative variables vector
$\mathbf{x}^*$	Centroid coordinate (m)
x	Position coordinates (m)

Greek Symbols

α	Runge-Kutta weighting coefficient
η	Compressible / incompressible switch

ε_2	Empirical constant used in shock capturing dissipation formulation
ε_4	Empirical constant used in JST dissipation formulation
ε_{c_τ}	Minimum pseudo acoustic velocity
γ	Ratio of a specific heats, $\frac{C_p}{C_v}$. Taken as 1.4 for air
Λ	Dissipation scaling factor
ϕ	Exact solution to problem on the finest mesh
ρ	Density (kg/m^3)
τ	Multigrid coarse grid correction or source term
Υ	Refers to an edge. Υ_{mn} refers to the edge connecting nodes m and n
$\Delta\chi$	Local measure of element / cell size (m)

Superscripts

B	Boundary entity
h	Mesh spacing. $2h$ would refer to double mesh spacing or one level of coarsening from h.
JST	Jameson, Schmidt & Turkel [1] dissipation term
sc	Shock capturing term
tet	Tetrahedron
$*$	Dimensional quantity

Subscripts

b	Boundary entity
e	Extrapolated quantity (from the interior of the domain)
∞	Subscript denotes free-stream value

i	subscript index
j	subscript index
m	Node index
n	Node index

Mathematical Operators

δ_{ij}	Kronecker delta function: unity if $i = j$ and zero if $i \neq j$
$\mid \bullet \mid$	Euclidean norm of $\mid \bullet \mid$ (absolute value in case of scalar)
∇	Gradient operator
$\Delta\bullet$	Increment in $\bullet$

Notes on Notation

In this thesis, both vector and index notation are used. Vectors and matrices are printed in bold. Where index notation is used, component subscripts appear as subscripts typically denoted by i, j and k. Einstein's summation convention is implied in the case of index notation.

Chapter 1

Introduction

1.1 Background and Project Motivation

Computational fluid dynamics (CFD) is a growing field which is focused on obtaining approximate solutions, via numerical means, to a set of equations that describe the motion of a fluid on a macroscopic or continuum level. The aforementioned numerical solution strategy has become essential for engineering related flow systems where analytical methods are not at present adequate. The use of these numerical techniques is impractical without the aid of a computer and as a result, it was only in the 1970's that CFD came into its own as a field. Since then, with the massive advances in computer technology and advances in solution algorithms, CFD has become an important tool in the design of vehicles, buildings and other structures and devices that interact in some way with a fluid. At this time however, the accuracy of solutions to problems with complex flows is not, as a general rule, of sufficient quality to accurately quantify all flow features found in a field such as aeronautics. It does however hold great promise to describe the pressure field around arbitrary streamlined bodies in regions where flow break-away or separation is not present. This is of practical value when screening potential aerodynamic design concepts to obtain estimates of lift and the positions of certain shocks.

The study group under which this work was completed, was approached by DENEL Aerospace systems to develop an in-house tool that could replace or supplement the analytical / empirical codes currently employed for concept phase screening. Typical aerodynamic systems of relevance are geometrically complex streamlined bodies which range from subsonic unmanned aerial vehicles (UAV's) to supersonic missiles. The technology to be developed is a CFD code which is capable of quantifying physics such as out-of-boundary layer flows as well as body surface pressure distributions. This is to be done over complex geometries while being a precursor to more expensive and detailed CFD analyses which include viscous flow effects.

1.2 Overview

As discussed in the previous section, the aim of this project is to design a CFD tool that can be used effectively in the concept phase of an aerodynamic design. The development of this tool starts with the selection of the governing equations.

The equation set that is most accepted to describe the fluid flows under consideration is generally referred to as the Navier-Stokes equations[1]. Ideally one would like to solve these equations directly but this is prohibitively expensive to do on complex aerodynamic geometries at the current time. The terms in the Navier-Stokes equations that require the most computational resources are the so called viscous terms. When these terms are omitted, the so-called Euler equations result. For globally attached flow at large Reynolds numbers, it has been shown that the Euler equations predict the out-of-boundary layer flows as well as body surface pressure distributions with reasonable accuracy [2]. As a consequence, in this work, these equations will be in essence solved. The classic form of the equations will be altered such that incompressible and compressible flow may be modelled with the same equation set. This is the focus of Chapter 2.

Once the governing equations are chosen, it remains to solve them in an efficient and accurate manner. This solution process is the major focus of this work. There are many ways to solve the Euler equations over complex domains. Most methods begin by dividing the domain of interest into smaller non-overlapping volumes and solving for the equations on each volume. The aforementioned process is called mesh generation and is the subject of Chapter 3. In this work

[1] Note that this is a misnomer, as the Navier-Stokes equations in reality refer to the momentum equations only.

non-conforming cut-cell Cartesian meshes are used as they can be generated with great efficiency on complex geometries [3]. This is ideal for the concept design process, particularly where minor inconsistencies exist in CAD geometry surfaces (known as “dirty” geometries). In the case of the latter, it can take more time to generate the mesh, via body-fitted type mesh generation techniques, than to solve the actual flow problem [4].

Once the domain has been decomposed into small non-overlapping volumes, the Euler equations need to be “discretised” on each. This process involves transforming the analytical equations into weak form and generating approximate algebraic expressions which may be solved numerically. This process is the subject of Chapter 4. As the solver developed in this work is to be started from scratch, it was deemed an ideal opportunity to test alternative discretisation and stabilisation options. In this work therefore, the vertex-centred finite-volume technique is used whereas most investigators use the cell-centred discretisation. The former technique should be competitive with the latter scheme and it has not been extensively investigated in the context of Cartesian meshes. The vertex-centred discretisation requires the construction of a so called *dual mesh*. In this work, an alternative dual mesh construction is proposed in addition to the standard method. These algebraic equations are stabilised using the artificial dissipation scheme of Jameson *et al.* [1]. This scheme is also at present not popular in the context of Cartesian meshes.

Following on from the spatial discretisation, the equations produced by the discretisation process have to be solved. In this work only the steady state solution to the Euler equations is sought. This is achieved through an iterative solution procedure in which the governing equation transient term is manipulated to render a Jacobi like process. Although this process is superbly memory efficient, many iterations are required in order to reach a converged solution. For this reason, in this work, solution acceleration is effected through the use of the multigrid procedure of Brandt [5] in its nonlinear form vis. Full Approximation Scheme (FAS) multigrid. This method requires the generation of successive coarse meshes for which the agglomerated grid coarsening strategy of Lallemand *et al.* [6] is employed. The description of the discretisation of the temporal term in the Euler equations as well as the subsequent acceleration of the solution process using volume agglomerated multigrid is the subject of Chapter 5.

The developed solver is then evaluated by applying it to a variety of test cases that encompass the entire flow range of interest, vis. from incompressible subsonic flow right through to compressible supersonic flow. The solutions obtained are compared to either analytical solutions or published results of others. Grid convergence studies are performed in all cases to ensure rigorous evaluation, while the speed-ups obtained through the use of the agglomerated Multigrid method are documented. The evaluation process is detailed in Chapter 6.

The work done as well as the most important findings are documented in Chapter 7. Further, recommendations are also made for future work.

1.3 Publication List

The publications forthcoming from the research follow[1]:

1.3.1 Journal Papers

- Pattinson, J., Malan, A.G. & Meyer, J.P. (2006). An agglomerated FAS multigrid accelerated cut-cell non-collocated Cartesian mesh method for incompressible and compressible flow. *South African Journal of Science*, **Under review**.

- Pattinson, J., Malan, A.G. & Meyer, J.P. (2006). A cut-cell non-collocated Cartesian mesh method for compressible and incompressible flow. *International Journal for Numerical Methods in Engineering*, **Under review**.

1.3.2 Conference Papers

- Malan, A.G., Pattinson, J. & Meyer, J.P. (2005). Modelling incompressible flow on cut-cell Cartesian meshes. 4th International Conference

[1] The full text versions of these publications can be found in the CD attached to this thesis.

on Heat Transfer, Fluid Mechanics, and Thermodynamics, HEFAT2005, Cairo, Egypt.

- Malan, A.G., Pattinson, J. & Meyer, J.P. (2006). An edge-based methodology for modelling incompressible flow on cut-cell non-collocated Cartesian meshes. 5th SA Conference on Computational and Applied Mechanics, SACAM06, Cape Town, South Africa.

- Pattinson, J., Malan, A.G. & Meyer, J.P. (2006). An edge-based methodology for modelling compressible flow on cut-cell non-collocated Cartesian meshes. 5th SA Conference on Computational and Applied Mechanics, SACAM06, Cape Town, South Africa.

1.3.3 Technical Reports

- Malan, A.G., Pattinson, J. & Meyer, J.P. (2004). Development of a fast non-conforming Cartesian mesh Euler solver for application to missile design: Preliminary literature survey: Discretization and solution strategy. *for Denel Aerospace Systems (Pty) Ltd*, **Report number 0403**, 1–18.

- Malan, A.G., Pattinson, J. & Meyer, J.P. (2004). Development of a fast non-conforming Cartesian mesh Euler solver for application to missile design: Preliminary report on the discretization: Developed 2-D mesh-cutting and CFD capabilities. *for Denel Aerospace Systems (Pty) Ltd*, **Report number 0405**, 1–21.

- Pattinson, J., Malan, A.G. & Meyer, J.P. (2005). Development of a fast non-conforming Cartesian mesh Euler solver for application to missile design: Multigrid solution acceleration: Developed solver technology. *for Denel Aerospace Systems (Pty) Ltd*, **Report number 0503**, 1–8.

- Pattinson, J., Malan, A.G. & Meyer, J.P. (2005). Development of a fast non-conforming Cartesian mesh Euler solver for application to missile design: Multigrid solution acceleration: Mesh agglomeration technology. *for Denel Aerospace Systems (Pty) Ltd*, **Report number 0504**, 1–11.

- Pattinson, J., Malan, A.G. & Meyer, J.P. (2005). Development of a fast non-conforming Cartesian mesh Euler solver for application to missile design - Multigrid solution acceleration: Further results. *for Denel Aerospace Systems (Pty) Ltd*, **Report number 0506**, 1–17.

- Pattinson, J., Malan, A.G. & Meyer, J.P. (2006). Development of a fast non-conforming Cartesian mesh Euler solver for application to missile design: Developed 3D preprocessor. *for Denel Aerospace Systems (Pty) Ltd*, **Report number 0602**, 1–10.

1.4 Purpose of Study

To summarise, the purpose of this study is to develop a CFD tool that can be used effectively in the concept phase of an aerodynamic design. To do this a solver for the compressible and incompressible Euler equations is to be developed in which alternative methods, not currently in widespread use in the context of Cartesian meshes, for discretisation and stabilisation are to be investigated. Further, the solver is to employ an explicit solution method accelerated with non-linear multigrid and using volume agglomeration to generate the coarse meshes.

Chapter 2

Problem Formulation

2.1 Introduction

The aim of this project is to model, to engineering accuracy, the steady-state out-of-boundary layer flow patterns and pressures around complex geometries subject to sub-, trans- and supersonic flow conditions. To do this efficiently, current researchers [7, 8, 9, 10, 11, 12, 13] employ a subset of the Navier-Stokes equations: the time-dependent compressible Euler equations[1]. The application of these equations to model both compressible and purely incompressible flow in a meaningful manner is the subject of this chapter.

2.2 Governing Equations

2.2.1 Euler Equations

The system of partial differential equations that describes the inviscid flow of an incompressible or compressible fluid, contains relations that enforce the principles of mass, momentum and energy conservation. The governing system of equations, with artificial compressibility implemented, may be written for a Cartesian coor-

[1] Note that as per many others, the steady state solution of the Euler equations is obtained by solving the transient system of governing equations from a given initial condition.

dinate system in the following non-dimensional conservative form

$$\frac{\partial \mathbf{W}}{\partial \mathbf{Q}}\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}^j}{\partial x_j} = \mathbf{0} \tag{2.1}$$

where

$$\mathbf{W} = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} p \\ u_1 \\ u_2 \\ \eta T \end{pmatrix}, \quad \mathbf{F}^j = \begin{pmatrix} \rho u_j \\ \rho u_1 u_j + p\delta_{1j} \\ \rho u_2 u_j + p\delta_{2j} \\ \eta(\rho E + p) u_j \end{pmatrix} \tag{2.2}$$

and t is time, ρ is the density; u_j the velocity component in direction x_j, E is the specific *total energy* (as given below), T is temperature, and p is the pressure. Further, δ_{ij} refers to the Kronecker delta function while η switches compressible flow specific terms on and off as:

$$\eta = \begin{cases} 1 & \text{for Compressible Flow} \\ 0 & \text{for Incompressible Flow} \end{cases} \tag{2.3}$$

In the case of incompressible flow, the above definition results in the isothermal form of the equations in which the energy equation plays no role. The artificial compressibility preconditioning matrix $\frac{\partial \mathbf{W}}{\partial \mathbf{Q}}$ is defined as

$$\frac{\partial \mathbf{W}}{\partial \mathbf{Q}} = \begin{pmatrix} \frac{1}{c^2} & 0 & 0 & 0 \\ \frac{a_u u_1}{c^2} & \rho & 0 & 0 \\ \frac{a_u u_2}{c^2} & 0 & \rho & 0 \\ \frac{E}{c^2} & \frac{\rho}{2} & \frac{\rho}{2} & \rho C_p \end{pmatrix} \tag{2.4}$$

where C_p denotes specific heat at constant pressure and a_u is the generalised artificial compressibility preconditioning coefficient. For compressible flow, a_u is 1 while for incompressible flow a_u is calculated in its localised form as proposed by Malan *et al.* [14] as

$$a_u = 2(1 - A_p) \tag{2.5}$$

where A_p is a pressure sensor. In one dimension where $p(x)$ is the pressure and x^m is a point in space, the latter is defined as

$$A_p = \lim_{x \to x^{m+}} \frac{|\nabla p(x^m) - \nabla p(x))|}{|\nabla p(x^m)| + |\nabla p(x))|} \tag{2.6}$$

where

$$\nabla p(x^m) = \lim_{x \to x^{m-}} \frac{p(x) - p(x^m)}{x - x^m} \tag{2.7}$$

This definition results in a_u varying from 2 in smooth regions to nearly 0 in regions with high gradients. The acoustic velocity c is calculated as follows:

$$c = \eta\sqrt{\gamma RT} + (1 - \eta)c_\tau \tag{2.8}$$

where γ is the ratio of specific heats and R is the gas constant. Also, c_τ is the pseudo-acoustic velocity as defined by Malan *et al.* [14]:

$$c_\tau = \begin{cases} \varepsilon_{c_\tau} & \text{if } |\mathbf{u}| \le \varepsilon_{c_\tau} \\ |\mathbf{u}| & \text{if } |\mathbf{u}| > \varepsilon_{c_\tau} \end{cases} \tag{2.9}$$

where $|\mathbf{u}| = \sqrt{u_j u_j}$ and ε_{c_τ} is typically set to $10^{-5}V_{max}$ with V_{max} being the maximum velocity magnitude in the field. ε_{c_τ} is defined in this manner to ensure that the pseudo-acoustic velocities do not go to zero at stagnation points.

In all of the above equations, non-dimensional quantities are related to their dimensional counterparts (depicted with superscript *) through the following relations

$$\begin{array}{cccc} t = \frac{t^* U^*_\infty}{L^*} & u_j = \frac{u^*_j}{U^*_\infty} & p = \frac{p^*}{\rho^*_\infty U^{*2}_\infty} & T = \frac{T^*}{U^{*2}_\infty / C_p} \\ \rho = \frac{\rho^*}{\rho^*_\infty} & x_j = \frac{x^*_j}{L^*} & E = \frac{E^*}{U^{*2}_\infty} & \end{array} \tag{2.10}$$

where t, L and T are respectively time, the characteristic length and temperature. Further, the subscript ∞ denotes free-stream conditions with U_∞ being the free-stream velocity.

2.2.2 Constitutive Equations

For the case of compressible flow, closure to the above formulation is obtained by assuming the gas to be ideal. The implication is that the following relations hold

$$p^* = \rho^* R T^* \tag{2.11}$$

where R is the gas constant given by $R = C_p - C_v$ where C_v is specific heat at constant volume. In this project it is assumed that the ratio of specific heats γ is constant. Thus

$$\gamma = \frac{C_p}{C_v} \tag{2.12}$$

which results in

$$R = C_p \frac{(\gamma - 1)}{\gamma} \tag{2.13}$$

The resulting gas law written in terms of non-dimensional variables follows

$$\rho = \frac{p}{T} \frac{\gamma}{\gamma - 1} \tag{2.14}$$

Further, the specific total energy is given by

$$E = \frac{T}{\gamma} + \frac{u_j u_j}{2} \tag{2.15}$$

where the first term on the right-hand-side embodies internal energy. The corresponding non-dimensional expression for *total specific enthalpy* is

$$H = \frac{T}{\gamma} + \frac{p}{\rho} + \frac{u_j u_j}{2} \tag{2.16}$$

Finally, the dimensional acoustic velocity is given by

$$c^* = \sqrt{\gamma R T^*} \tag{2.17}$$

or in terms of non-dimensional relations

$$c = \sqrt{(\gamma - 1) T} \tag{2.18}$$

where $c = c^*/U_\infty$. The above description of the physical system is only valid for media which may be accurately described as a continuous medium, and is as a result not applicable to rarefied gas (low densities) such as would be the case at the outer bounds of the atmosphere. It is further assumed that heat transfer through radiation is negligible and that no chemical reactions take place in the gas. Finally, gravitational body-forces have been assumed negligible, which is a reasonable assumption in the case of convection dominated flows [15].

2.2.3 Boundary Conditions

For a unique solution to the Euler equations, appropriate boundary conditions are to be prescribed. These include the application of artificial boundary conditions to the outer boundary which represents a truncated version of the spatial domain. In the context of this work, two types of boundaries are encountered vis. solid walls and far-field boundaries. In the case of the latter, characteristic type boundary conditions are employed (this is to account for the truncation of the domain).

The application of boundary conditions on the far-field of a domain depends on the local Mach number (or pseudo Mach number in the case of incompressible flow) and whether the flow is leaving or entering the domain. The four cases that are encountered in this work are as follows: supersonic inlet, supersonic outlet, incompressible as well as compressible subsonic in- and outlet conditions. At the supersonic inlet, free stream conditions are prescribed, while at the outflow all the flow variables are extrapolated from the interior of the domain via the method described in Section 4.7. At a subsonic inlet, three flow characteristics must enter the domain and one must leave, while at a subsonic outflow three characteristics must leave and one must enter the domain. To determine which characteristics must enter the domain and which must leave, an altered characteristic type analysis based on one-dimensional Riemann invariants [7] is used. The alteration is required to extend the method's applicability to incompressible flow systems. The procedure is outlined next.

Riemann invariants G, based on the free-stream and extrapolated values are calculated as:

$$G_\infty = \mathbf{u}_\infty \cdot \mathbf{n} - \frac{2c_\infty}{\gamma - 1} \tag{2.19}$$

$$G_e = \mathbf{u}_e \cdot \mathbf{n} + \frac{2c_e}{\gamma - 1} \tag{2.20}$$

The subscript e denotes values determined from the extrapolated primitive variables (Section 4.7) and $\mathbf{n}$ denotes the boundary outward pointing normal unit vector. The sonic velocities for both compressible and incompressible flows are calculated as follows:

$$c_\infty = \eta\sqrt{(\gamma-1)T_\infty} + (1-\eta) \times max\left(\sqrt{a_r \mathbf{u}_\infty \cdot \mathbf{u}_\infty}, c_\tau\right) \tag{2.21}$$

$$c_e = \eta\sqrt{(\gamma-1)T_e} + (1-\eta)c_\infty \tag{2.22}$$

Note that the above sonic velocities in the incompressible case are calculated as shown in the interest of consistency with the artificial compressibility solution method. Further, in this work the artificial compressibility related parameter, a_r, is set to 1.2. Finally, γ in Equation (2.20) is calculated as

$$\gamma = \gamma_{air}\eta + \left(\frac{c_\infty^2}{T_\infty} + 1\right)(\eta - 1) \tag{2.23}$$

where in the case of incompressible flow $T_\infty = 1.0$.

From the above, the sonic velocity at the boundary is now calculated as follows

$$c_b = 0.25\eta(\gamma-1)(G_e - G_\infty) + (1-\eta)c_\infty \tag{2.24}$$

and the magnitude of the velocity normal to the boundary is found by

$$u_n = 0.5(G_e + G_\infty) \tag{2.25}$$

Using the normal velocity the velocity at the in- and outflow boundaries are calculated as:

$$\mathbf{u}_b = \mathbf{u}_\infty + (u_n - \mathbf{u}_\infty \cdot \mathbf{n}) \cdot \mathbf{n} \quad \text{and} \quad \mathbf{u}_b = \mathbf{u}_e + (u_n - \mathbf{u}_e \cdot \mathbf{n}) \cdot \mathbf{n} \tag{2.26}$$

respectively. The temperature at the in- and outflow boundaries follow from

$$T_b = T_\infty \left(\frac{c_b^2}{c_\infty^2} \right) \quad \text{and} \quad T_b = T_e \left(\frac{c_b^2}{c_e^2} \right) \tag{2.27}$$

respectively. Finally, the pressure at the boundary is determined by applying the ideal-gas constant-entropy relations at the inflow and outflow conditions respectively:

$$p_b = p_\infty \left(\frac{T_b}{T_\infty} \right)^{\frac{\gamma}{\gamma-1}} \quad \text{and} \quad p_b = p_e \left(\frac{T_b}{T_e} \right)^{\frac{\gamma}{\gamma-1}} \tag{2.28}$$

The above relations hold for both compressible and incompressible flow systems.

At the surface of the aerodynamic object, a symmetry or slip boundary condition is employed. Here the component of the velocity normal to the boundary is set to zero i.e.

$$\mathbf{u} \cdot \mathbf{n} = \mathbf{0} \tag{2.29}$$

while zero gradients in density and specific energy are applied as Neumann type boundary conditions:

$$\frac{\partial \rho}{\partial x_j} n_j = \frac{\partial E}{\partial x_j} n_j = 0 \tag{2.30}$$

where the nomenclature is as previously defined.

2.3 Conclusion

In this chapter the mathematical description of the physics being modelled is given. The governing equations are presented in a form appropriate for the solution of fully incompressible as well as fully compressible flows. The latter includes sub-, trans-, and supersonic systems. Closure to the aforementioned system is given by the definition of appropriate constitutive relations and boundary conditions.

Chapter 3

Mesh Generation

3.1 Introduction

The finite volume technique used in this work may be used to effectively solve systems of coupled non-linear partial differential equations over arbitrary spatial domains. It is however a prerequisite that the spatial domain be decomposed into a number of discrete non-overlapping regions. This process is commonly known as mesh-generation. A host of techniques have been devised by which to do this as summarised by Baker [16].

The requirements for CFD computations during the concept design phase poses strict limits on the choice of mesh generation technique. It is required that the overall time taken to generate the mesh be short, while being insensitive to surface definition inconsistencies and thus facilitate short total analysis times. The amount of user input in the mesh and geometry generation is the major factor in determining the speed at which these tasks are performed (which has been shown can take more time to generate the mesh than model the flow [4]). It is therefore required above all that the mesh be generated automatically and with speed around complex geometries with as little user input as possible. At present, this criterion leaves one with only two realistic choices of mesh generation strategy:

- Unstructured triangular in 2D or tetrahedral meshes in 3D.

- Cartesian meshes consisting of mainly quadrilaterals in 2D and hexahedra in 3D.

The unstructured meshes are generated by either the Delaunay triangulation or the advancing front method. They have been used extensively in solving problems of all types [17, 18, 19, 20]. Indeed most of the solution technologies used in this work were first conceived for use on this type of mesh. The main hindrance in generating such meshes is the need for a consistent or "clean" CAD geometry. The latter is essential for surface triangulation, and is often not naturally available during the initial phase of an aerodynamic design. As automatic tools to resolve inconsistencies in CAD geometry do not always work effectively, the desired water-tight geometry is only achieved by direct user intervention. The latter is a tedious and human resource intensive exercise.

Cartesian meshes do not suffer from the requirement of a consistent CAD geometry to the same degree as unstructured body-fitted meshes. They can further be generated with great efficiency on complex geometries. This is demonstrated by Aftosmis *et al.* [3] who shows that the complexity of the method is $O(N \log N)$ where N is the number of cells. Cartesian mesh generation is also known to be highly amenable to automation as has been reported by several authors [3, 10, 12, 21, 22] but do however have two main drawbacks, viz. the boundary of a Cartesian mesh is by definition a stair-step one[1], and the cells are refined as a rule in a two to one ratio. Related to the fixed refinement ratio is the formation of so called hanging nodes (also referred to as non-conforming nodes). These drawbacks can have unwanted accuracy implications if not appropriately dealt with.

The choice between the above two competing mesh generation strategies depends entirely on the intended application. For the solution of the Euler equations, anisotropic meshes are not required and in addition the quality of a Cartesian mesh is theoretically superior to that of a tetrahedral mesh of the same grid resolution in all areas except at the boundary and where hanging nodes are present [23]. This superior quality over a large part of the domain significantly reduces the computational complexity as shown by Aftosmis *et al.* [3]. Berger

[1] Also, and perhaps more commonly known as a Cartesian mesh with embedded geometry.

et al. [21] has shown that the higher mesh quality and local regularity can also improve the cache and parallel performance of the solver if written to take advantage of this feature. A further advantage of Cartesian mesh generation is the ease with which mesh refinement can be done. Adaptive mesh refinement is not implemented in this work but has been shown to work on these meshes by Popinet [24]. In the related area of moving body problems, Cartesian meshes have further been used with success by Yang *et al.* [25] and Murman *et al.* [10]. For these reasons it was decided that the Cartesian method is the superior option for the application under consideration.

This chapter will cover aspects related to the generation and preparation of the mesh in anticipation of the chosen discretisation strategy. The focus will be on the processes pertaining to furnishing the cut-cell mesh using a stair-step Cartesian mesh as a starting point.

3.2 Cartesian Mesh Generation

The focus of this work is not on the generation of Cartesian meshes and therefore commercial packages were used for this purpose. Harpoon [26] was the primary package employed, although Gambit [27] (from Fluent software) was also used in the early stages of the work. As both packages are unable to generate 2D Cartesian meshes, code was written to extracts the latter from a 3D mesh.

Cartesian meshes contain geometric planes that run through the mesh without intersecting any elements. These planes are made up of faces of 3D elements and are therefore similar to a 2D Cartesian mesh. It is one of these planes that is used to create the 2D mesh. Note that the planes only occur naturally throughout the domain on meshes where no 3D cell cutting has been performed. This is why the latter feature, which is available in Harpoon, was not used for the 2D cases and it was opted to develop it. This is detailed next.

3.3 Redefinition of the Boundary

Overlapping stair-step Cartesian meshes are not optimal for computational use as the accuracy of the solution on the boundary is compromised. They are thus altered to obtain a body conforming mesh. Two techniques are available for this. The technique used by most groups is the so called cut-cell approach (Murman *et al.* [11], Yang *et al.* [12], Aftosmis *et al.* [28], Coirier & Powell [8] and Ingram *et al.* [4]). Here, the cells that intersect the boundary are cut so that their boundary conforms to the local domain boundary. The other technique, as devised by Koh *et al.* [29], involves using a least squares mesh-less method to address the mesh boundary - geometry boundary interface. In this work the cut-cell approach was selected as the latter technique is not inherently conservative. Both 2- and 3D cases are considered. As noted previously, the mesh generator Harpoon is capable of furnishing a 3D cut-cell mesh which is why cut-cell technology only needed to be developed for 2D cases.

3.3.1 Cell Cutting Algorithm

The code developed to effect the cell-cutting requires as input an overlapping Cartesian mesh together with a representation of the geometry (see Figure 3.1). Detail on the representation of the geometry and supported geometry file formats is provided in Appendix B.2. It is desired that this cell-cutting algorithm robustly and automatically furnish the solver with a mesh that conforms to the geometry supplied, while maintaining a tolerance to "dirty" geometries.

The cell cutting algorithm commences by visiting all boundary nodes and checking the attached edges for an intersection with the geometry. If no intersection is found, the checked edges are deleted (as these are located inside the geometry) and the boundary redefined. If an intersection is found the edge is marked for cutting. The procedure is then repeated until all edges completely interior to the geometry have been deleted and all edges that intersect the boundary found (which is the case when at least one edge attached to each boundary node intersects the boundary). All boundary nodes are then deleted and the new boundary is defined by creating nodes at edge-geometry intersections and connecting these nodes with edges.

It is important to note that the above algorithm only works in isolation on meshes that completely overlap the boundary. The generation of the 2D mesh from a 3D one sometimes results in meshes where this is not so i.e. containing

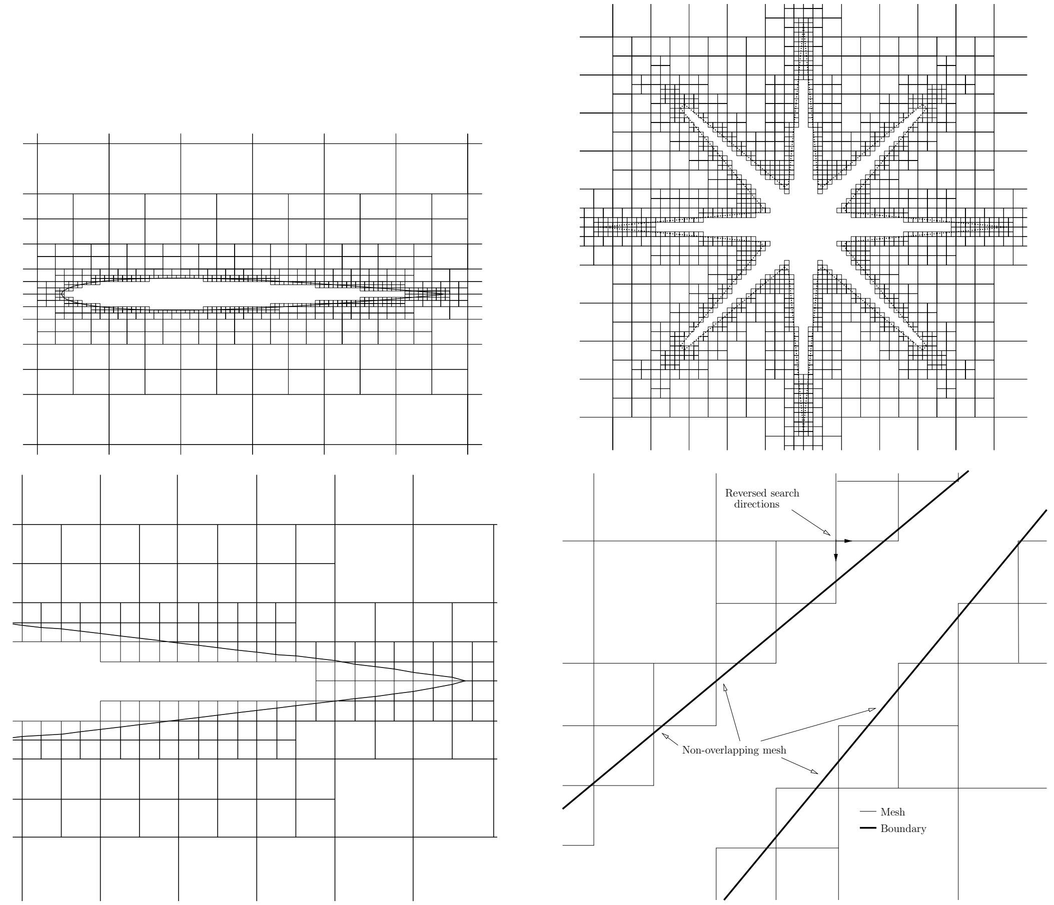

Figure 3.1: Examples of overlapping Cartesian mesh test-cases (with body geometry definition) employed for testing of the developed preprocessor (cell cutting and merging technology).

"voids". An example of this can be found in Figure 3.1 (bottom right).
To address this, code has been written to find these voids and fill them with triangles or quadrilaterals.

The above code has proven to be robust and generic in furnishing body-conforming meshes. No information on the type of boundary element nor what type of intersection the element has with the boundary is required. The algorithm has also proven to be insensitive to "dirty" geometries because, for the redefinition of an element, the intersections of only two edges with the boundary is all that is required. The chances that either of these edges penetrate the geometry through a geometric inconsistency is small.

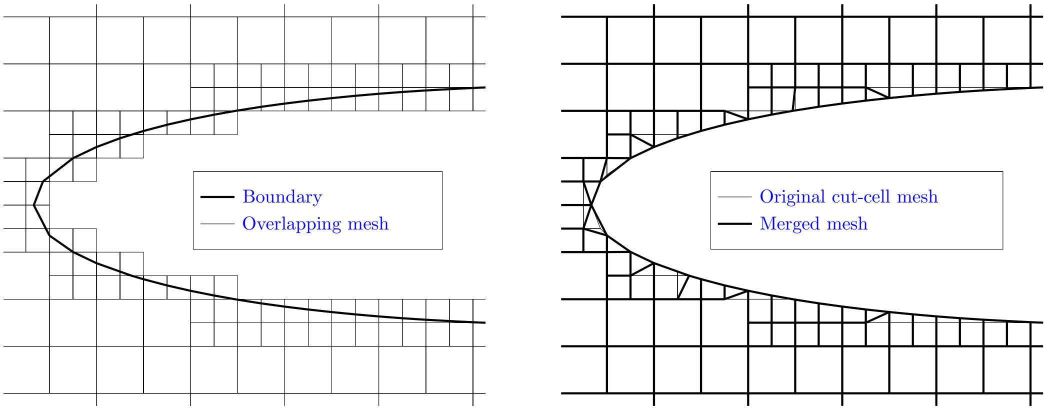

Figure 3.2: Shown is an example (left) of an overlapping Cartesian mesh generated with the Harpoon software and (right) the mesh resulting after applying cell-cutting (denoted "original cut-cell mesh") and cell-merging.

3.3.2 Cell Merging Operation

Pure cutting of the edges and filling of voids, as described above, may result in minuscule elements being created at the boundary. These elements are often many times smaller than adjacent elements, and can seriously impair the accuracy and stability of the solution scheme as well as result in long solution times. To deal with this, offending elements are merged into their neighbours.

Cell-merging commences by performing a loop over all boundary elements. If an edge attached to a particular element is smaller than the largest edge of that specific element by more than a certain factor (in this work a factor of 2 was found to yield acceptable results for the cases tested), the edge is collapsed. Of the two nodes attached to the edge to be collapsed, the node that does not lie on the boundary is omitted. If both nodes lie on the boundary, then either is selected for deletion. Finally, a second loop is performed over the boundary elements and those that contain less than three nodes removed. An example of the resulting cell-merged mesh is shown together with the original overlapping mesh in Figure 3.2 (right).

Note that the above algorithm does not, in most cases, explicitly collapse edges within the small elements themselves. This is as all the edges in these elements are often of similar lengths. Small elements are however destroyed from within their neighbours. This means that the programmer does not have to choose the neighbouring element in which to merge the small element. The aforementioned makes for a robust and generic cell-merging operation.

Once again this algorithm does not require any information about the type of element to be merged or the type of element that borders it. In this, it is completely generic.

3.4 Conclusion

This chapter was concerned with the chosen mesh generation method viz. cut-cell non-conforming Cartesian meshes. Technology developed to furnish 2D body-conforming meshes from 3D ones was documented. In the following chapters the resulting Cartesian meshes will be employed for the purpose of the discretisation and solution of the system of governing equations previously detailed.

Chapter 4

Spatial Discretisation

4.1 Introduction

The coupled non-linear partial differential equation set, which describes the flow under consideration, is not at present solvable via analytical means over complex domains. Their solution by numerical means is a viable alternative, with the methods of weighted residuals being well suited. Two popular techniques are the finite volume methods and the finite element methods. The former is currently the most popular scheme when solving the Euler equations on unstructured grids. In particular, the cell centred finite volume method is used almost exclusively in the context of cut-cell Cartesian meshes [4, 7, 8, 9, 10, 12, 13, 21, 22, 23, 24, 28, 30, 31, 32, 33]. This is attributed in part due to the cell-centred method being applicable in an unaltered state to non-conforming meshes (i.e. meshes containing hanging nodes) [23]. The work of French [34] is, to the authors knowledge, the only previous application of the vertex-centred method to Cartesian grids.

In this work, a vertex-centred edged based finite volume approach is employed, primarily to evaluate and promote such technology in the context of Cartesian meshes. This method is expected to be competitive with the cell centred variant as it offers similar accuracy vs. computational cost [14] while allowing Dirichlet boundary conditions to be enforced more directly (no extrapolation or ghost cell generation is required). It also works in an unaltered form at hanging nodes, but it is expected to reduce to first order accuracy here. For this reason, an alternative implementation is in addition investigated as part of this research.

The vertex-centred edged based finite volume method constitutes a notionally second-order accurate (central difference) type method, which is known to exhibit a tendency for odd-even decoupling of the solution. To suppress this tendency, stabilisation has to be added. Stabilising such schemes without the loss of second order accuracy is a major concern in the field of computational fluid dynamics [35]. This has led to the development of various stabilising methods by researchers such as Beam & Warming [36], Jameson *et al.* [1] and MacCormack & Baldwin [37]. One such class of scheme is termed *artificial viscosity* or *artificial dissipation*, and involves the addition of a biharmonic operator to the system of equations in the regions devoid of discontinuities. A harmonic operator (first order accurate upwinding) is to be added in the vicinity of shocks in order to render a stable non-physical oscillation-free method [35]. Another class of method employed to suppress non-physical oscillations is the so called *upwind* schemes (Godunov type / Riemann solver). In the context of Euler flow on unstructured Cartesian meshes, these schemes are the most popular [4, 8, 11, 13, 21, 23, 28, 30, 33] and the *artificial dissipation* methods do not at present feature. Due to the latter, as well as to in addition evaluate its applicability to Cartesian meshes, the Jameson, Schmidt & Turkel [1] variant of *artificial dissipation* will be employed in this work.

4.2 Finite Volume Discretisation of the Euler Equations

The governing equations presented in Chapter 2 (Equation (2.1)) may be written in non-dimensional integral form, on a three-dimensional Cartesian domain, by integration over an arbitrary volume $\mathcal{V} \in \mathbb{R}^3$ enclosed by bounding surface $\mathcal{A}$ as:

$$\int_{\mathcal{V}} \frac{\partial \mathbf{W}}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial t} d\mathcal{V} + \int_{\mathcal{A}} \mathbf{F}^j n_j d\mathcal{A} = \mathbf{0} \tag{4.1}$$

The cell-centred and the vertex-centred finite-volume methods differ in the manner in which the finite volume is constructed on a mesh. In the case of the former, the cells generated during mesh generation constitute the finite volumes, in each of which a node is placed at its geometric centre. The vertex-centred method differs in that computational nodes are placed at the vertexes, around which finite volumes are then constructed in order to furnish the so-called dual mesh.

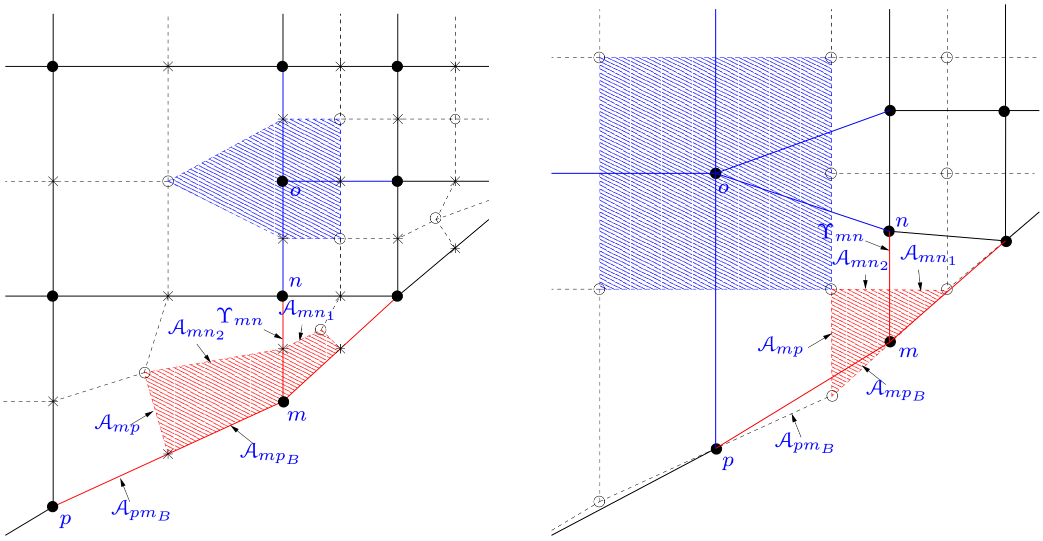

Figure 4.1: Schematic of the standard median-dual-mesh construction methodology (left) and the alternative scheme (right). Solid lines represent the computational edges and filled circles depict the nodes. The hatched region denotes the finite volumes $\mathcal{V}_m$ (red) and $\mathcal{V}_o$ (blue). Note that both dual-mesh constructions are derived from exactly the same mesh.

4.3 Dual Mesh Construction

As noted earlier, dual mesh construction is the process that produces the computational volumes employed by the vertex-centred scheme. In this work both the standard dual mesh construction of Vahdati *et al.* [38] as well as a new alternative method will be employed. The following sections describe each in detail.

4.3.1 Standard Dual Mesh Construction

This procedure will be discussed with reference to Figures 4.1 (left) and Figure 4.2. In 2D, dual-mesh bounding surface facets (lines) are constructed by connecting

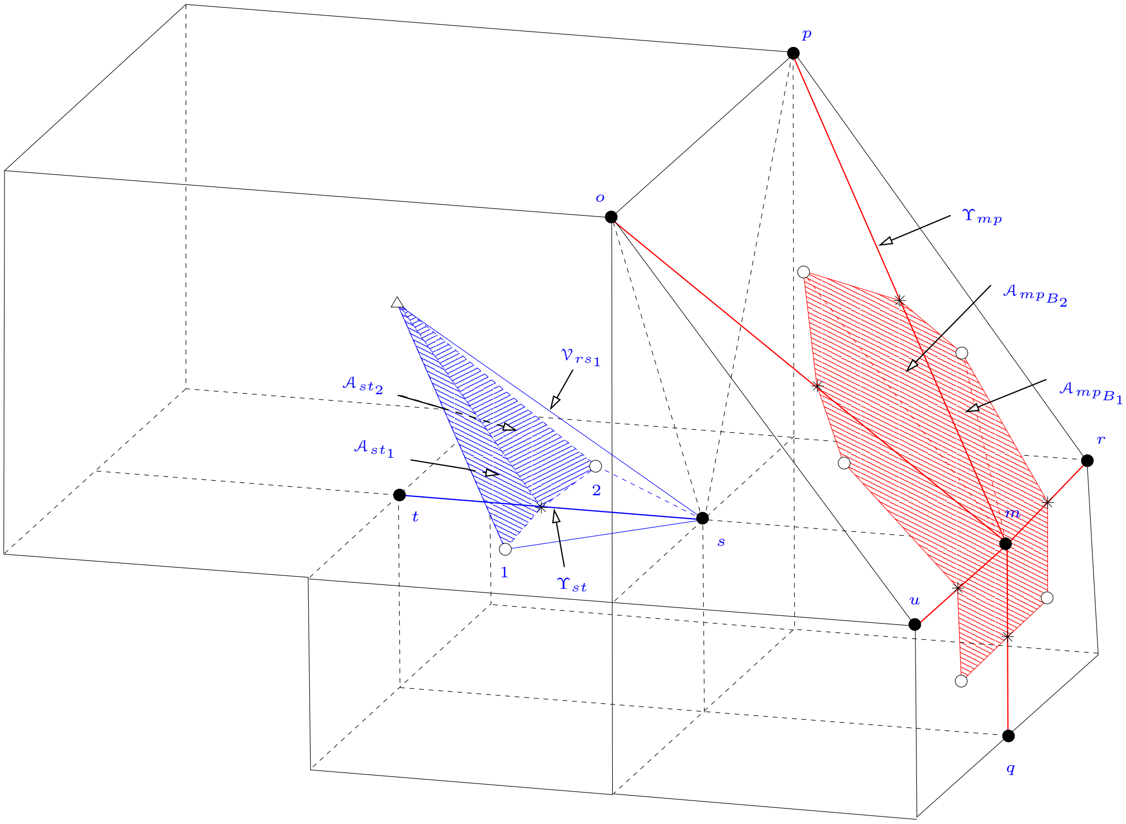

Figure 4.2: Schematic of the standard dual-mesh construction methodology in 3D. Black lines denote the inactive edges, while thick red lines are the active edges. The hatched regions are dual volume bounding surface facets.

edge centres (stars in Figures 4.1) with element centroids (open circles). These facets are then joined to form an enclosed volume around a node (filled circles) as shown schematically for a node o in the figure. The dual-mesh is constructed in 3D, by the definition of finite volume bounding facets formed by connecting edge centres (stars) with face-centroids (open circles) and element centroids (triangles in Figure 4.2). Two of the resulting facets for an edge s-t are shown in the figure as the blue hatched surfaces. Further, the completed 3D volume around node s is shown from various perspectives in Figure 4.3.

The above construction made use of surface and volume centroids. It is clear from the respective figures that these calculations are to be made on arbitrarily

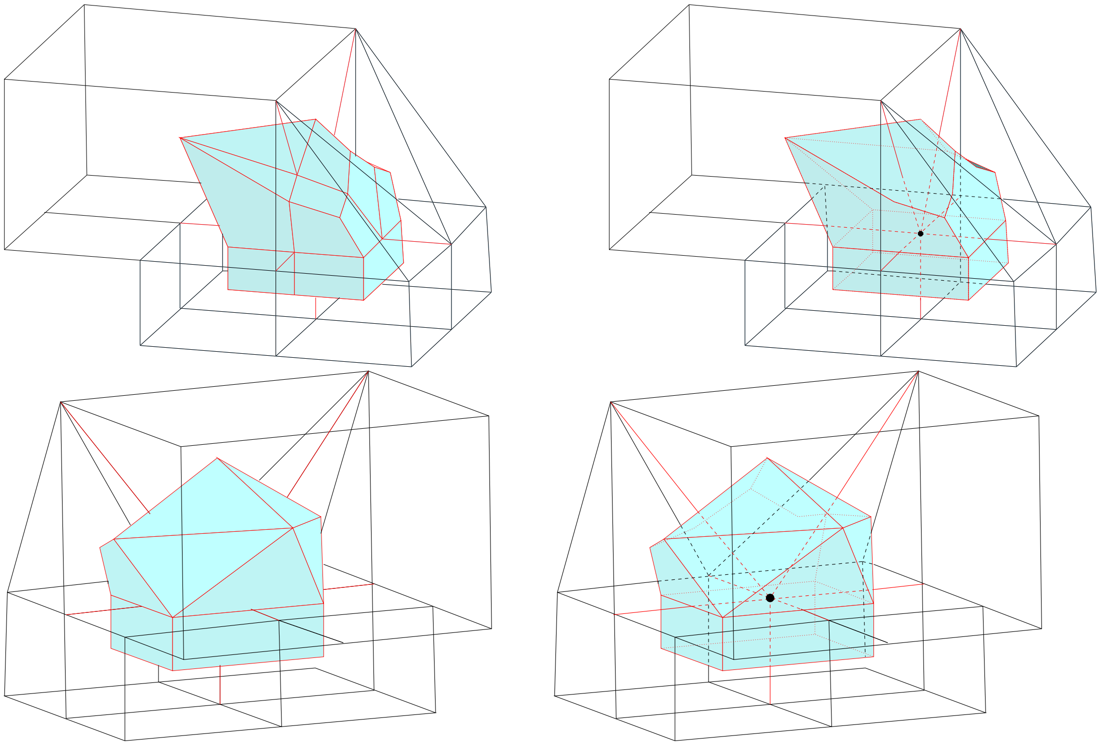

Figure 4.3: Two different views of the constructed dual-mesh volume for the node indicated by the black circle. The figures on the left do not show any hidden detail.

shaped elements and faces. These calculations are now discussed. A face centroid (in 3D) or an element centroid (2D) refers the area weighted centroid of a 2D element or a 3D element face. In order to ensure generic applicability, these are calculated by a two step process. First an estimate of the centroid is obtained by performing the vector average of all the nodal coordinates x_i on the face as:

$$\mathbf{x}^*_{est} = \frac{\sum_{i=1}^{N} \mathbf{x}_i}{N} \tag{4.2}$$

where N is the number face vertexes. By connecting all face vertexes to $\mathbf{x}^*_{est}$, a series of non-overlapping triangles are formed. The centroid of the face is then calculated by performing the area weighted average of the centroids of these triangles as:

$$\mathbf{x}^* = \frac{\sum_{i=1}^{N} \mathbf{x}_i^* \mathcal{A}_i}{\sum_{i=1}^{N} \mathcal{A}_i} \tag{4.3}$$

Here $\mathbf{x}^*$ is the centroid of the face and $\mathbf{x}_i^*$ denotes the centroid of triangle $\mathcal{A}_i$. The latter is calculated simply by the averaged vector sum of the triangles coordinates (which gives the exact centroid).

The above centroid calculation is valid for all convex faces and slightly concave ones. It is invalid however, if all the nodes of the face do not lie in a plane. These so-called "skew" faces are encountered in the 3D case as depicted in Figure 4.4. It was found that all such skew faces, generated by Harpoon, contain a maximum of four nodes while the "skewness" is limited to 40 degrees (user setting in Harpoon). The centroid of this face is therefore approximated by the mid-point of the line joining either of the opposing corners. To maintain the conservative nature of the discretisation scheme, the same set of opposite corners used to find the centroid must be used for this face whenever it is visited. Note that it is expected that skew faces containing exactly four nodes is not likely to be universal to Cartesian meshes, rather it is thought to be a product of the mesh generator and settings used in this work.

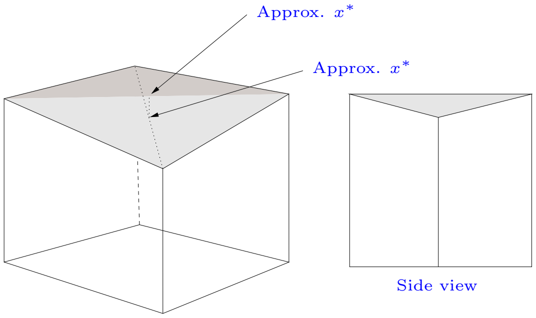

Figure 4.4: An example of a skew face. The two possible positions of the approximate centroid are indicated.

The calculation of element centroid in 3D (the volume centroid of an arbitrary 3D element), again involves a two step process. Firstly, an estimate of the centroid of the element is obtained by the sum of the nodal coordinates. The element is then divided into tetrahedra by triangulating each face of the element and joining the resulting vertexes to the aforementioned centroid (two such tetrahedra are shown in Figure 4.2).

The actual element centroid is then calculated as the volume weighted average of the centroids of the tetrahedra as:

$$\mathbf{x}^* = \frac{\sum_{i=1}^{N} \mathbf{x}_i^* \mathcal{V}_i^{tet}}{\sum_{i=1}^{N} \mathcal{V}_i^{tet}} \tag{4.4}$$

where x_i is the centroid of the tetrahedron i (calculated from the average vector sum of the vertexes) and $\mathcal{V}_i$ is the volume of the latter. This is calculated as:

$$\mathcal{V}_i^{tet} = \frac{1}{6} \begin{vmatrix} x_1^1 & x_2^1 & x_3^1 & 1 \\ x_1^2 & x_2^2 & x_3^2 & 1 \\ x_1^3 & x_2^3 & x_3^3 & 1 \\ x_1^4 & x_2^4 & x_3^4 & 1 \end{vmatrix} \tag{4.5}$$

where x_i^j refers to the coordinates of each node j in the tetrahedron. Now that the above geometric entities have been defined the construction of the dual mesh is continued.

The volume associated with a node m is designated $\mathcal{V}_m$ and the bounding surfaces by $\mathcal{A}_m$ and $\mathcal{A}_{m_B}$. The latter denote internal (dual) and computational boundary surfaces respectively. To compute fluxes between nodes, the faces that make up the boundary surface of the control volume have to be expressed mathematically. This is effected in an edge based manner through the definition of so called *edge coefficients*. An edge coefficient is defined as the area of the bounding surface of a computational volume associated with a particular edge multiplied by the unit normal vector of that face. This is expressed for an edge between nodes m and n as

$$\mathbf{C}_{mn} = \sum_{\mathcal{A}_{mn_q} \in \mathcal{A}_{mn}} \mathbf{n}^{mn_q} \mathcal{A}_{mn_q} \tag{4.6}$$

where $\mathcal{A}_{mn_q}$ refers to a facet of this surface and $\mathbf{n}_{mn_q}$ is the unit normal vector to $\mathcal{A}_{mn_q}$ (in this work the normal vector is chosen to point from the smaller to the larger node number). For the case shown in Figure 4.1, the edge coefficient is formed from two surfaces $q = 1$ and $q = 2$. For the case shown in Figure 4.2, the edge coefficient for the edge s-t is formed from four facets of which the two corresponding to $q = 1$ and $q = 2$ are shown.

The calculation on the boundary is similarly performed for an edge m-p:

$$\mathbf{B}_{mp} = \sum_{\mathcal{A}_{mp_{B^q}} \in \mathcal{A}_{mp_B}} \mathbf{n}^{mp^{B^q}} \mathcal{A}_{mp_{B^q}} \tag{4.7}$$

where $\mathbf{B}$ refers to a *boundary edge coefficient* and $\mathbf{n}^{mn^{B^q}}$ is the outward pointing normal vector to $\mathcal{A}_{mn_{B^q}}$. For the 2D case shown in the figure q is not required as the edge coefficient is made up of only one facet labelled $\mathcal{A}_{mp_B}$. For the 3D case shown in the figure for edge m-p, $q = [1, 2]$ as depicted.

As $\mathbf{C}_{mn} = -\mathbf{C}_{nm}$, only one internal edge coefficient need be stored per edge. This is where edge-based schemes are considerably more efficient than element based methods. With respect to the boundary edge coefficient, two values are stored at a node to ensure general applicability with regards to the dimension of the problem (In 3D $\mathbf{B}_{mp} \neq \mathbf{B}_{pm}$). The additional storage and computational cost is negligible as this is only needed for boundary edges.

4.3.2 Alternative Dual Mesh Construction

It is evident from the previous section that, at hanging nodes, the standard procedure results in dual-mesh volumes with geometric centres which are considerably removed from the node. This has serious accuracy implications. In an attempt to circumvent this, an alternative scheme was developed wherein nodes are placed in the centroids of the mesh elements, and the elements then constitute the dual-mesh or finite volumes. This is shown schematically in Figure 4.1 (right). The edges are then constructed between these nodes. The boundary volume is however to be given special treatment. As opposed to previously, a node is not placed in the centroid of the element, but rather on the boundary. More specifically, it is placed at the centre of the boundary edge (of the element). This enables the boundary conditions to be applied as per the standard variant of the vertex-centred scheme.

Having defined the dual mesh, edge coefficients are calculated as before. Equation (4.6) again applies to edge Υ_{mn} as does Equation (4.7) to edge Υ_{mp}. As in the standard scheme internal edge coefficients are stored once per edge as $\mathbf{C}_{mn} = -\mathbf{C}_{nm}$. This is not possible for the boundary edge coefficients as $\mathcal{A}_{mp_B} \neq \mathcal{A}_{pm_B}$ even in 2D. On this account, two boundary edge coefficients need to be stored. This however results in no alteration to the above methodology.

Importantly, the furnished dual-mesh fundamentally differs from that of the standard vertex-centred scheme in that volume bounding surfaces may no longer be assumed to intersect edges midway between nodes. In order to ensure spatial accuracy on the new dual-mesh, the standard edge-based vertex-centred discretisation scheme is to be accordingly generalised. This is detailed in Section 4.4.

The scope of this work was limited to the application of this alternative scheme to 2D only. The method is however not limited to 2D, and is expected to be easily extensible to 3D.

4.4 Discretisation of Convective Term

To benefit from the edge-based nature of the numerical procedure employed, the convective term $\frac{\partial \mathbf{F}^j}{\partial x_j}$ is discretised as

$$\int_{\mathcal{A}_m} F^{ij} n_j d\mathcal{A} \approx \sum_{\Upsilon_{mn} \cap \mathcal{V}_m} \overline{F}^{ij}_{mn} C^j_{mn} + \sum_{\Upsilon^B_{mn} \cap \mathcal{V}_m} F^{ij}_m B^j_{mn} \tag{4.8}$$

where Υ_{mn} denotes the edge connecting nodes m and n, and $\overline{\mathbf{F}}_{mn}$ is the averaged flux along the edge. The standard method employed to calculate the latter [14, 38, 39, 40] is

$$\overline{F}^{ij}_{mn} = \frac{1}{2}\left(F^{ij}_m + F^{ij}_n\right) \tag{4.9}$$

On a finite difference mesh the above is thought of as formally second-order accurate, which is however not the case [41]. In the case of the standard dual-mesh construction methodology, this deficiency may be rectified by employing the following relation:

$$\overline{\mathbf{F}}^{j}_{mn} = \begin{bmatrix} \overline{\rho}_{mn}\overline{u_j}_{mn} \\ \overline{\rho}_{mn}\overline{u_1}_{mn}\overline{u_j}_{mn} + \overline{p}_{mn}\delta_{1j} \\ \overline{\rho}_{mn}\overline{u_2}_{mn}\overline{u_j}_{mn} + \overline{p}_{mn}\delta_{2j} \\ \eta(\overline{\rho}_{mn}\overline{E}_{mn} + \overline{p}_{mn})\overline{u_j}_{mn} \end{bmatrix} \tag{4.10}$$

where the over-line quantities are calculated for a scalar field ϕ as

$$\overline{\phi}_{mn} = \frac{(\phi_m + \phi_n)}{2} \tag{4.11}$$

with ϕ_m and ϕ_n denoting the nodal (vertex) values.

It is implicitly implied in Equation (4.11) that the dual mesh bounding surface at which the flux is being calculated midway between the nodes. In the case of the proposed new dual-mesh construction procedure this is not always the case, and Equation (4.11) is to be generalised as:

$$\overline{\phi}_{mn} = (1 - r_{mn})\phi_m + r_{mn}\phi_n \tag{4.12}$$

where

$$r_{mn} = \frac{\sqrt{\sum_i^N (x_{m_i} - x_{\mathcal{A}_i})^2}}{\sqrt{\sum_i^N (x_{m_i} - x_{n_i})^2}} \tag{4.13}$$

Here $x_{\mathcal{A}_i}$ denotes the Cartesian co-ordinate i of the spatial position where the bounding surface of volume $\mathcal{V}_m$, namely $\mathcal{A}_m$, intersects edge $m - n$. Further, δ_{jj} is the Kronecker delta.

To ensure the overall computational efficiency of the numerical scheme, the above edge-based flux calculation procedure is employed in both forms shown viz. Equation (4.11) and (4.12). The latter, which is more costly to compute, is only used in the irregular parts of the cut-cell Cartesian mesh i.e. where $r_{mn} \neq \frac{1}{2}$.

4.5 Stabilisation: Artificial Dissipation

As noted in the introduction to this chapter, the equations will be stabilised without the loss of second-order accuracy by using the scalar valued dissipation

(JST) model developed by Jameson *et al.* [1]. This method, which does not yet feature in Cartesian mesh solvers, is suited to solving both compressible as well as incompressible flows via the proposed single equation methodology. Further, it has proven very effective when applied to complex flows [42] while offering a balance between accuracy and computational efficiency [43].

As per Sørensen *et al.* [40], artificial dissipation is implemented in conjunction with first-order accurate upwinding so as to avoid non-physical spurious oscillations across discontinuities in field variables i.e. across supersonic shocks. The resulting stabilising term to be added to the right-hand-side of the discretised governing equation is as follows

$$\mathbf{D}_m = \eta \mathbf{D}_m^{sc} + \mathbf{D}_m^{JST} \tag{4.14}$$

where $\mathbf{D}_m^{sc}$ and $\mathbf{D}_m^{JST}$ denote the shock-capturing and artificial dissipation terms respectively. The shock-capturing or upwinding term is detailed first. It may be calculated at each node via a scaled approximation of a Laplacian operator similar to Mavriplis [18]:

$$\mathbf{D}_m^{sc} = \sum_{\Upsilon_{mn} \cap \mathcal{V}_m} \varepsilon_2 \overline{\Lambda}_{mn} P_{mn} \left(\mathbf{W}_n - \mathbf{W}_m \right) \tag{4.15}$$

where ε_2 is an empirical constant, the value of which is to be determined for each problem through numerical experimentation (the objective is to use the smallest possible value to stabilise and eliminate spurious oscillations). Further, $\overline{\Lambda}_{mn}$ is the edge-based interpolated scaling factor and P_{mn} is the pressure switch. The former scaling factor is calculated in an edge-wise manner at each node as

$$\Lambda_m = \sum_{\Upsilon_{mn} \cap \mathcal{V}_m} \left| \overline{\mathbf{u}}_{mn} \cdot \mathbf{C}_{mn} \right| + \overline{c}_{mn} \left| \mathbf{C}_{mn} \right| \tag{4.16}$$

where $|\mathbf{C}_{mn}|$ denotes the Euclidean norm of $\mathbf{C}_{mn}$ and c is the non-dimensional acoustic velocity. In the case of the standard edge-based finite volume method, all edge-averaged quantities (over-line) are calculated as the average of the two nodal values. For the new alternative dual-mesh construction strategy with associated proposed edge-based flux averaging procedure, the over-line quantities are however to be calculated via Equation (4.12). This is in the interest of consistency.

The pressure switch is tasked with sensing large pressure gradients, and scaling the above upwinding term accordingly. It is calculated in a similar way to Mavriplis [18] for an edge $m - n$ as follows

$$P_{mn} = max\left[|\Delta p_m|, |\Delta p_n|\right] \tag{4.17}$$

where

$$\Delta p_m = \sqrt{12}\frac{\sum_{\Upsilon_{mn}\cap\mathcal{V}_m}(p_m - p_n)}{\sum_{\Upsilon_{mn}\cap\mathcal{V}_m}(p_m + p_n)} \tag{4.18}$$

The above term is large across a shock and tends to zero in smooth fields. The result is that the numerical scheme is reduced to first-order accuracy in the vicinity of shocks while the above upwinding term tends to zero away from shocks. In these regions the JST stabilisation term is employed to effect stability and ensure high resolution accuracy (notionally second-order accurate).

The JST stabilising term is constructed on a cut-cell Cartesian grid through the use of a biharmonic operator as proposed by Mavriplis [18]. This involves conducting two loops over edges, where the first loop entails the construction of a harmonic operator. In the interest of computational efficiency, this is approximated for both compressible and incompressible flows as follows:

$$\nabla^2\mathbf{W}_m \approx (1-\eta)\sum_{\Upsilon_{mn}\cap\mathcal{V}_m}\left.\frac{\partial\mathbf{W}}{\partial\mathbf{Q}}\right|_{mn}(\mathbf{Q}_n - \mathbf{Q}_m) + \eta\sum_{\Upsilon_{mn}\cap\mathcal{V}_m}(\mathbf{W}_n - \mathbf{W}_m) \tag{4.19}$$

where summation is performed over all edges connected to node m, and $\mathbf{W}$ and $\mathbf{Q}$ denote the dependent variable vectors. The preconditioned relation employed above for incompressible flow, is due to Malan *et al.* [14].

The biharmonic operator is constructed by essentially repeating the above procedure as follows

$$\mathbf{D}_m^{JST} = \sum_{\Upsilon_{mn}\cap\mathcal{V}_m} -max\left[0, (\varepsilon_4 - \eta\varepsilon_2 P_{mn})\right]\overline{\Lambda}_{mn}\left(\nabla^2\mathbf{W}_n - \nabla^2\mathbf{W}_m\right) \tag{4.20}$$

where ε_4 is an empirical constant to be determined similar to ε_2.

The calculation of $\mathbf{D}_m^{JST}$ is known to be problematic on a boundary. The procedure used in this work to calculate this term is that proposed by Mavriplis [18]. The contribution from an internal edge mn (see Figure 4.5) in Equation (4.19), is approximated by replacing $\mathbf{w}_n - \mathbf{w}_m$ with $\mathbf{w}_{n'} - \mathbf{w}_m$. Here $\mathbf{w}_{n'}$ is the projection of the value of $\mathbf{w}_n$ on to the boundary edge. If a normal gradient of zero is assumed, it can be shown that the contribution from edge mn in Equation (4.19) is equal to $C(\mathbf{w}_o - \mathbf{w}_m)$, where C is a constant. Note that the assumption of a zero normal gradient here is consistent with the boundary condition applied at slip boundaries. The calculation of the biharmonic operator (Equation (4.20)) is unaffected.

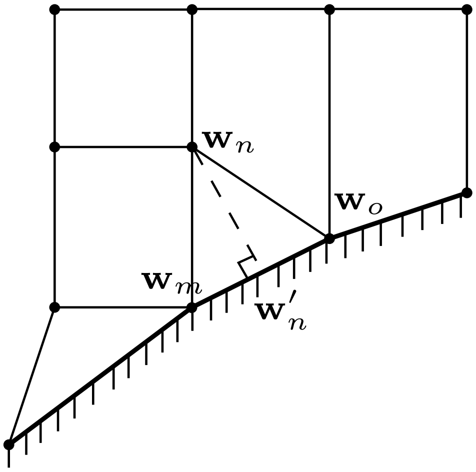

Figure 4.5: Calculation of artificial dissipation term on the boundary

4.6 Semi-Discrete Equation

The semi-discrete form of the governing equations (Equation (2.1)), that results from the vertex-centred discretisation process described above now follows:

$$
\begin{aligned}
\frac{\partial \mathbf{W}}{\partial \mathbf{Q}}\frac{\partial \mathbf{Q}}{\partial t} \approx \sum_{\Upsilon_{mn}\cap\mathcal{V}_m} \overline{F}^{ij}_{mn} C^{j}_{mn} + \sum_{\Upsilon^{B}_{mn}\cap\mathcal{V}_m} F^{ij}_{m} B^{j}_{mn} \\
+\eta \sum_{\Upsilon_{mn}\cap\mathcal{V}_m} \varepsilon_2 \overline{\Lambda}_{mn} P_{mn} \left(\mathbf{W}_n - \mathbf{W}_m\right) + \\
\sum_{\Upsilon_{mn}\cap\mathcal{V}_m} -max\left[0, \left(\varepsilon_4 - \eta\varepsilon_2 P_{mn}\right)\right] \overline{\Lambda}_{mn} \left(\nabla^2\mathbf{W}_n - \nabla^2\mathbf{W}_m\right)
\end{aligned}
\tag{4.21}
$$

where the second and third terms on the right-hand-side are the convective part of the equations. The last two terms are the stabilisation terms. The discretisation of the temporal term on the left-hand-side is detailed in the next chapter.

4.7 Boundary Conditions

This section details the numerical implementation of the characteristic boundary conditions described in Section 2.2.3. These characteristic far-field boundary conditions require that an extrapolation be made from the interior of the domain to the boundary. Unfortunately this is not as simple as it seems in the case of a supersonic outflow. Here it is necessary that no reflection of downstream boundary effects is propagated back into the domain. Two methods of extrapolation were investigated to ensure this viz. the method of linear least squares and the method of inverse distance weighting of Watson [44] (see Appendix C for a detailed explanation). Both of the aforementioned methods require a minimum of three non co-linear points in order to get a meaningful approximation. It has been found that the selection of these points is crucial to ensure minimum reflection.

Figure 4.6 depicts a typical far field boundary on a Cartesian mesh. The algorithm to find the points to be used in the extrapolation proceeds as follows. Using stored local connectivity information around the boundary node of interest (coloured red in the figure), the nearest internal node is found. In the figure it is coloured blue. Again using local connectivity information all internal nodes attached to this node are found (shown in green). These nodes, together with the closest node, constitute the body of nodes from which information is extracted for the purposes of extrapolation. If the boundary is a subsonic outflow, all the

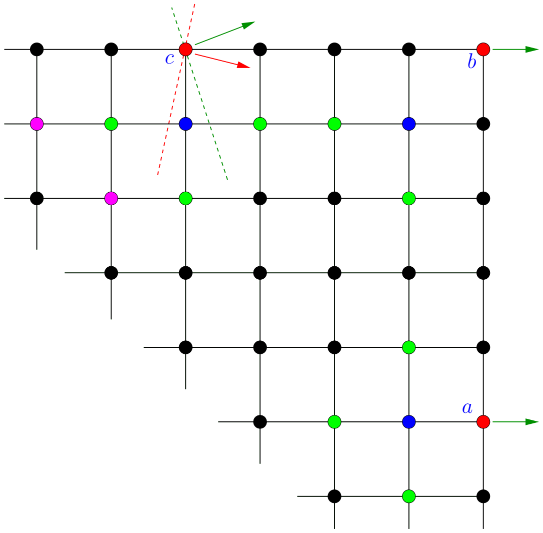

Figure 4.6: Selection of points for boundary condition extrapolation.

nodes are employed. To prevent reflection of the boundary information in the case of a supersonic outflow, only nodes that are upstream of the current node are used in the approximation.

To identify upstream nodes for the supersonic case, the local velocity vector on the boundary node itself is used. As an example of this we consider node c in Figure 4.6. If the flow is parallel to the green arrow, then any node to the left of the constructed perpendicular line (green dashed line) may be used (of which there are three in this case). If however the local velocity at node c is in the direction indicated by the red arrow, one can see that only one of the identified nodes can be selected. As this solitary node is insufficient for extrapolation purposes, its local connectivity is queried to find more upstream nodes. These nodes are coloured pink. To alleviate difficulties in finding sufficient numbers of nodes in close proximity to the boundary node, downstream nodes that were less than 3° from the perpendicular (dashed) line were also used in this work. Note that boundary nodes are naturally never selected.

Both of the two chosen extrapolation methods were tested in this work. The method of inverse distance weighting is computationally less expensive than the least squares method and was found to work better for all the test cases presented.

It was thus used as the method of preference.

4.8 Conclusion

In this section the procedure employed to discretise the governing equations on Cartesian meshes is presented. To summarise, the vertex-centred finite volume method is employed with two different definitions of the dual mesh viz. the standard method of Vahdati *et al.* [38] and a newly developed alternative. To account for the latter, the discretisation scheme is accordingly enhanced. A variant of the Jameson, Schmidt & Turkel [1] artificial dissipation scheme is used for stabilisation purposes. Finally, the implementation of the characteristic far-field boundary conditions is presented. In the following chapter, the procedure used to solve the resulting semi-discrete equation set is discussed.

Chapter 5

Temporal Discretisation and Solution Procedure

5.1 Introduction

The previous chapters described the set of spatially discretised governing equations used in this work. What remains is to discretise the temporal term ($\frac{\partial \mathbf{Q}}{\partial t}$ in Equation (2.1)), followed by the simultaneous solution of the resulting set of discrete equations. This is the subject of this chapter.

The two main temporal discretisation strategies which may be employed for the work under consideration are explicit and implicit. Implicit methods can be loosely defined as those methods that may require matrix inversions to be performed, while explicit methods naturally do not [20]. The main advantage of the basic explicit method is that the memory cost is $O(N)$, where N denotes the number of unknowns to be solved for. The disadvantage is that there is a severe time-step size restriction (Courant-Friedrichs-Lewy number), which results in the need for a large number of time-steps (iterations) in order to reach the steady-state (converged) solution. The basic implicit method does not pose the same time-step restriction, but requires the setting up of a matrix, the storage cost of which may be $O(N^2)$. Further, the computational cost of iterative matrix-inverters is $O(N^2)$.

Due to the disadvantages of the basic explicit and implicit solution strategies, they are rarely used to model flow on unstructured Cartesian meshes. To circumvent the aforementioned disadvantages, more advanced methods have been developed, and a summary of these in the context of Cartesian meshes now follows.

An implicit-time integration scheme currently in use on unstructured Cartesian meshes, which solves a matrix without storing it, is the so-called Lower-upper Symmetric Gauss-Seidel (LU-SGS) as implemented by Ogawa [22]. In the cited work this method is used in conjunction with a variant of the Full Approximation Storage (FAS) multigrid method in order to ensure acceptable computational cost (solution times) in addition to low storage. Another implicit method employed in recent relevant research is a block-lower-upper symmetric Gauss-Seidel method used by Wang & Chen [23] and Zhang & Wang [33]. The LU-SGS (BLU-SGS) method requires less memory than the fully implicit scheme, while involving a comparable amount of iterations. Finally, a semi-implicit method is used by Ham *et al.* [9]. This involves decomposing the system of discrete equations into a number of one-dimensional problems that can be solved directly by inverting a set of tridiagonal matrices.

The majority of authors using Cartesian meshes use explicit schemes. These vary from multi-stage time-stepping (Coirier & Powell [8]), to employing Runge-Kutta relaxation in conjunction with FAS-multigrid solution acceleration (Aftosmis *et al.* [28] and Murman *et al.* [45]. The memory cost of these techniques is optimal ($O(N)$), while computational cost tends toward $O(N)$. The split type (functional step) method developed by Popinet [24] (developer of GERRIS), employs a point-relaxation scheme (mathematically analogous to explicit time-stepping), which is again accelerated via FAS-multigrid.

It is clear from the above that, in the context of unstructured Cartesian mesh based solvers, both implicit and explicit type temporal discretisation methods are currently in use. Implicit methods range from block-implicit to matrix-free LU-SGS used in conjunction with multigrid solution acceleration. The drawback of the aforementioned methods is that the memory cost still far exceeds that of explicit type schemes. Further, implicit methods require linearisation due to the non-linear nature of the system of PDEs being solved, which notably adds to computational cost. Explicit methods which are employed in conjunction with Runge-Kutta relaxation and FAS- multigrid, are on the other hand found to be very promising. The required memory and solution cost is exceptional while no linearisation is required. However, the cited explicit multigrid methods employ a complex unstructured Cartesian-mesh data structure with which to generate the course meshes. This data structure also limits the applicability of the solver to Cartesian meshes.

Due to the above, a multigrid acclerated explicit temporal discretisation was selected for the work under consideration. This ensures optimal memory efficiency, such that large meshes, which may contain millions of elements, may be solved on a standard desktop personal computer. Next, Runge-Kutta time-stepping is employed in conjunction with FAS-multigrid to effect fast solution times, which is again required when dealing with large meshes. Further, as the discretisation scheme discussed in Chapter 4 is generically applicable to any mesh type, it is undesirable and unnecessary to employ a Cartesian mesh specific data structure for multigrid accleration. Rather a more generic FAS agglomerated multigrid technique similar to that recently used by Sørensen *et al.* [40] (which was applied to hybrid unstructured body-fitted meshes) will be employed.

5.2 Jacobi Iterative Scheme

As noted previously, an explicit multi-stage Runge-Kutta temporal discretisation is employed. This is to ensure the matrix free nature of the system and hence optimal memory efficiency. The solution is advanced from time t to $t + \Delta t$ via a four stage Runge-Kutta method designed for application to partial differential equations (Tannehill *et al.* [46]) as follows:

$$\begin{aligned}
\mathbf{Q}^{0}_{fm} &= \mathbf{Q}^{t}_{m} \\
\mathbf{Q}^{k}_{fm} &= \mathbf{Q}^{0}_{fm} + \alpha_k \frac{\partial \mathbf{W}}{\partial \mathbf{Q}}^{-1} \Delta t^{k-1} (\mathbf{RHS}^{k-1}_{m}) \text{ for } k = 1 \text{ to } 3 \qquad (5.1) \\
\mathbf{Q}^{t_{f_\tau}+\Delta t_\tau}_{m} &= \mathbf{Q}^{4}_{fm}
\end{aligned}$$

where the coefficients α_k are taken as 0.11, 0.2766, 0.5 and 1 as per Lallemand *et al.* [6]. **RHS** constitutes all terms to the right of the equal sign in Equation (4.21). To save computational time, local time stepping is employed in addition to the stabilising terms (upwinding and artificial dissipation) being calculated only at the first and third stages [41].

The maximum allowable time-step size is calculated at a node m such that stability is ensured as:

$$\Delta t_{\tau_m} = \frac{CFL\ \Delta\chi_m}{|\mathbf{u}_m| + c_m} \tag{5.2}$$

where CFL is the Courant-Friedrichs-Lewy number (typically set to between 1.5 and 2) and $\Delta\chi_m$ is a measure of the spatial cell / element size. The latter is calculated as

$$\Delta\chi_m = \sqrt{\left(\frac{1}{\Delta x_j \Delta x_j}\right)^{-1}}\,\Bigg|_m \tag{5.3}$$

where Δx_j is the smallest distance in the x_j direction to an adjacent node. This time-step calculation procedure is similar to that used in finite difference electromagnetic wave propagation problems [47] and analogous to the effective element size used in explicit finite element computational fluid dynamics calculations (Zienkiewicz & Taylor [48]). The methodology may be implemented at a node m in an edge-based manner as follows

$$\Delta t^m = min\left\{\Upsilon_{mn} \cap \mathcal{V}_m \left[\frac{CFL\ \Delta\chi_m}{|\overline{\mathbf{u}}_{mn}| + \overline{c}_{mn}}\right]\right\} \tag{5.4}$$

where the over-line quantities denote edge averaged quantities.

5.3 FAS Multigrid

Multigrid is a solution acceleration algorithm that has the potential of reducing the number of floating point operations required to simultaneously solve an $N \times N$ system of equations, in a matrix free manner, from $O(N^3)$ when the Jacobi iterative method is used (which is the current basic method discussed in the previous section), to $O(N)$. This implies a significant potential reduction in computational times. An explanation of the basic multigrid methodology is detailed by Brandt [5], and more recently by Briggs *et al.* [49] and Sørensen [20].

The underlying concept is relatively simple. When using an iterative scheme like Jacobi or Gauss-Seidel, only high frequency errors are eliminated at each iteration. This is because the nodes are only strongly connected with their direct neighbours and it takes time for information to propagate through the mesh. In the multigrid procedure, successive levels of coarse grids[1] are generated and information is passed between these grids allowing information to pass between nodes that are far apart from each other relatively quickly. This means that low frequency errors are eliminated more rapidly which enhances the overall solution scheme convergence rate.

There are a number of different types of non-linear multigrid. They essentially differ in the method used to generate the coarse grids. Algebraic multigrid uses a mathematical operator that gives the effect of the coarse grids. The geometric multigrid methods use the actual geometry and generate new meshes for each coarse mesh level. Agglomerated multigrid methods are a mix between the aforementioned two methods. Here, coarse meshes are generated by agglomerating computational cells followed by the appropriate redefinition of volumes and edge coefficients. In this work, the agglomerated multigrid method was selected as it is possible to automate the agglomeration process and it is generally applicable to any type of mesh.

5.3.1 Mesh Agglomeration Procedure

The strategy employed to generate the agglomerated or coarse mesh levels depends on the type of mesh that is used. Structured finite-difference type meshes lead very naturally to coarsening viz. by simply removing major grid lines. Complex unstructured meshes need to be dealt with in a more sophisticated geometric manner. Cut-cell Cartesian meshes fall somewhere in between. These grids contain structured type regions but also decidedly unstructured areas. Examples of the latter are boundaries and areas where hanging nodes are present.

The approaches prevalent in the literature when dealing with cut-cell unstructured Cartesian meshes is essentially of the structured type [9, 28, 50]. This

[1] Note that the term "coarse grids" will be used to refer to both the geometric multigrid form and its matrix-based algebraic multigrid equivalent.

method makes use of the hierarchical nature of unstructured Cartesian meshes. There is an obvious parent-child relationship in these meshes in which a parent cell can consist of up to four children (conversely, four small quadrilateral elements can be viewed as constituting one large quadrilateral element). The agglomeration strategy involves identifying children cells and agglomerating them into their parent cell. The child cells may be identified via either a tree based data structure (such as that used in the generation of the mesh) or by an unstructured approach using space filling curves (as per Aftosmis *et al.* [28]). Examples of agglomeration strategies using a tree based data structure can be found in the work of Charlton [50] and Ham *et al.* [9].

An alternative coarse mesh generation strategy, which currently does not feature in the context of cut-cell unstructured Cartesian meshes, is the completely unstructured type. This approach does not rely on the structured nature of the mesh at all, and is therefore in principle, applicable to any type of mesh. The method hinges on employing the local connectivity of the mesh, whereby nodes that surround a so called super-node are agglomerated into one another. This scheme has been successfully applied to body-conforming unstructured-hybrid meshes in two and three dimensions by a number of researchers viz. Lallemand *et al.* [6], Sørensen [20], Hannemann [51], Mavriplis & Venkatakrishnan [52]. Due to its generic applicability, it not requiring a specific data structure, as well as it being novel in the context of Cartesian meshes, the unstructured mesh agglomeration method will be used in this work.

5.3.2 Implemented Agglomeration Strategy

The selected agglomeration procedure (Hannemann [51]) commences by selecting a so called *super-node* from the fine cut-cell mesh. A super-node is selected according to one of the following four user-selected strategies:

- The “simple” strategy. This is the most simple of the strategies, where in any instance, the first possible node is selected. With this strategy the first super node would be the first node in the node table.

- The "closest node" strategy. In this strategy the selected node will always be the one in the list that is closest to a user defined spatial point.

- The "furtherest node" strategy. The chosen node will always be the furtherest node from a user specified spatial point.

- The "finite difference" strategy. In this strategy a node numbering scheme is assumed such that the resulting coarse mesh would still be a finite difference mesh if the fine mesh was one. It was developed for debugging purposes but has been found to work on the Cartesian meshes under consideration.

Once a super-node has been found, its neighbours are fused into it (by means of combining dual mesh volumes) in the following manner: First a loop is performed over all the attached edges to identify the so called direct neighbours (see Figure 5.1). The direct neighbours that have not already been fused with a super-node, are fused to the super-node being dealt with. If there are four or less direct neighbours attached to a specific super-node, a loop is in addition performed over the edges attached to all non-fused direct neighbours, and indirect neighbours identified. An indirect neighbour is a node that is connected to two direct neighbours of a specific current super-node, as depicted in Figure 5.1. Identified indirect neighbour-nodes are then fused to the super-node if they have not already been fused. Note that indirect neighbours are only fused if there are already direct neighbours in the fused set.

Once the above procedure has been completed for a specific super-node, a *front* of nodes is defined. This consists of all direct neighbours of the nodes that have just been fused to the super-node under consideration, with the added criteria that they have not already been fused to a super-node. From this front, a new super-node is chosen according to the following criteria (listed in order of descending preference):

1. It is a boundary node on a corner in the domain e.g. trailing edge or corner of the mesh.

2. It is a boundary node on the geometry.

3. It is a far-field boundary node

4. The node is chosen using one of the user selected strategies described previously

In the case where a specific super-node does not contain a front (all its neighbours have already been fused), the node table is traversed and a super-node selected based on the user-selected strategy. This procedure continues until all nodes in a specific mesh are either super-nodes or have been fused to one.

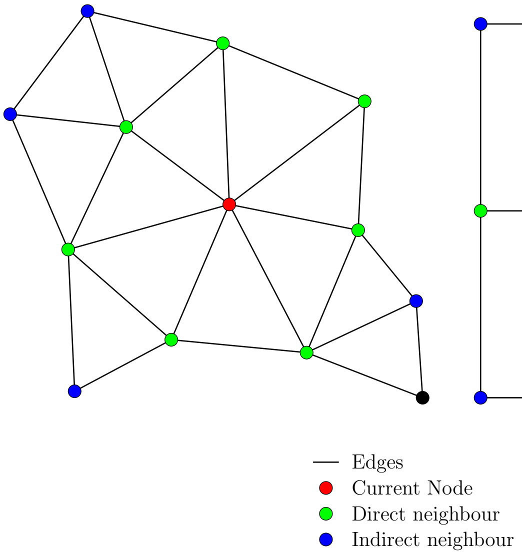

Figure 5.1: Direct and indirect neighbours as identified via edge-connectivity.

Once the above procedure has been completed, a loop is performed over all super-nodes and those that are not agglomerated to at least one other node are merged to a neighbouring super-node. This ensures that there is not an overly large disparity between adjacent volume sizes in the new mesh.

On completion of the above procedure, coarse mesh edges are constructed by connecting adjacent super-nodes. Adjacent super-nodes are defined as nodes that share an agglomerated super volume boundary. These shared volume boundaries are identified by finding fine mesh edges that join nodes which are fused into neighbouring super-node groups. The portion of the agglomerated or coarse mesh volume bounding surface between two neighbouring super-nodes is mathematically expressed as the sum of the edge coefficients of the overlapping fine mesh edges. This can be expressed in mathematical form for the edge coefficient associated with the coarse mesh edge, Υ_{mn}^{2h} between super-nodes m and n as

$$\mathbf{C}_{mn}^{2h} = \sum_{\Upsilon_{mn}^{h} \cap \mathcal{A}_{mn}^{2h}} \mathbf{C}_{mn}^{h} \tag{5.5}$$

where Υ_{mn}^{h} refers to the fine mesh edges that overlap the boundary between the super-node volumes denoted $\mathcal{A}_{mn}^{2h}$. Edge coefficients of fine mesh edges that do not overlap coarse mesh boundaries do not form part of the new mesh and are ignored.

Boundary edge coefficients, which express the boundary of the domain, are never ignored and all are added to their respective coarse mesh edges[1]

The above edge coefficient calculations apply to both discretisation schemes. In the case of the alternative scheme, it remains to define the ratio r_{mn} (Equation 4.13) on the coarse meshes. This is approximated using the following relation:

$$r_{mn}^{2h} = \frac{\sum_{\Upsilon_{mn}^{2h} \cap \mathcal{V}_{m}^{2h}} r_{mn}^{h} |\mathbf{C}_{mn}^{h}|}{\sum_{\Upsilon_{mn}^{2h} \cap \mathcal{V}_{m}^{2h}} |\mathbf{C}_{mn}^{h}|} \tag{5.6}$$

where h and $2h$ refer to fine and coarse mesh entities respectively.

An example of the agglomeration process on a simple Cartesian type grid is depicted in Figure 5.2. Shown in the top left figure is the red node that has been selected as a super-node. Its direct and indirect neighbours which will be fused into it are coloured green. In the background of this, the dual mesh is indicated with the red dashed lines. Some of the volumes have been greyed out to help identify them. In the top right figure the nodal front is constructed around the fused nodes (coloured blue). In the bottom left figure the new agglomerated coarse volumes are depicted. Here the fine mesh edges are divided into two groups viz. the dashed edges which do not cross the new volume boundaries, and the solid edges that do. Remember that only the edges that cross over coarse mesh volume boundaries contribute their edge coefficients to the new coarse edge coefficients. The final figure (bottom right) depicts the agglomerated (coarse) dual-mesh shown.

[1] Note that to determine which edges these are, it is required that one determine to which side of the super-node the fine mesh edge lies and to which adjacent super-node it is closest to.

Having completed the agglomeration exercise, a check is performed on the new coarse mesh to ensure its integrity. This involves the summing of edge coefficients around super-nodes. These should sum to zero and this sum gives a good indication of whether there are holes in the mesh or edges that are missing.

The above algorithm has been successfully implemented and is completely automatic and robust. It can, in principle, be applied to any conceivable mesh as it operates only on the data structure of the fine mesh.

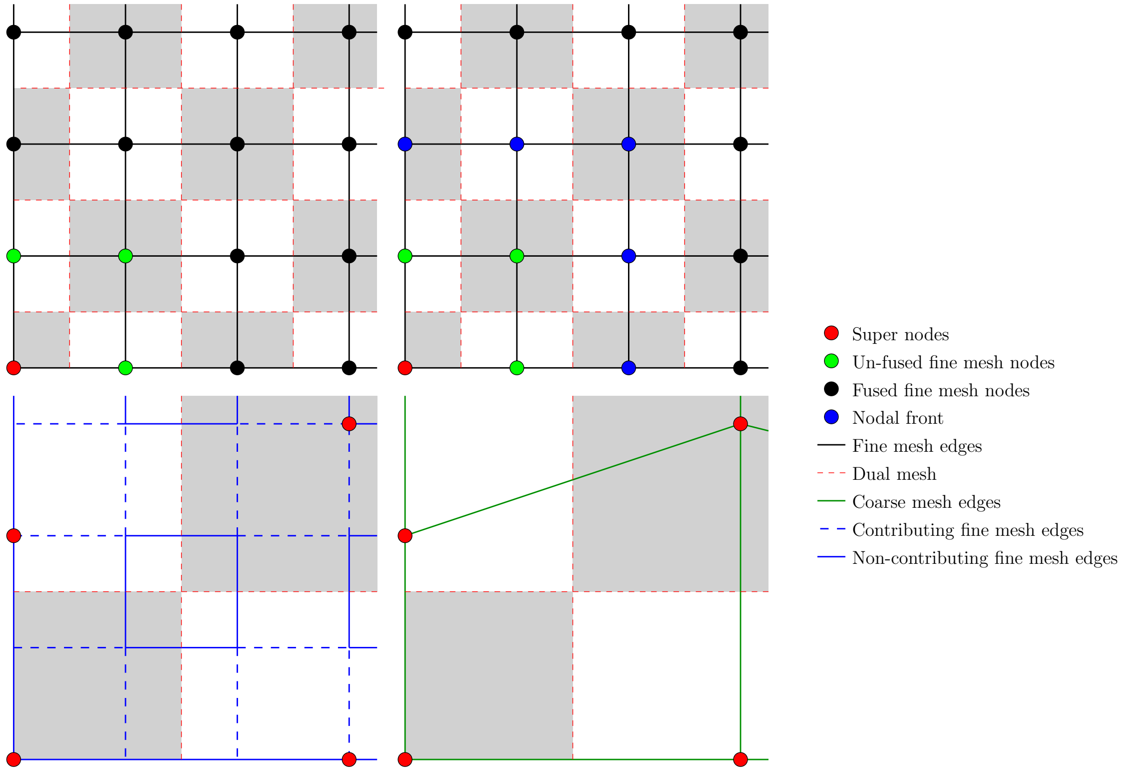

Figure 5.2: The implemented course mesh generation strategy.

5.4 Formulation of FAS Multigrid

A description of the FAS Multigrid method can be found in Briggs *et al.* [49]. Only the final result as applied in this work is presented here. The discretised

Euler equations to be solved (see Section 2.2) can be written in the following non-linear algebraic form:

$$\mathbf{A}^h(\phi) = \mathbf{0} \tag{5.7}$$

where ϕ is the solution to the problem on the finest grid represented by the superscript ^{h}, and A is the coefficient matrix. When an iterative solution method is used, the above expression can be written as

$$\mathbf{A}^h(\mathbf{v}) \approx \mathbf{0} \tag{5.8}$$

where $\mathbf{v}$ denotes the latest approximation to ϕ. The multigrid solution process commences by performing a number of iterations or relaxation sweeps (employing the multi-stage Runge-Kutta (RK) method as detailed in Section 5.2) on the finest mesh. An improved approximate solution $\mathbf{v}$ is thus obtained. Next an altered form of the above discrete equation is solved on a coarser grid. The discrete equation solved for on the next coarser grid reads as

$$\mathbf{A}^{2h}(\mathbf{v}^{2h}) = \tau_h^{2h} \tag{5.9}$$

where the superscript $2h$ denotes the next coarser grid's values and expressions and the coarse grid correction term (also referred to as the source term). τ_h^{2h} is calculated as:

$$\tau^{2h} = \mathbf{A}^{2h}(I_h^{2h}\mathbf{v}^h) - I_h^{2h}\mathbf{A}^h(\mathbf{v}^h) \tag{5.10}$$

Here I_h^{2h} represents the so-called *restriction* operator which is defined in Section 5.4.2. To aid in the discussion later and to prevent confusion, $I_h^{2h}\mathbf{v}^h$ is from this point forward, referred to as $\mathbf{u}^{2h}$. Equation (5.9) is relaxed on the coarse grids to give a new approximation to $\mathbf{v}^{2h}$ (an approximate solution to the altered algebraic equation on the course mesh). Following on from the above, Equation (5.9) is set-up and solved for on all generated coarse meshes. Note that boundary conditions are not applied on coarse grids explicitly.

After having traversed all meshes (from finest to coarsest), the *error* calculated on each course mesh is passed back (formally known as *prolongation*) down to the next finer grid. On a generic coarse grid this error is calculated as:

$$\mathbf{e}^{2^i h} = \mathbf{v}^{2^i h} - \mathbf{u}^{2^i h} \tag{5.11}$$

from which the flow solution on the next finer mesh is updated as follows:

$$\mathbf{v}^{2^{i-1}h} = \mathbf{v}^{2^{i-1}h} + I_{2^i h}^{2^{i-1}h} \mathbf{e}^{2^i h} \tag{5.12}$$

where $I_{2^i h}^{2^{i-1}h}$ is the prolongation operator that will be defined in Section 5.4.3.

5.4.1 The V-cycle

The solution process outlined above in which meshes are traversed from fine to coarse and then back to fine is known as a V-cycle. This method is similar to work done in the field by other researchers [6, 19, 20, 53]. The V-cycle has been implemented in this work in the recursive form outlined by Briggs *et al.* [49]. In a concise form the recursive V-cycle can be represented by:

$$\mathbf{v}^{2^{i-1}h} \leftarrow V^{2^{i-1}h}(\mathbf{v}^{2^{i-1}h}, \tau^{2^{i-1}h}) \tag{5.13}$$

which implies that given the approximations $\mathbf{v}^{2^{i-1}h}$ and $\tau^{2^{i-1}h}$ the V-cycle function denoted V will return a new approximation $\mathbf{v}^{2^{i-1}h}$ to the current mesh level. The V-cycle function $V^{2^{i-1}h}$ is defined as:

1. Iterate (relax) $\mathbf{A}^{2^{i-1}h}(\mathbf{v}^{2^{i-1}h}) = \tau^{2^{i-1}h}$, v_1 times. Note that on the finest grid $\tau^{2^{i-1}h} = \mathbf{0}$

2. If $2^{i-1}h$ represents the coarsest grid then go to step 3.
   Else

   - Calculate $I_{2^{i-1}h}^{2^i h}\mathbf{A}^{2^{i-1}h}(\mathbf{v}^{2^{i-1}h})$ and $I_{2^{i-1}h}^{2^i h}\mathbf{v}^{2^{i-1}h}$
   - Compute $\mathbf{A}^{2^i h}(I_{2^{i-1}h}^{2^i h}\mathbf{v}^{2^{i-1}h})$ on the next coarse grid.
   - Calculate the correction source term $\tau^{2^i h} = \mathbf{A}^{2^i h}(I_{2^{i-1}h}^{2^i h}\mathbf{v}^{2^{i-1}h}) - I_{2^{i-1}h}^{2^i h}\mathbf{A}^{2^{i-1}h}(\mathbf{v}^{2^{i-1}h})$

- Set as an initial approximation on the coarse grid $\mathbf{v}^{2^ih} = I_{2^{i-1}h}^{2^ih}\mathbf{v}^{2^{i-1}h}$
- Evaluate this function again on the next mesh $\mathbf{v}^{2^ih} \leftarrow V^{2^ih}(\mathbf{v}^{2^ih}, \tau^{2^ih})$
- Compute the error $\mathbf{e}^{2^ih} = \mathbf{v}^{2^ih} - \mathbf{u}^{2^ih}$
- Update $\mathbf{v}^{2^{i-1}h} = \mathbf{v}^{2^{i-1}h} + I_{2^ih}^{2^{i-1}h}\mathbf{e}^{2^ih}$

3. Iterate $\mathbf{A}^{2^{i-1}h}(\mathbf{v}) = \tau^{2^{i-1}h}$, υ_2 times.

where υ_1 and υ_2 denote the number of relaxation sweeps performed on a mesh. Note that this definition of V is recursive because the function V is called from within itself (part of step 2). The optimal choice of υ_1 and υ_2 in the procedure has been found to be flow regime dependent, i.e. sub-, trans- or supersonic. Typical values used for υ_1 and υ_2 in this work range from as little as $\upsilon_1 = \upsilon_2 = 1$ on supersonic cases to, $\upsilon_1 = \upsilon_2 = 10$ for subsonic cases. The values used for each of the test cases is presented with the results (Chapter 6).

5.4.2 Restriction Operator

The restriction operator $I_{2^{i-1}h}^{2^ih}$ on the interior nodes of an agglomerated coarse mesh is computed as a volume weighted interpolation. The restriction of $\mathbf{v}^{2^{i-1}h}$ is computed as follows:

$$\mathbf{u}^{2^ih} = I_{2^{i-1}h}^{2^ih}\mathbf{v}^{2^{i-1}h} = \frac{\sum_{p=0}^{n} \mathbf{v}_p^{2^{i-1}h} V_p^{2^{i-1}h}}{V^{2^ih}} \tag{5.14}$$

where n refers to the total number of fine mesh volumes fused into the coarse mesh volume V^{2^ih}. The restriction of the $\mathbf{A}^{2^{i-1}h}(\mathbf{v}^{2^{i-1}h})$ term is performed similarly.

5.4.3 Prolongation Operator

The prolongation operator $I_{2^ih}^{2^{i-1}h}$ presents a challenge on unstructured agglomerated grids. This is as there is no obvious way in which to pass the errors between meshes in an accurate and efficient manner. To guarantee efficient convergence rates, the order of accuracy of the prolongation and restriction operators should satisfy the following relation [54]:

$$m_r + m_p > m \tag{5.15}$$

where m_r and m_p are defined as the highest degree plus one of the polynomials that can be exactly interpolated by the restriction and prolongation operators respectively. Further m is the order of the partial differential equation to be solved, which for the Euler equations is one. As the order of restriction operator defined in the previous section is one, m_r equals 2 and the relation is satisfied with a prolongation operator of order zero. The order zero prolongation operator can be written as

$$\mathbf{v}_p^{2^{i-1}h} = \mathbf{v}^{2^i h} \tag{5.16}$$

for any p. This implies that the value on the coarse mesh node is applied to all the fine mesh nodes (that make up that coarse mesh node). In this work it was found that this operator was not sufficient and that improvements in convergence could be realised if a higher order accurate prolongation operator was employed. As a result, two different higher order operators were tested. These are the inverse distance weighting operator of Watson [44] and the linear least squares method as documented by Blazek [53]. After experimentation, the linear least squares operator was found to result in superior multigrid performance and was subsequently employed for all the cases presented in this report. Note that the least squares method is presented in Appendix C.

5.4.4 Stabilisation

The stabilisation terms, described in Section 4.5, form part of the coefficient matrix $\mathbf{A}$ in the FAS Multigrid scheme. As the final solution on the fine mesh is not influenced by the solution on the coarser meshes, first order accurate schemes are sufficient here [53]. Therefore, on coarse meshes first order accurate stabilisation can be employed with the higher order stabilisation only used on the finest mesh. This tactic has been employed with success previously by Mavriplis & Venkatakrishnan [19] and Lallemand *et al.* [6]. In this work, it was found that it was only necessary to use this on the compressible cases.

5.5 Conclusion

In this chapter methods are presented by which to solve the discrete equations derived in the previous chapter. As the basic Jacobi method is slow to converge, a FAS multigrid method is employed. This method uses coarse grids generated by volume agglomeration. In the following chapter the developed modelling technology will be evaluated in terms of accuracy and computational cost.

Chapter 6

Results

6.1 Introduction

The preceding chapters detailed the mathematical development of a Cartesian mesh-based solver for the efficient modelling of inviscid incompressible and compressible flow. It now remains to evaluate the developed technology in terms of accuracy and computational cost. For this purpose, the solver was applied to a number of benchmark test cases from the literature, which span the entire scope of flow regimes. These are: incompressible flow over a cylinder and a Van de Vooren aerofoil at an angle of attack; sub- and transonic flow over a NACA0012 aerofoil at an angle of attack; and a double wedge under supersonic flow conditions. The results obtained using the solver are compared to the analytical solutions (where they exist) or the solutions of others. For the purposes of evaluation, the following comparisons are made:

- The predicted flow via both the standard and alternative schemes are compared to that of a benchmark solution.
- The improvement in *CPU time* obtained from multigrid is assessed for each discretisation method.
- The two discretisation methods are compared in terms of accuracy, as well as computational cost.

In the interest of a rigorous investigation, a mesh convergence study was performed for each test case using the grid convergence index (GCI) method of Roache [55] (this procedure is detailed in Appendix A). The GCI error bound obtained is quoted in all cases along with the ratio that indicates whether, in each case, asymptotic convergence has been achieved. Further, actual CPU times are quoted where computational performance is documented. The analyses were performed on a PC with 3GHz CPU and 2GB of 400MHz DDR RAM.

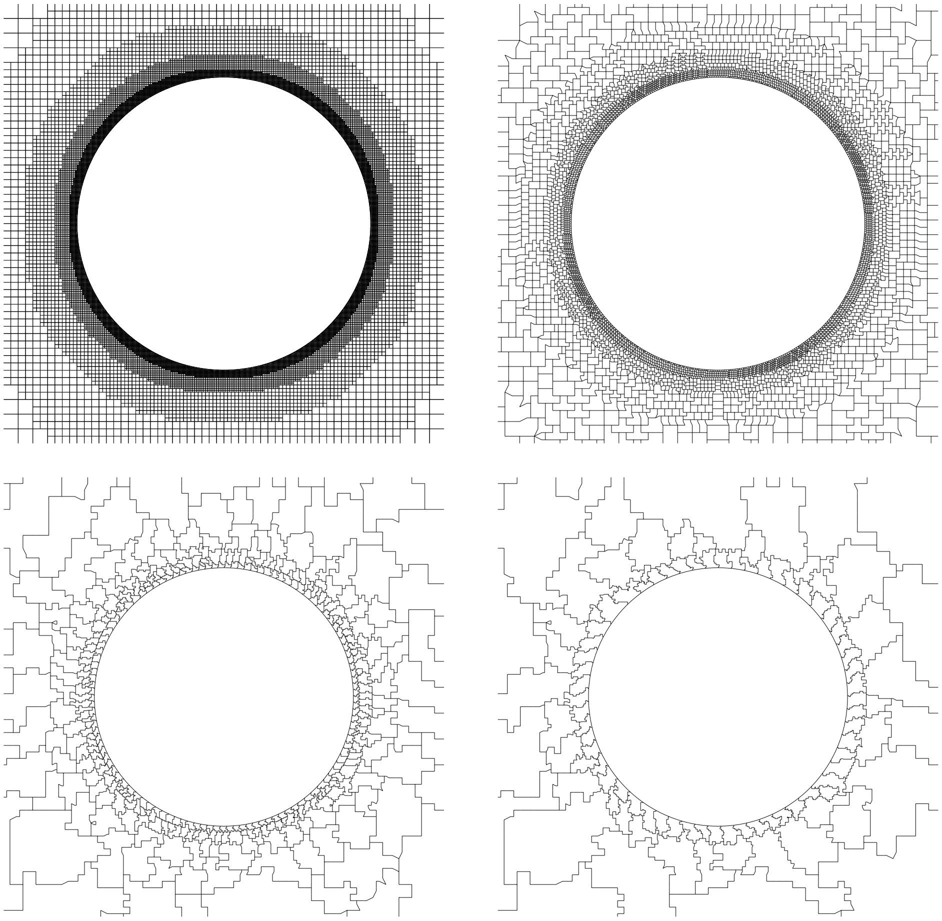

Figure 6.1: Cylinder incompressible flow test case - standard dual-mesh construction scheme: The fine mesh (top left) and corresponding first, second and fourth coarse dual meshes.

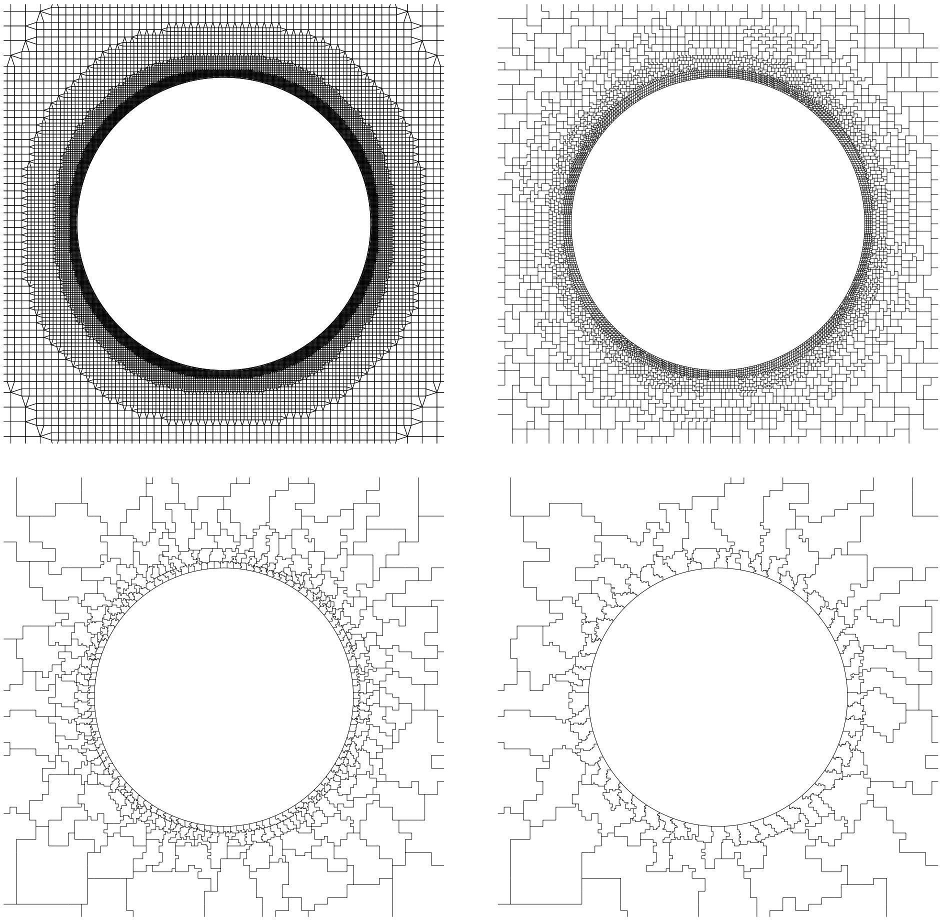

Figure 6.2: Cylinder incompressible flow test case - alternative dual-mesh construction scheme: The fine mesh (top left) and corresponding first, second and fourth coarse dual meshes.

6.2 Incompressible Flow: Cylinder in cross flow

The incompressible cross flow over a cylinder case poses a severe test for numerical schemes, as no natural physical dissipation effects are present [56]. Further, the problem contains two stagnation points while the analytical solution exists for validation purposes. The chosen case therefore provides a strict test for both the spatial discretisation accuracy as well as stabilisation aspects of a numerical scheme.

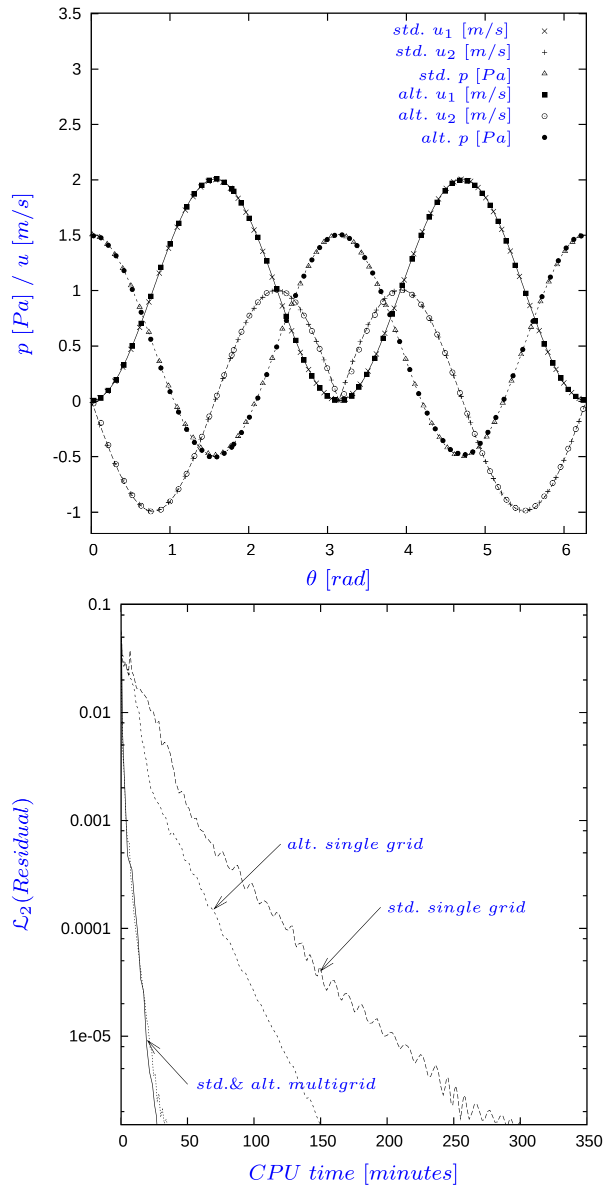

Figure 6.3: Cylinder incompressible test case: Cylinder surface pressures and velocities (top) where the lines denote the analytical solution and symbols are the calculated counterparts via the standard (std.) and alternative (alt.) schemes; and convergence plots (bottom). In the figure, θ is the angular position on the cylinder measured from the leading stagnation point in a clockwise direction.

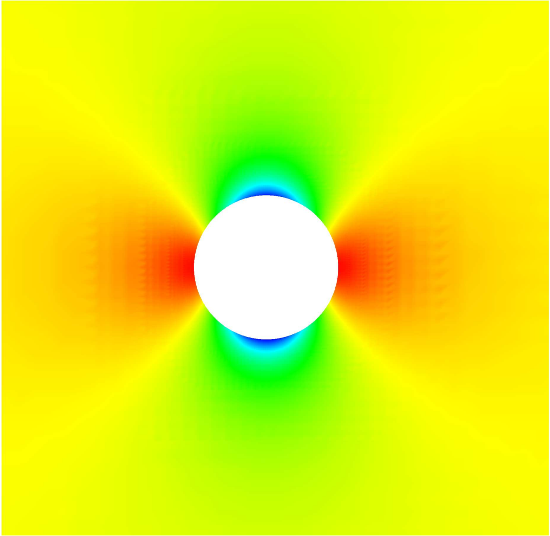

Figure 6.4: Pressure contours of cylinder test case.

The computational domain extends 10 diameters away from the cylinder. At the far-field boundaries characteristic boundary conditions (Section 2.2.3) are applied while at the cylinder surface, slip conditions are employed. Various meshes were generated for the purpose of the grid convergence study. A close-up view of the finest meshes employed are depicted in Figures 6.1 and 6.2 (top left). These meshes contained $27,070$ and $25,886$ computational nodes (cells) for the standard and alternative schemes respectively. The finest three meshes produced results which gave asymptotic convergence ratios of 1.033 and 1.06 for the standard and alternative discretisation methods respectively. These results indicate that for both discretisation schemes asymptotic convergence has indeed been achieved. The finest mesh produced a solution with a GCI error bound of 0.95% for the standard scheme, and 1.15% in the case of the alternative method (the error bounds are calculated by comparing the analytical and predicted pressures on the surface of the cylinder).

Five and four multigrid coarse meshes were employed for the standard and alternative schemes respectively (depicted in Figures 6.1 and 6.2). The closest node agglomeration strategy was used in the generation of these meshes. The node-based coarsening ratios achieved were between 3.6 and 4.5 for all successive

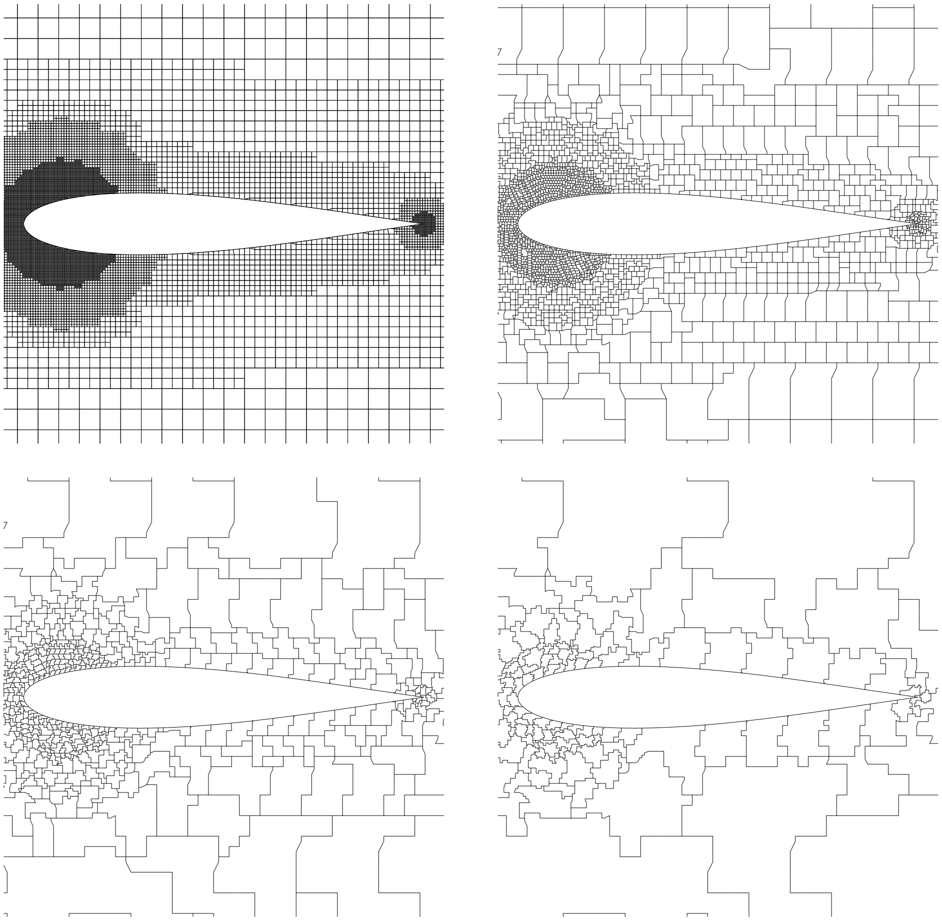

Figure 6.5: Van de Vooren test case - standard scheme: The fine mesh (top left) and corresponding first, second and third coarse dual meshes.

levels of mesh coarsening. The number of relaxation iterations in the up- and downward sweeps of the V-cycle (v_1 and v_2 from Equation 5.13) were set to 10 and 15 respectively.

The solution obtained and convergence plots are presented in Figure 6.3, with pressure contours plotted in Figure 6.4. As shown, both alternative and standard schemes resulted in similar solutions and at comparable multigrid computational cost. The speed-up achieved via multigrid for the standard scheme was 12, and 6 for the alternative. Note however, that although there is a reasonable disparity in speed-ups between the two schemes, the accelerated CPU times are almost

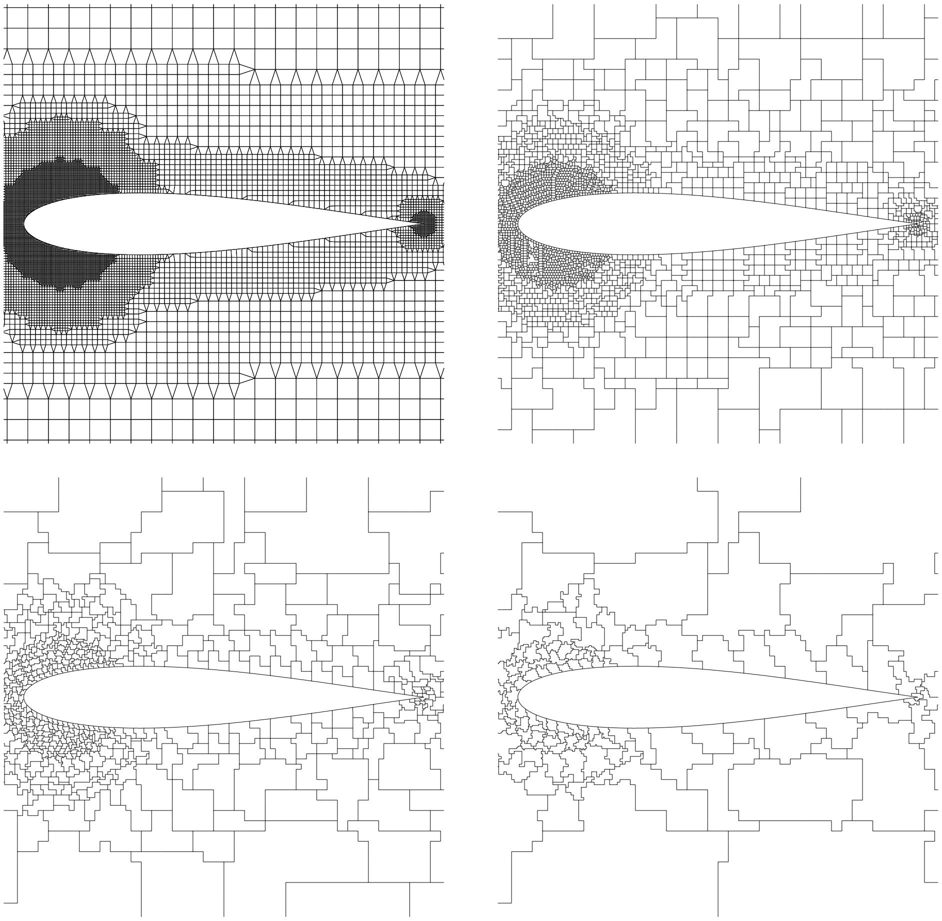

Figure 6.6: Van de Vooren test case - alternative scheme: The fine mesh (top left) and corresponding first, second and third coarse dual meshes.

identical.

6.3 Incompressible Flow: Van de Vooren Aerofoil

The second numerical example involves the incompressible flow over a *Van de Vooren* aerofoil with 15% thickness at 2° angle of attack. Once again this problem is purely convective and contains two stagnation points. An analytical solu-

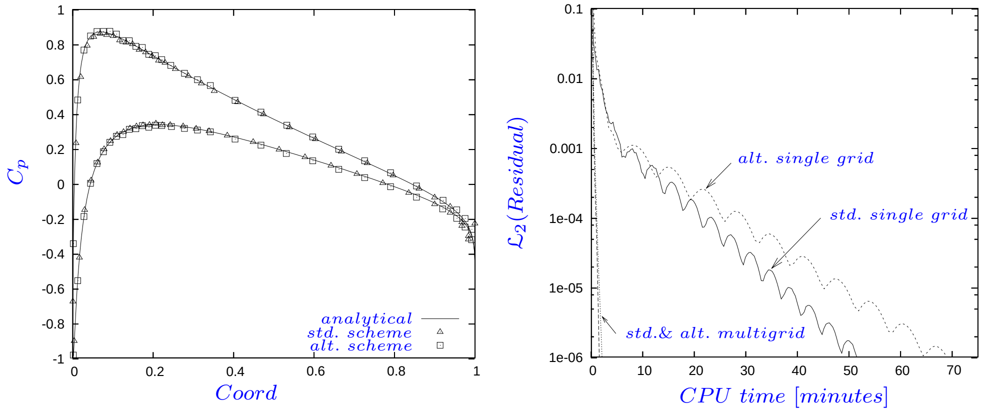

Figure 6.7: Van de Vooren test case - Solution (left) and convergence plot (right). Here *std.* and *alt.* denote the standard and alternative discretisation schemes respectively.

tion to this problem is also available. Similar to the previous test-case, slip type boundary conditions were applied at the surface of the aerofoil while characteristic conditions were prescribed at the outer boundary (15 chord lengths from the profile). The computational meshes employed for the standard and alternative dual-mesh construction strategies together with each set of multigrid agglomerated meshes, in the vicinity of the aerofoil, are shown in Figures 6.5 and 6.6 respectively. The fine meshes in each case contain $15,420$ and $14,748$ computational points. For the multigrid run, three coarse meshes, generated with the finite difference strategy, were used throughout. Node based coarsening ratios of between 3.8 and 4.6 were achieved. The V-cycle relaxation settings v_1 and v_2 were both set to 10.

Figure 6.7 compares the aerofoil surface pressure coefficients predicted via the various scheme variants to the analytical solution. From the convergence plot in the figure it is clear that the alternative method only marginally out-performs the standard form in terms of computational cost. Multigrid speed-ups of 35.46 and 30.49 are obtained for the standard and alternative cases respectively. The GCI error bound, based on the error between the computed and analytical solutions, is 0.22% for the standard case and 0.36% for the alternative case. Asymptotic convergence ratios of 0.996 and 1.11 were achieved for both cases respectively.

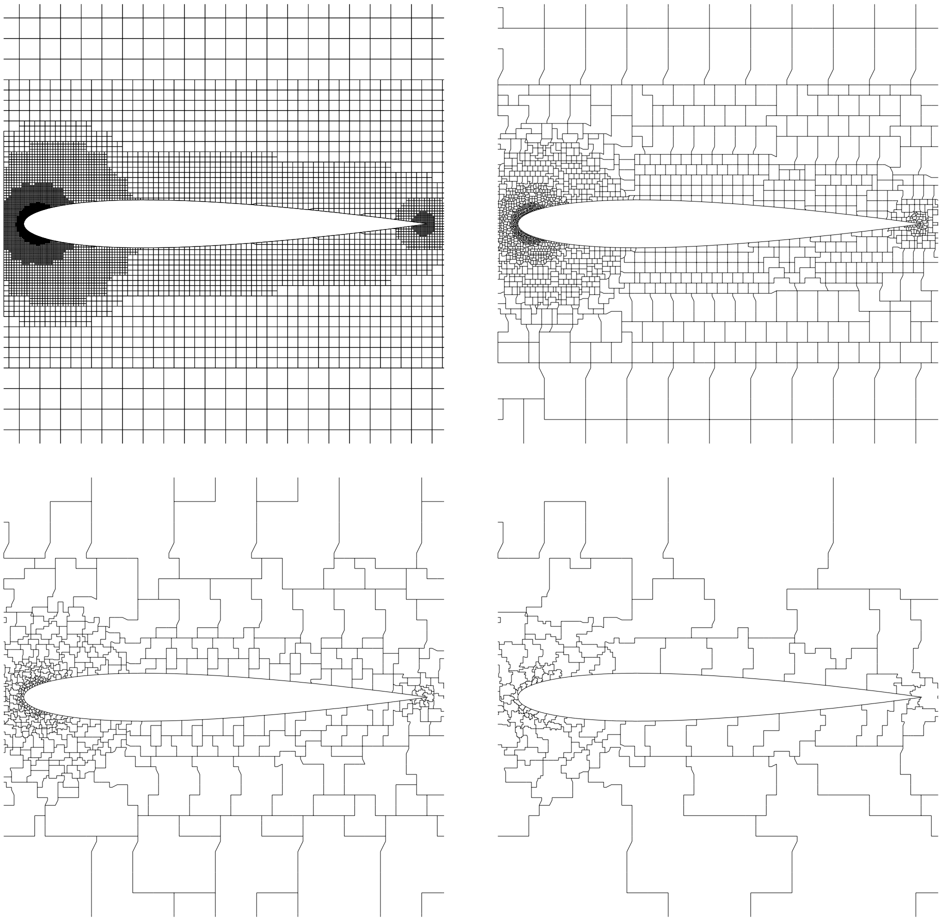

Figure 6.8: NACA0012 test cases - standard scheme: The fine mesh (top left) and corresponding first, second and fourth coarse dual meshes.

6.4 Subsonic Compressible flow: NACA0012

The first compressible test case presented here involves subsonic flow over a NACA0012 aerofoil at $M = 0.63$ and 1° angle of attack. The cut-cell meshes in the vicinity of the aerofoil resulting from the standard dual-mesh construction scheme and accompanying agglomerated coarse meshes (closest node agglomeration strategy) are depicted in Figure 6.8. A similar plot, Figure 6.9, shows that resulting from the new dual-mesh construction scheme. Four coarse meshes were used for this problem, with resulting coarsening ratios of between 4.0 and 4.8

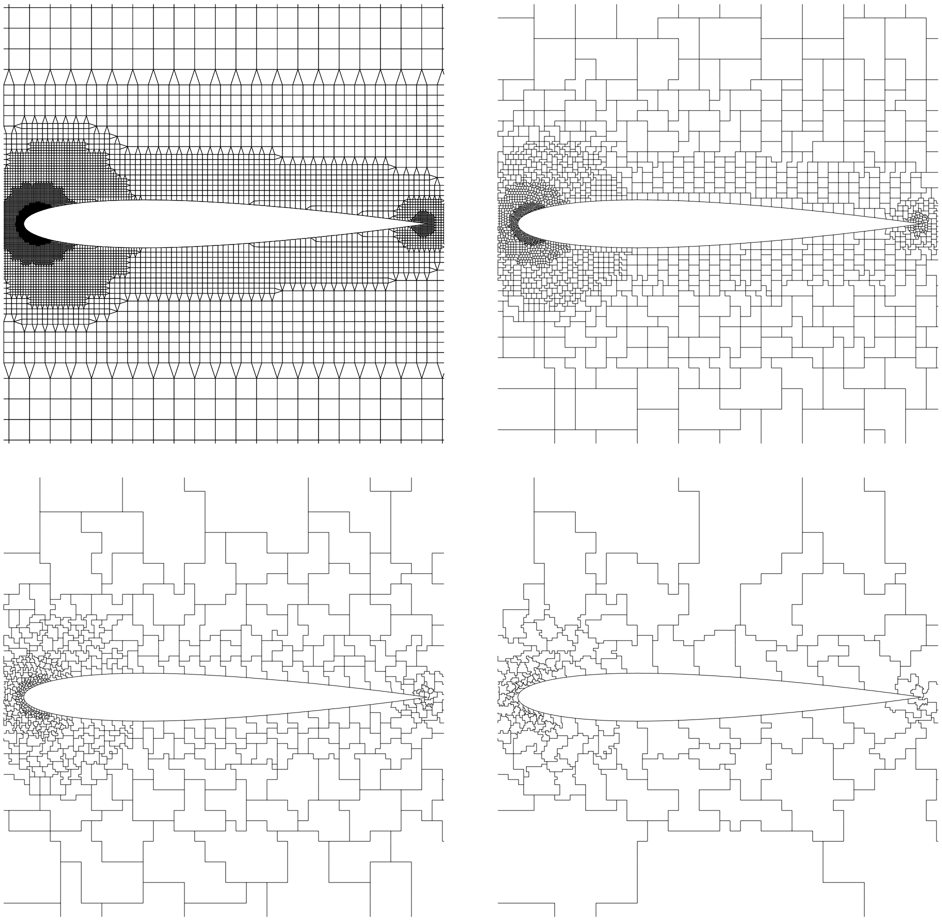

Figure 6.9: NACA0012 test cases - alternative scheme: The fine mesh (top left) and corresponding first, second and fourth coarse dual meshes.

for both schemes. The meshes extend 15 chord lengths away from the aerofoil surface. In the case of the standard dual-mesh construction strategy, the computational mesh contains $12,819$ nodes, while the developed dual-mesh strategy results in $12,160$ nodes.

Figure 6.10 compares the solution of Sørensen [20] to the predicted pressure coefficient on the surface of the aerofoil. For this test case, the standard and newly developed finite volume methodologies once again resulted in similar solutions, and at comparable computational cost. The GCI error bound, based on the computed lift coefficient, is 0.41% for the standard scheme, and 0.39% for the

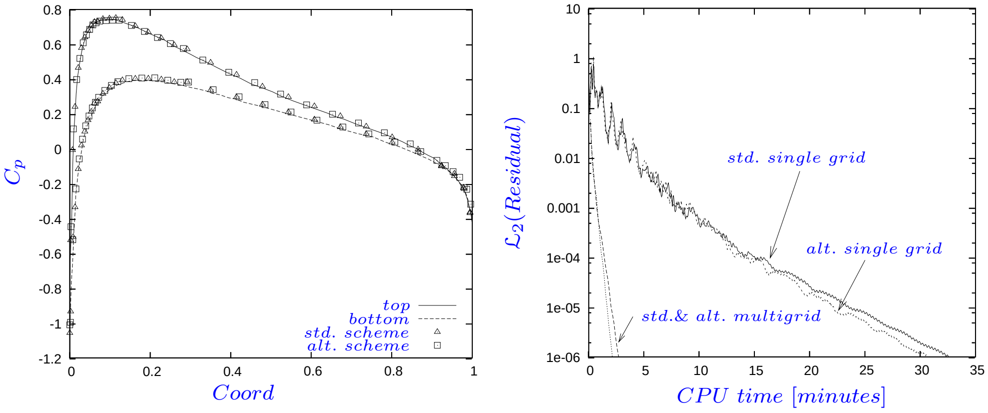

Figure 6.10: Solution (left) and convergence plot (right) for the subsonic test case. Here *std.* and *alt* denote the standard and alternative discretization schemes respectively.

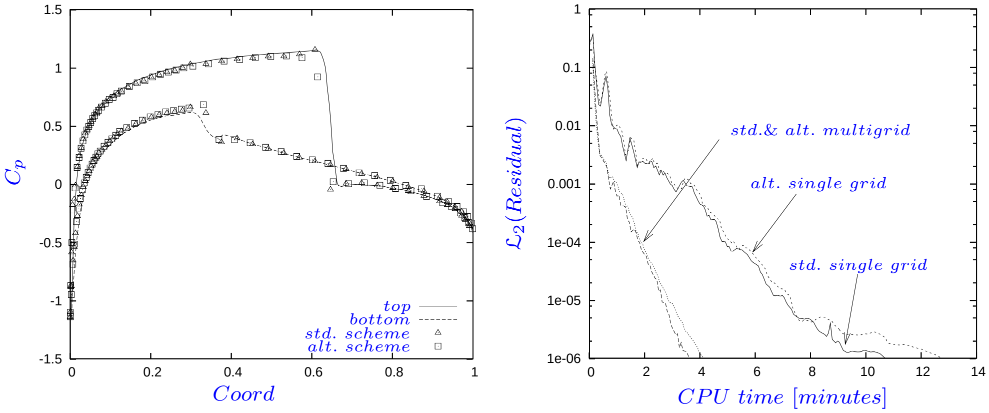

Figure 6.11: Solution (left) and convergence plot (right) for the transonic test case. Here *std.* and *alt.* denote the standard and alternative discretisation schemes respectively.

alternative method. Asymptotic convergence was achieved as indicated by GCI ratios of 1.001 and 1.02 for the standard and alternative schemes respectively. In terms of multigrid performance, the number of relaxation iterations in the up- and downward sweeps of the V-cycle found to produce best performance was 10 in both directions. The improvement in computational time achieved using multigrid, based on a drop in residual of 6 orders of magnitude, are circa 19 times

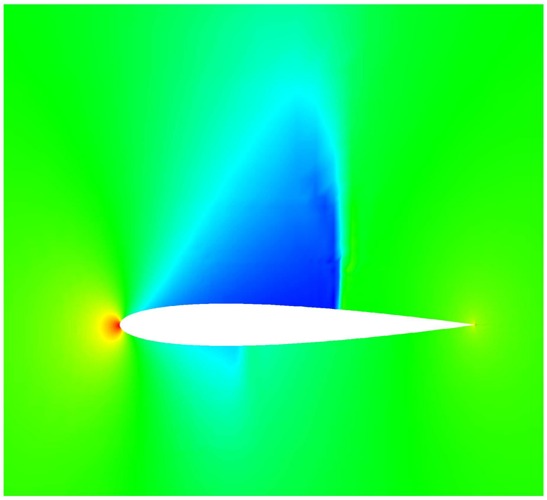

Figure 6.12: Pressure contours of the NACA0012 transonic test case.

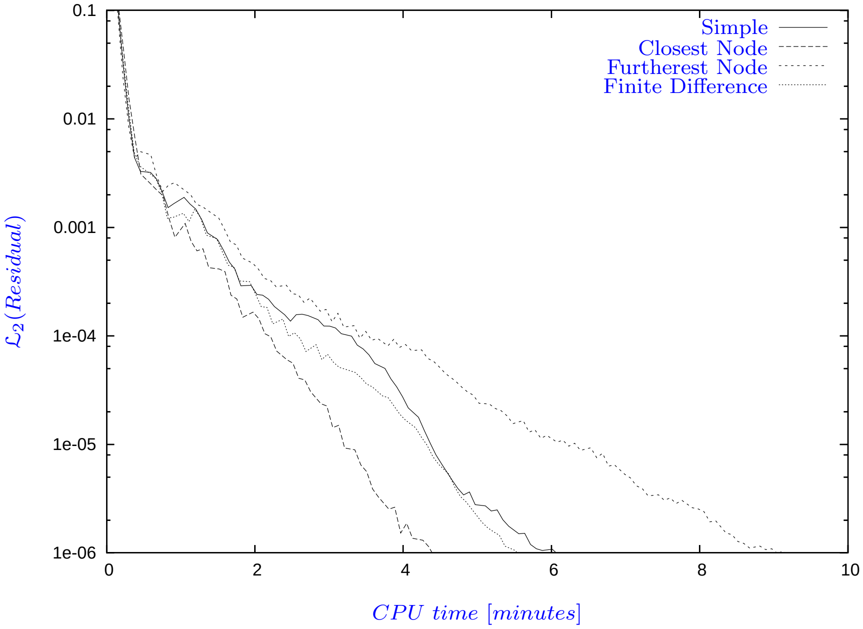

Figure 6.13: The effect of the choice of agglomeration strategy the NACA0012 transonic test case.

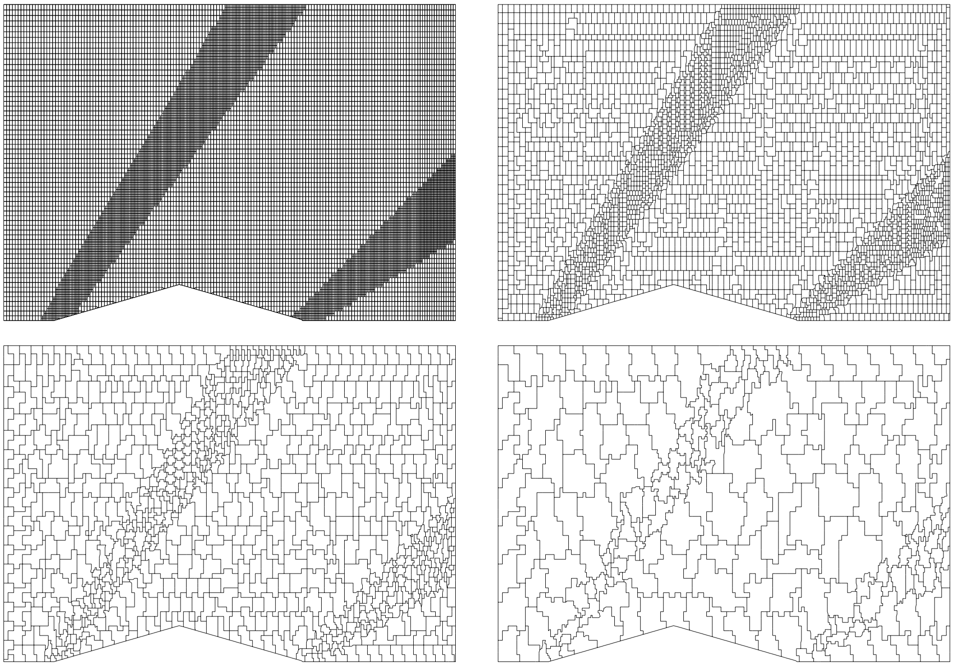

Figure 6.14: Supersonic test case - standard discretisation scheme: The fine and coarse dual meshes.

better than the Jacobi scheme for both discretisation schemes.

6.5 Transonic Compressible flow: NACA0012

The second compressible example is concerned with the transonic flow over a NACA0012 aerofoil at $M = 0.8$ and 1.25° angle of attack. The same computational meshes used for the previous test case are once again employed (Figures 6.8 and 6.9). The solutions obtained are compared to the benchmark solution in Figure 6.11. The GCI error bound for the standard scheme is 0.737%, and for the alternative scheme 0.835%. Asymptotic convergence was once again obtained as indicated by calculated ratios of 0.999 for both cases. Once again these values are based on the computed lift coefficient. The aforementioned is an indication of the equivalence in accuracy between the two schemes employed. As further shown in the figure, similar computational costs and multigrid speed-ups were

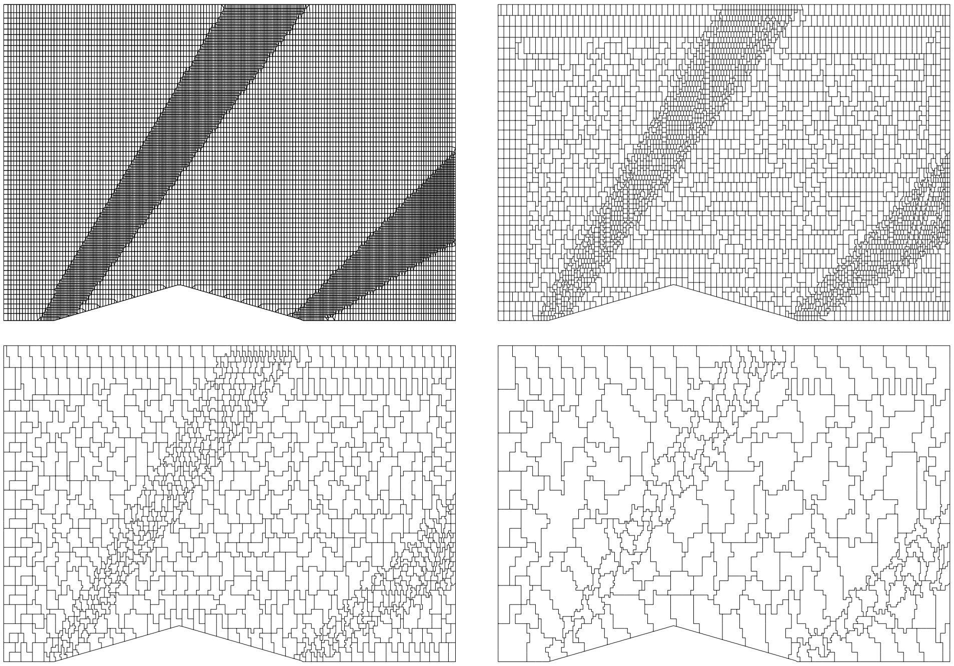

Figure 6.15: Supersonic test case - alternative discretisation scheme: The fine and coarse dual meshes.

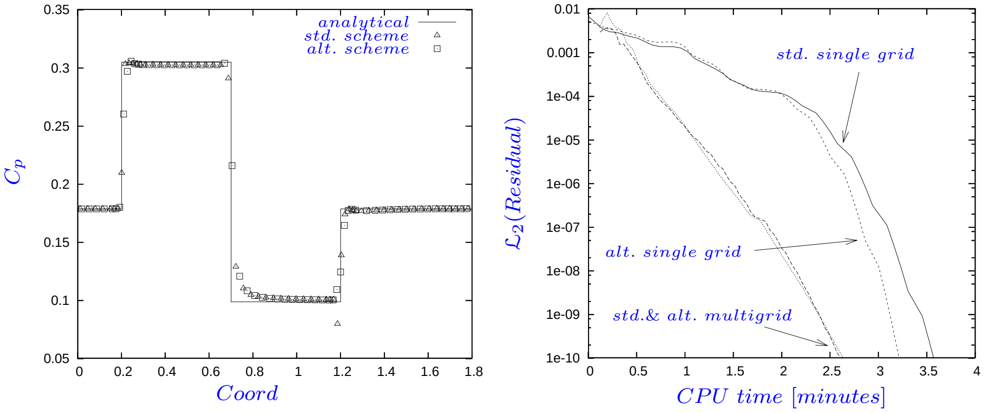

Figure 6.16: Supersonic test case: Solution (left) and convergence plots (right).

achieved. The latter was circa 3 in CPU time for a drop in residual of 6 orders of magnitude. In this case, the number of relaxation iterations was set to 4 on both up- and downward sweeps. A contour plot of the predicted non-dimensional pressure around the aerofoil is presented in Figure 6.12. The shocks on both upper and lower surfaces of the aerofoil are clearly visible.

Finally, as the transonic test case contains subsonic regimes as well as shocks, (being representative of a range of flow regimes) it was selected to evaluate the various coarsening strategies. The CPU times resulting from each are depicted in Figure 6.13. What is clear from the figure is that all but the "furtherest node" option resulted in similar CPU times. The latter was inferior to the others, which is expected due to it generating coarse meshes which are least representative of the fine mesh.

6.6 Supersonic Compressible flow: 10° Double Wedge

For the supersonic flow test case, the 10° double-wedge benchmark problem at $M = 2.0$ was modelled. The angle of attack for this problem is 0° and supersonic in- and outflow boundary conditions were employed to allow shocks to cross the computational domain boundaries. The fine mesh for the standard and new dual-mesh construction schemes is shown together with a selection of their agglomerated coarse meshes in Figures 6.14 and 6.15. The fine mesh on the standard scheme contains $16,237$ nodes and the alternative scheme, $15,770$ nodes. The finite difference agglomeration scheme was used to generate the coarse meshes. Node-based coarsening ratios of between 3.7 and 4.3 were achieved and a single relaxation sweep on each leg of the V-cycle was found to yield best convergence.

The solution is compared to the analytical solution in Figure 6.16. The GCI error bound for the standard scheme is 1.333% and the alternative scheme 1.264%, with GCI convergence ratios of 1.022 and 0.998 for the standard and alternative schemes respectively. These results are based on the error between the analytical and computed solutions. As per the previous test cases, the two discretisation schemes resulted in comparably accurate solutions and comparable computational cost and multigrid speed-ups.

Finally, this test case further provides a strict test for the far-field boundary conditions. This is as strong supersonic shocks cross this boundary. The calcu-

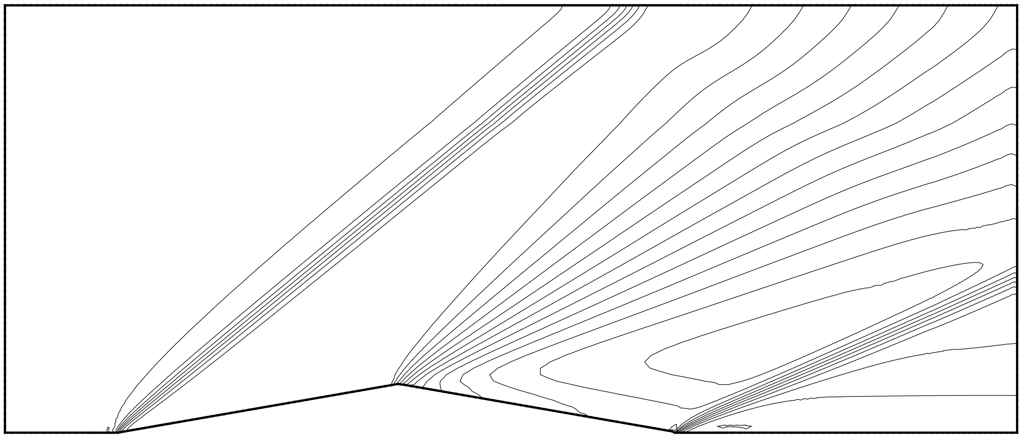

Figure 6.17: Mach number contours of the supersonic test case indicating the success of the non reflecting boundary conditions.

lated Mach number contours over the domain are presented in Figures 6.17 and 6.18. As shown, the shock clearly crosses the far field boundaries indicating the efficiency of the boundary modelling technology.

6.7 Spatial Convergence of Artificial Dissipation

The stabilisation scheme used in this work is not popular in the context of Cartesian meshes, which is thought to be due to the scheme being better suited to meshes with less dramatic mesh stretching ratios i.e. not a factor 2 or more. Although overall asymptotic convergence is achieved in the test cases presented, it is therefore of interest to quantify the influence of the artificial dissipation terms. Of interest here is firstly, if the dissipation terms are showing asymptotic convergence behaviour (these terms should tend towards zero), and secondly the relative magnitude of these terms as compared to the discretised spatial (convective) term. It is proposed to obtain a measure of this by comparing the spatial and the dissipation's contribution to the converged residual, to one another.

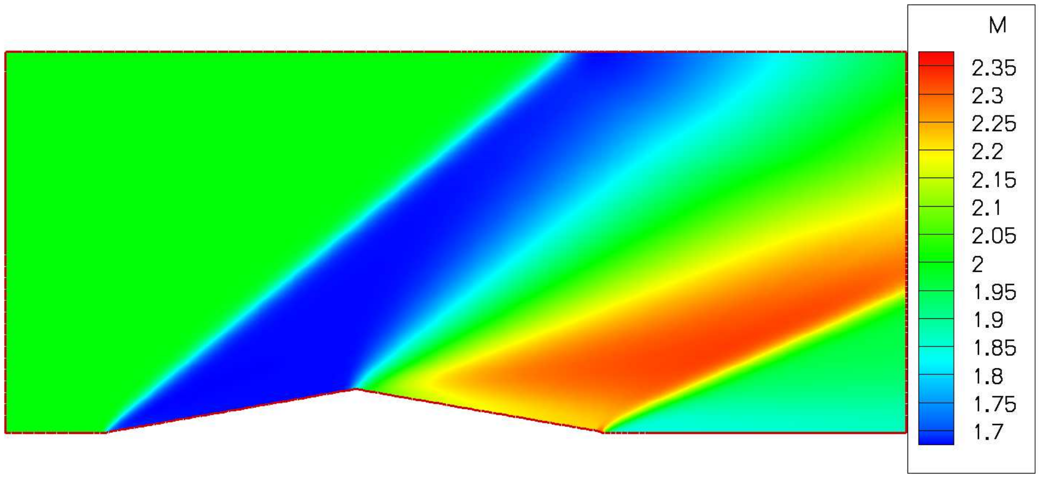

Figure 6.18: Mach number contours on supersonic test case.

For the purpose of this exercise, the subsonic case is considered as it does not contain any shock regions. This is so that the artificial dissipation can be assessed without the influence of the shock capturing (first order accurate) term. The Euclidean norm of the spatial contribution towards the residual as well as that of the dissipation terms, for various meshes, is shown in Figure 6.19. It can be seen from the figure that in the case of the standard scheme, the dissipation over the finest three meshes shows asymptotic like convergence to the same extent as the spatial terms. Further, the dissipation is always considerably smaller than the spatial terms. However, in the case of the alternative scheme, the aforementioned convergence is not evident, which is to the detriment of solution accuracy. This is thought to be due to the off-centre face positions (with regard to placing on the edges) being accounted for in the discretisation of the convective term, but not when calculating dissipation. This is the case as artificial dissipation does not naturally lend itself to account for off-centre face positions (apart from affecting the dissipation scaling parameter Λ (Equation 4.16)). It may therefore be advantageous to employ a high-resolution upwinding flux limiter in conjunction with the alternative discretisation method (that does explicitly account for off-centre face positions).

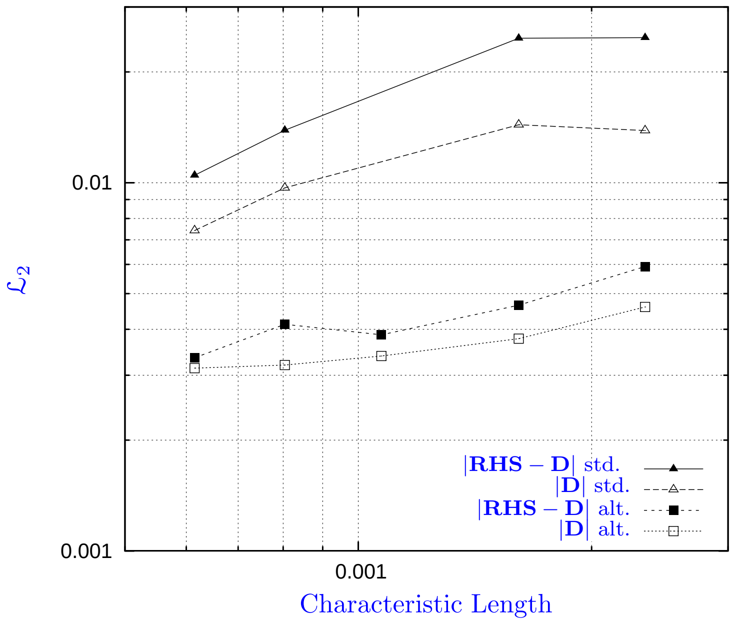

Figure 6.19: Spatial convergence of artificial dissipation terms on the NACA0012 subsonic test case.

6.8 Results discussion

Table 6.1 summarises all the numerical results presented in this chapter. In the first column, the number of nodes that the fine meshes contain is listed. Looking first at solution accuracy, the GCI error bound for all cases is less than 2% and there is little difference between the standard and alternative schemes. The ratio that indicates asymptotic convergence, denoted r_{asym}, is again in all instances comparably close to one. At this point a comment on the GCI process is necessary. The definition of the characteristic length parameter used for the evaluation of the GCI error bound and asymptotic convergence on a Cartesian mesh, is not obvious. In this work three definitions were investigated as described in Appendix A viz. $\mathtt{r} = \frac{N_1}{N_2}^{1/\mathcal{D}}$; $\mathtt{r} = \frac{E_1}{E_2}$ and where $\mathtt{r}$ is the smallest uncut edge in the mesh (not on a cut cell). Unfortunately, it was found that the GCI process is very sensitive to which definition is employed. It was found that the smallest edge length parameter resulted in GCI error bounds which appeared the most consistent with the actual error trend (difference between predicted and actual solution) and it was therefore selected for the purposes of the study. Other concerns about this process stem from the ideal requirement of a refinement ratio of circa 2. This is not always achievable with the meshes under consideration due to limitations in generating very fine meshes (memory cost) and accuracy concerns with very coarse meshes. It is important to note however that despite the aforementioned limitations, consistent convergence data is obtained from this study and that the GCI process does serve as an indicator of spatial convergence.

In terms of multigrid, near optimal average node coarsening ratios (r_{coarse}) were achieved for all cases. The multigrid speed-ups in CPU time range between a factor 2 and one order of magnitude. The speed up is seen to be flow regime dependent with the best performance achieved when no shocks are present. This may be alleviated by applying more advanced relaxation operators[20]. Unfortunately, a comparison of the multigrid performance to others was not possible in a quantitative sense, as little 2D multigrid accelerated results on Cartesian grids have been published. A result found in the literature, that is qualitatively comparable, is the 5 times speed-up achieved by Aftosmis *et al.* [28] on a 3D ONERA M6 wing transonic case. This is of similar order to that found in this work.

Finally, in the interest of completeness the CPU/time per node is given in the last column. This value is based on the time taken for the multigrid accelerated solution process to reach a residual of 1×10^{-6} (drop in residual of circa 5 orders). As opposed to multigrid speedup the worst performing case was that of the incompressible flow over a cylinder. This may be a flow / mesh related effect, but further investigation is in order.

6.9 Conclusions

The modelling technology presented in this work has been validated by application to incompressible and compressible sub-, trans- and supersonic flow problems. It is demonstrated that the new dual-mesh construction strategy with enhanced discretisation scheme, offers the same accuracy as the standard scheme. Both methodologies were found to have similar computational cost and convergence characteristics. Achieved multigrid speed-ups ranged from one order of magnitude for subsonic flow, to a factor 2 for the supersonic case.

Test Case	**Finest mesh size (nodes)**	**GCI**	$\mathbf{r_{asym}}$	$\mathbf{r_{coarse}}$	**Multigrid speed-up**	**CPU-time/Node (sec/node)**
Incompressible						
Cylinder std.	$27,070$	0.95%	1.033	3.81	11.8	0.059
Cylinder alt.	$25,886$	1.15%	1.06	3.87	6.0	0.081
Van de Vooren aerofoil std.	$15,420$	0.22%	0.996	3.87	35.46	0.0033
Van de Vooren aerofoil alt.	$14,748$	0.36%	1.11	3.89	30.49	0.0047
Compressible						
NACA0012 subsonic std.	$12,819$	0.41%	1.001	3.92	12.21	0.0127
NACA0012 subsonic alt.	$12,160$	0.39%	1.02	3.87	14.04	0.0105
NACA0012 transonic std.	$12,819$	0.74%	0.999	3.92	3.08	0.0168
NACA0012 transonic alt.	$12,160$	0.84%	0.999	3.87	2.94	0.0204
Double wedge supersonic std.	$16,237$	1.333%	1.022	4.11	1.35	0.0053
Double wedge supersonic alt.	$15,770$	1.264%	0.998	4.05	1.23	0.0058

Table 6.1: Summary of results. Here r_{asym} refers to the ratio that indicates asymptotic convergence and r_{coarse} refers to the average node based coarsening ratio between coarse grid levels.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The aim of this project was to develop an in-house CFD tool that could be used effectively in the concept phase of an aerodynamic design. This tool solves the Euler equations for compressible and incompressible flows through the use of a single equation set. These equations are bounded though the application of characteristic far-field boundary conditions that have been extended in their application to incompressible problems.

Non-conforming cut-cell Cartesian meshes are employed for numerical solution purposes. These meshes can be generated with great efficiency on complex geometries and have a high tolerance for so-called "dirty" geometries. Commercial mesh generators are employed to generate these meshes but are limited to generating 3D meshes only. Thus code was developed to extract a 2D mesh from a 3D one. Related to this was the development of a cell cutting algorithm to furnish a body conforming mesh using an overlapping grid (that results from the aforementioned extraction) as input. This algorithm was developed such that it retains the innate tolerance for "dirty" geometries in Cartesian meshes.

As the CFD tool developed in this work was started from scratch, alternative techniques, which are not at present popular in the context of cut cell Cartesian meshes, were investigated. The foremost of these are the single equation set used to model both incompressible (via artificial compressibility) and compressible flow and the use of the vertex-centred finite volume discretisation strategy. This strategy operates in an unaltered form in the cut cell and hanging node regions of the mesh and should prove competitive with the cell centred scheme prevalent in the context of cut cell Cartesian meshes. As the accuracy of the scheme in the region of hanging nodes was expected to be reduced to first order, an alternative dual mesh construction strategy was developed and the discretisation suitably enhanced. The scope of this study was limited to applying this alternative strategy to 2D problems only, but it is expected that it will easily extend to 3D. To prevent the odd-even decoupling of the solution known to occur with this type of discretisation, the Jameson, Schmidt & Turkel [1] artificial dissipation scheme, which is novel in the context of Cartesian meshes, was employed.

Explicit multi-stage Runge-Kutta temporal discretisation is employed to obtain the steady-state solution to the discretised governing equations. This is coupled with FAS multigrid to accelerate convergence. The coarse grids used in the multigrid process were generated through a generic volume agglomeration technique. This method of generating the coarse grids is novel in the context of Cartesian meshes.

The developed modelling technology was evaluated by applying it to suitable test cases that cover the entire flow range of interest. The accuracy of the scheme is evaluated by comparing the predicted flows to published or analytical solutions. The multigrid solution acceleration strategy employed in this work has been shown to provide reductions in the CPU time needed for a converged solution of between circa two and twenty. It is also noted that the standard and alternative dual mesh construction strategies provide similar results with similar convergence characteristics. In an attempt to provide a scientific validation process, a grid convergence study, based on the calculation of the GCI error bound, was performed for all test cases. Overall asymptotic grid convergence was obtained throughout, with error bounds of less than 2% for all cases. To further evaluate the method and to test the influence of the dissipation scheme, a study was performed on the grid related convergence of the latter. It was found that although the dissipation was behaving appropriately when employed together with the standard dual mesh construction, it was not in the case of the alternative scheme. This is suspected to be due to the dissipation scheme not naturally lending itself to account for off-centre face positions that occur with the alternative construction. It is believed that this should be further investigated by the application of a more compact stabilisation method.

7.2 Future Work

The scope of this study was limited to the solution of the 2D Euler equations. A 3D preprocessor, to implement the standard vertex-centred discretisation strategy, was developed but the alternative form was not implemented. The following are recommended extensions of this work:

- First and foremost, is the extension of the solver to 3D as well as the implementation of the alternative discretisation scheme into the 3D preprocessor.
- As discussed the artificial dissipation does not work well together with the alternative dual mesh construction scheme. It is recommended that a more modern stabilisation scheme be implemented and the alternative dual mesh construction re-evaluated. Here it is important that the stabilisation scheme selected be capable of taking into account the off-centre face positions in the dual mesh.
- In this work, grid independent solutions were obtained by brute force i.e. generating a mesh, obtaining a solution and then refining the mesh manually where necessary. This method is not practical in the context of a concept screening tool and a more elegant solution could be obtained by the implementation of a grid adaption routine.

Appendix A

Verification and Validation of Code

A.1 Introduction

In this work the results of the code are verified using the techniques presented by Slater [57] in his tutorial. This tutorial follows published AIAA Guidelines [58]. It should be noted here that there is no set standard for the verification and validation of CFD software. Indeed there is, according to Slater [57], "professional disagreement" on the exact procedures for this.

A.2 Convergence Analysis

Two types of convergence are assessed in this study. The first, iterative convergence, is a concerned with how much a solution changes per iteration. The second, spatial or grid convergence, deals with how much a solution changes when the grid is refined. An iterative convergence study for each test case can be found the relevant sections in Chapter 6. Here, solutions in which the Euclidean norm of the residual (all terms on the right hand side of Equation 4.21) drops below $1e - 6$ are said to have converged.

The procedure for verifying spatial grid convergence is based on the calculation of three parameters that indicate if grid convergence has been achieved. These parameters are; f_{exact} an estimate of the solution at zero grid spacing, the so-called Grid Convergence Index (GCI) which is an estimated error bound for the spatially converged solution, and r_{asym} which indicates if asymptotic convergence is achieved. These parameters are calculated from the results of three successively refined meshes, differentiated by the definition of a characteristic length.

No standards exist for determining the characteristic length of a Cartesian mesh and in this work three different characteristic lengths are defined. The first is the physical length of the one side of the smallest Cartesian cell in the mesh i.e. not a cut-cell. A further characteristic length based on the total number of nodes is defined as per the tutorial as

$$\mathtt{r} = \frac{N_1}{N_2}^{1/\mathcal{D}} \tag{A.1}$$

where $\mathtt{r}$ is the grid refinement ratio, N the total number of nodes from a mesh and $\mathcal{D}$ is the mesh dimension. Note that $N_1 > N_2$. The final parameter used is based on the number of edges in the mesh and is defined as simply

$$\mathtt{r} = \frac{E_1}{E_2} \tag{A.2}$$

where E is the number of edges in the mesh. Similar to the above convention, $E_1 > E_2$. All three characteristic lengths are used in the following stages and the results are compared. To obtain accurate results it is desired that the characteristic lengths of successive grids vary by a ratio of 2 or as close as possible to this. In other words if a grid has a spacing of h then the finer grid would have a spacing of approximately $\frac{h}{2}$.

All three of the parameters to be calculated depend on an estimate of the order of convergence of the scheme. To find this value, denoted $\mathtt{p}$, it is instructive to look at the error in any numerical method. The error $\mathbf{e}$ is defined as the difference between the actual solution ϕ and the estimated solution $\mathbf{v}$ and can be written as

$$\mathbf{e} = \phi - \mathbf{v} = Ch^{\mathtt{p}} + \text{Higher order terms} \tag{A.3}$$

where C is a constant. This can further be rewritten as a function of a characteristic parameter of the solution, such as lift, denoted f from here on,

$$\mathbf{E} = f_{exact} - f^{h} = Ch^{\mathtt{p}} + \text{Higher order terms} \tag{A.4}$$

There are three unknowns in this equation and thus it can be solved with the solutions of three successively refined grids. To this end the above equation is rewritten in the following form.

$$\frac{f^{4h} - f^{2h}}{f^{2h} - f^{h}} - \frac{1 - \mathtt{r}^{\mathtt{p}}}{\frac{1}{\mathtt{r}^{\mathtt{p}} - 1}} = 0 \tag{A.5}$$

This equation is solved for $\mathtt{p}$ using the bisection method [59]. It has been previously stated that the discretisation used in the work is notionally second order accurate. This means that, in this work, $\mathtt{p} \approx 2$.

Given the value of $\mathtt{p}$ the first of the three parameters, f_{exact} can be calculated. This parameter, an estimation of the exact solution of a numerical problem at zero grid spacing, is calculated using a Richardson extrapolation as suggested by Roache [55]:

$$f_{exact} = f^{h} + \frac{f^{h} - f^{2h}}{\mathtt{r}^{\mathtt{p}} - 1} \tag{A.6}$$

The second parameter, the GCI, is now calculated, for each pair of meshes as:

$$GCI_0 = \mathfrak{F}_s * \frac{\left|\frac{f^{2h} - f^{h}}{f^{h}}\right|}{\mathtt{r}^{\mathtt{p}} - 1} GCI_1 = \mathfrak{F}_s * \frac{\left|\frac{f^{4h} - f^{2h}}{f^{2h}}\right|}{\mathtt{r}^{\mathtt{p}} - 1} \tag{A.7}$$

where $\mathfrak{F}_s$ is a factor of safety. In this work the value of 1.25 was used. Using this information the third parameter can now be calculated,

$$\mathtt{r}_{\mathtt{asym}} = \frac{GCI_1}{2^{\mathtt{p}} GCI_0} \tag{A.8}$$

This convergence parameter provides information on whether asymptotic convergence has been achieved. Asymptotic convergence is said to have been achieved when it is evident that further levels of refinement will not change the value of `p`. This is indicated by the value of r_{asym} which will tend to one, once asymptotic convergence has been achieved.

Appendix B

Preprocessor

B.1 Introduction

The major parts of the preprocessing phase of a run are covered in Chapter 3. In this section a brief description of how the preprocessor interprets the geometry for the cell-cutting algorithm will be given. Further, the method employed to change the element based data structure of the mesh from the mesh generator to the edge based data structure used in the solver is discussed.

The preprocessor and the preprocessing phase of any run is responsible for taking the mesh generated by a mesh generator and delivering it in a form suitable for the solver i.e. to perform all computations on the mesh that do not change per iteration in the solver. This phase includes the extraction of a 2D mesh from a 3D mesh, cell-cutting, marking boundaries for the application of boundary conditions and applying the discretisation strategy.

This phase is to be performed on meshes that may contain, in 3D, *hexahedron (brick)*, *tetrahedron*, *prism (wedge)* and *pyramid* type elements and in 2D *trianglar* and *quadrilateral* type elements. Further these meshes will contain hanging nodes. The preprocessor should also be fast and exhibit $O(N)$ complexity, where N is the number of nodes and should be able to interpret the chosen mesh generators file format (Gambit neutral file format).

B.2 Geometry Interpretation

A multitude of geometry definitions are available. These range from very simple systems such as STL (which is just a surface triangulation), to complex NURBS format such as STEP. There also are a number of propriety formats available, with an associated licence fee. Aftosmis [60] uses a surface triangulation approximation of the CAD geometry. This triangulated geometry is then interpreted by the preprocessor. Charlton [50] uses a parametric representation of the geometry. In this work it was decided that the geometry be interpreted by the following existing geometry protocols STL (used by Harpoon), STEP and IGES. OpenCASCADE [61] open source C++ libraries were employed for this purpose as it contains all the necessary file readers and geometry functions.

B.3 Mesh Data Structure Manipulations

In this work the preprocessor was separated into two separate codes for the 2D and 3D cases respectively. This is not strictly necessary but it was found that many of the algorithms used in the 2D work did not simply scale to 3D. This is particularly noticeable in the algorithms used to change the element-based data structure into an edge-based one.

A mesh generated using a typical mesh generator consists of a numbered node table containing node coordinates and an element table consisting of node numbers that make up individual elements. Other information such as boundary nodes is communicated differently in the various formats but typically a list of two dimensional boundary elements is stored. This is the case with the Gambit generic mesh file format that the chosen mesh generator [26] is capable of generating. This is what is referred to as the element based format. The construction of the edge-based vertex-centred dual-mesh requires the addition of, among other things, an edge table to this existing format.

The mesh from the mesh generator is made up of four very specific 3D element types and two, 2D element types. Boundaries between elements are not explicitly defined and it is possible for elements of different types to lie next to one another.

This results in both duplicate edges as well as overlapping faces. As the latter two are mathematically unacceptable entities, they are removed by the preprocessor. A graphical view of some of these discrepancies encountered in the various meshes is presented in Figure B.1. This process will now be described.

To aid the discussion of the removal of these discrepancies it is necessary to first mention the data structure of both 2 and 3D elements. It is obvious that in removing these inconsistencies one can no longer retain the specific element types used by the mesh generator and that a more generic format must be developed. The format chosen in this work requires that three data-tables be constructed for *each* element. The three tables are:

- A list of nodes. A numbered list of nodes that form the element.
- A face table. This contains a list of all faces in the element in terms of local node number as defined by the above list.
- An edge table. This table lists all the edges in the element, once again in terms of local node numbers.

All elements contain these tables but in the case where an element is of a standard type, it is unnecessary to store the face table and edge table more than once for the whole mesh. In 2D and for the 2D boundary element table in the 3D case the face and edge tables are redundant. Instead nodes are stored in the element's node list consecutively such that each node is connected with an edge to its neighbours in the table.

The procedure for removing these discrepancies is divided into two phases, detection and removal. In the 2D case the only discrepancies that occur are those surrounding hanging nodes. To detect hanging nodes in this case a table of all elements in which a node appears is created. As hanging nodes only occur in Cartesian mesh regions one can correctly assume that an interior hanging node is only surrounded by two neighbouring elements[1]. On the boundary any existing hanging nodes are detected because they are those boundary nodes that occur in only one element.

[1] Note. This step is performed after the splitting of the 3D mesh into a 2D mesh and before the cell-cutting occurs.

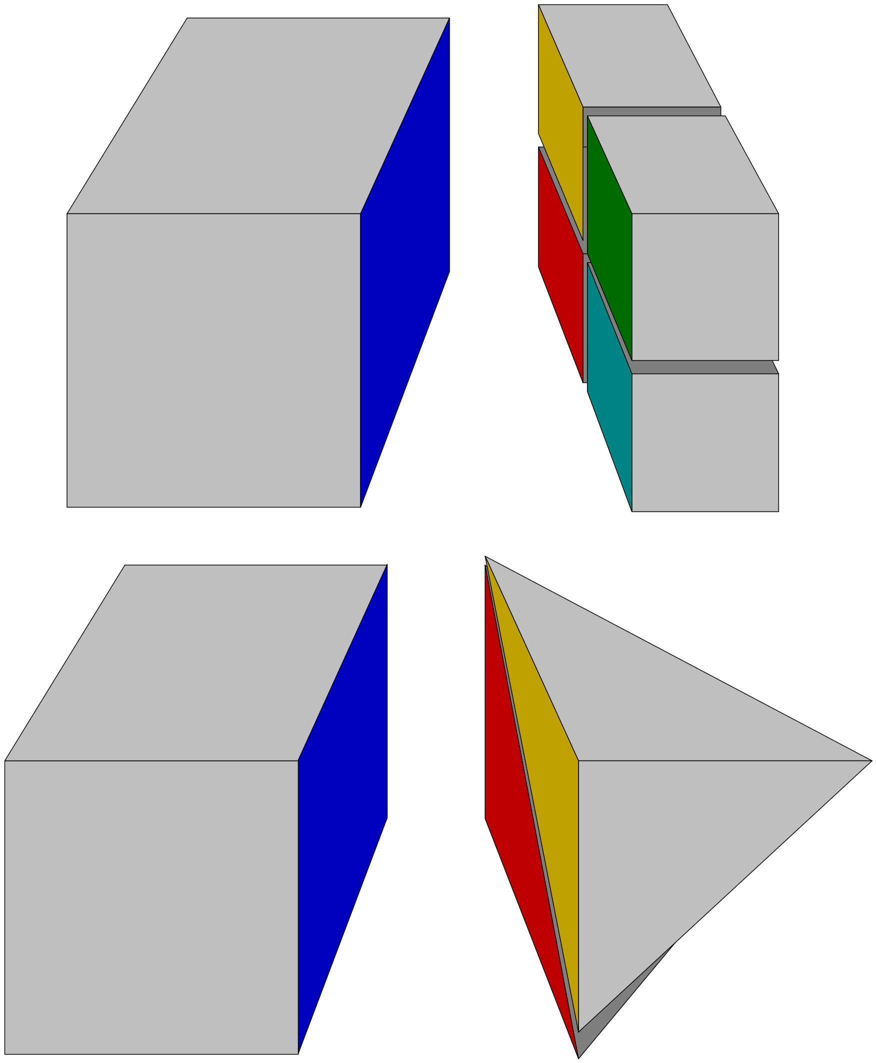

Figure B.1: Face discrepancies in element-based data structure. In this figure adjacent elements have been moved apart to reveal the adjacent element faces (coloured). It is then evident that the edges on the blue face on the left are not duplicated exactly on the right. This implies duplicate edges and is corrected by splitting up the blue face on the left to reflect the coloured faces.

The discrepancies caused by internal hanging nodes are removed by finding the missing neighbouring element and adding the hanging node to its node table. To reduce searching in the mesh, these elements are found using the node-element table.

In 3D the procedure is different as mesh discrepancies are caused by mismatched faces as opposed to mismatched edges. These mismatches are not only caused by hanging nodes but also by different elements types neighbouring each other. An algorithm based on numbers of elements connected to a node is bound to fail as there is no clear-cut criterion here. Instead discrepancies in the mesh are detected by checking if every face in every element has a neighbouring face in another element[1]. Should a face not match any of its neighbouring faces, that face is then marked. A search is then conducted of all unmatched faces to find faces that should be combined. These will be faces that are co-planar with- and completely contained within another unmatched face. If one is looking for neighbours of a small face that lie within a large face, this condition will not be met. This is however, of no concern as the link will be made from the larger face.

Once the mismatched faces have been found, they are described consistently in the large element by adding faces to that elements face table. As this process then invalidates other faces the procedure is performed again to fix those faces. It has been found that only two loops are necessary to fix any mesh although this is not assumed in the program and this procedure will be repeated until no further inconsistencies have been found. This procedure has been found to be robust and capable of redefining all inconsistent faces encountered to date. Having removed all these discrepancies an edge table is built for the dual mesh construction.

B.4 Speed

To compliment the speed of mesh generation and solution process it is important to make the preprocessing step as fast as possible. In this work we seek to ensure

[1] Note that this holds even on the boundary as all boundary faces are duplicated in a 2D boundary element table. Thus each boundary element should match with the corresponding face in its neighbouring 3D element.

the computational cost of preprocessing a mesh varies linearly with the number of nodes in the mesh. That this has been obtained is clear from Figure B.2 for the 2D preprocessor and Figure B.3 for the 3D preprocessor.

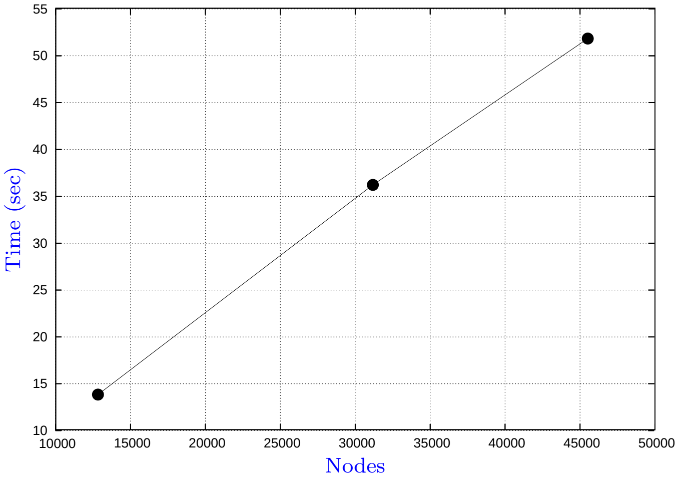

Figure B.2: Performance of the 2D preprocessor in terms of time on three NACA0012 test cases.

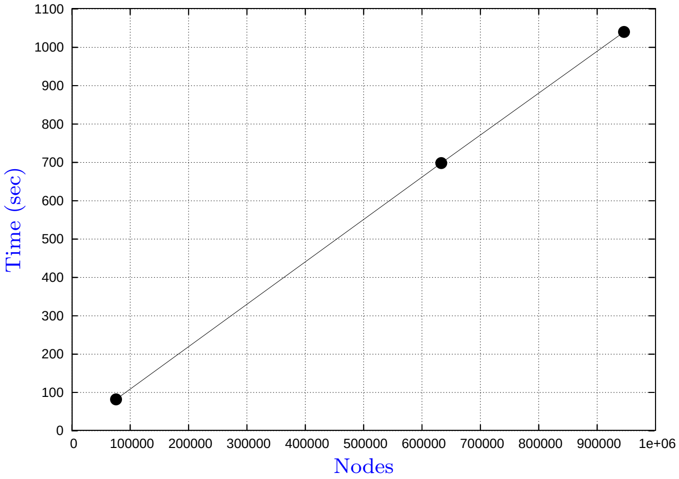

Figure B.3: Performance of the 3D preprocessor in terms of time on three NACA0012 test cases.

Appendix C

Extrapolation and Interpolation Procedures

For the application of the characteristic boundary conditions and the prolongation operators extrapolation and interpolation operators are needed. Two such operators have been employed. These are the method of linear least squares and the method of inverse distance weighting of Watson [44].

The method of linear least squares attempts to generate a planar approximation to the values at the selected points. This plane is then used to provide an approximation to the value at a particular point of interest . To determine the coefficients of the plane $\mathcal{P}(x_0, x_1) = a_0 + a_1 x_0 + a_2 x_1$ the following equation must be solved,

$$\begin{bmatrix} \sum_{i=1}^{M} \mathcal{P}_i \\ \sum_{i=1}^{M} x_{1i}\mathcal{P}_i \\ \sum_{i=1}^{M} x_{0i}\mathcal{P}_i \end{bmatrix} = \begin{bmatrix} M & \sum_{i=1}^{M} x_{0i} & \sum_{i=1}^{M} x_{1i} \\ \sum_{i=1}^{M} x_{1i} & \sum_{i=1}^{M} x_{1i}^2 & \sum_{i=1}^{M} x_{0i}x_{1i} \\ \sum_{i=1}^{M} x_{0i} & \sum_{i=1}^{M} x_{0i}x_{1i} & \sum_{i=1}^{M} x_{0i}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \tag{C.1}$$

where x_0 and x_1 are the coordinate directions, $\mathcal{P}$ refers to the value of interest at each selected point. M is the total number of selected points. A direct Gauss-elimination solver [59] was coded to solve this equation.

The method of inverse distance weighting computes a weighted average of all selected points to provide an estimate of the value at a point of interest. The following equation is used

$$\mathcal{P}_{boundary} = \frac{\sum_{i=1}^{M} \frac{\mathcal{P}_i}{d_i}}{\sum_{i=1}^{M} \frac{1}{d_i}} \tag{C.2}$$

where $d_i = \sum_{i=1}^{M} (\mathbf{x}_{boundary} - \mathbf{x}_i)^2$ i.e. the shortest distance between the each of the selected nodes and the current boundary node.

References

[1] Jameson A., Schmidt W. & Turkel E. (1981). Numerical simulation of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. In *AIAA paper 81-1259, AIAA* 5^{th} *Computational Fluid Dynamics Conference*. xi, 3, 22, 31, 36, 73

[2] White F.M. (2006). *Viscous Fluid Flow*. McGraw-Hill International Editions, Singapore, 3rd edn. 2

[3] Aftosmis M.J., Berger M.J. & Melton J.E. (1998). Robust and efficient Cartesian mesh generation for component-based geometry. *AIAA Journal*, **36**, 952–960. 3, 15

[4] Ingram D.M., Causon D.M. & Mingham C.G. (2003). Developments in Cartesian cut cell methods. *Mathematics and Computers in Simulation*, **61**, 561–572. 3, 14, 17, 21, 22

[5] Brandt A. (1977). Multi-level adaptive solutions to boundary value problems. *Mathematics and Computation*, **21**, 333–390. 3, 40

[6] Lallemand M.H., Steve H. & Dervieux A. (1992). Unstructured mulitgridding by volume agglomeration: Current status. *Computers and Fluids*, **21**, 397–433. 3, 39, 42, 48, 50

[7] Clarke D.K., Salas M.D. & Hassan H.A. (1986). Euler calculations for multi-element airfoils using Cartesian grids. *AIAA Journal*, **24**, 353–358. 7, 11, 21

[8] Coirier W.J. & Powell K.G. (1995). An accuracy assessment of Cartesian-mesh approaches for the Euler equations. *Journal of Computational Physics*, **117**, 121–131. 7, 17, 21, 22, 38

[9] Ham F.E., Lien F.S. & Strong A.B. (2002). A Cartesian grid method with transient anisotropic adaptation. *Journal of Computational Physics*, **179**, 469–494. 7, 21, 38, 41, 42

[10] Murman S.M., Aftosmis M.J. & Berger M.J. (2002). Numerical simulation of rolling-airframes using a multi-level Cartesian method. In *AIAA*, vol. Paper 2002-2798, St. Louis. 7, 15, 16, 21

[11] Murman S.M., Aftosmis M.J. & Berger M.J. (2003). Simulations of 6-DOF motion with a Cartesian method. In *AIAA*, vol. Paper 2003-1246, Reno. 7, 17, 22

[12] Yang G., Causon M., Ingram D.M., Saunders R. & Batten P. (1997). A Cartesian cut cell method for compressible flows part A: Static body problems. *The Aeronautical Journal*, **101**, 47–56. 7, 15, 17, 21

[13] Yang G., Causon D.M. & Ingram D.M. (1999). Cartesian cut-cell method for axisymmetric separating body flows. *AIAA Journal*, **37**, 905–911. 7, 21, 22

[14] Malan A.G., Lewis R.W. & Nithiarasu P. (2002). An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part I. Theory and implementation. *International Journal for Numerical Methods in Engineering*, **54**, 695–714. 8, 9, 21, 29, 32

[15] Prandtl L. (1904). Über flüssigkeitsbewegung bei sehr kleiner reibung. In *Proc. Third Int. Math. Cong.*, Heidelberg. 11

[16] Baker T.J. (2003). Three decades of meshing; a retrospective view. In *AIAA Computational Fluid Dynamics Conference*, vol. Paper 2003-3563, Orlando, Florida. 14

[17] Jameson A. & Mavriplis D. (1986). Finite volume solution of the two-dimensional Euler equations on a regular trianguler mesh. *AIAA Journal*, **24**, 611–618. 15

[18] Mavriplis D.J. (1990). Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes. *AIAA Journal*, **28**, 213–221. 15, 31, 32, 33

[19] Mavriplis D.J. & Venkatakrishnan V. (1995). Agglomeration multigrid for two-dimensional viscous flows. *Computers and fluids*, **24**, 553–570. 15, 48, 50

[20] Sørensen K.A. (2002). *A Multigrid Acceleration Procedure for the Solution of Compressible Fluid Flows on Unstructured Hybrid Meshes*. Doctor of Philosophy Thesis submitted to the University of Wales Swansea, Swansea. 15, 37, 40, 42, 48, 61, 70

[21] Berger M., Aftosmis M. & Adomavicius G. (2000). Parallel multigrid Cartesian meshes with complex geometry. In *Proc. Parallel* CFD *Conference*, Holland. 15, 16, 21, 22

[22] Ogawa T. (1999). Development of a flow solver using the adaptive Cartesian mesh algorithm for wind environment assessment. *Journal of Wind Engineering and Industrial Aerodynamics*, **81**, 377–389. 15, 21, 38

[23] Wang Z.J. & Chen R.F. (2002). Anisotropic solution-adaptive Cartesian grid method for turbulent flow simulation. *AIAA Journal*, 1969–1978. 15, 21, 22, 38

[24] Popinet S. (2003). Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries. *Journal of Computational Physics*, **190**, 572–600. 16, 21, 38

[25] Yang G., Causon M., Ingram D.M., Saunders R. & Batten P. (1997). A Cartesian cut cell method for compressible flows part B: Moving body problems. *The Aeronautical Journal*, **101**, 57–65. 16

[26] Harpoon (2005). CEI, http://www.ceintl.com/products/harpoon.html, version 1.4.0 a_2 edn. 16, 80

[27] Gambit (2004). Fluent Inc., http://www.fluent.com/software/gambit/index.htm, version 2.1 edn. 16

[28] Aftosmis M.J., Berger M.J. & Adomavicius G. (2000). A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries. In *AIAA*, vol. Paper 2000-0808, Reno. 17, 21, 22, 38, 41, 42, 70

[29] Koh E.P.C., Tsai H.M. & Liu F. (2005). Euler solution using cartesian grid with a gridless least-squares boundary treatment. *AIAA Journal*, **43**, 246–255. 17

[30] Causon D.M., Ingram D.M., Mingham C.G., Yang G. & Pearson R.V. (2000). Calculation of shallow water flows using a Cartesian cut cell approach. *Advances in Water Resources*, **23**, 545–562. 21, 22

[31] Tucker P.G. & Pan Z. (2000). A Cartesian cut cell method for incompressible viscous flow. *Applied Mathematical Modelling*, **24**, 591–606. 21

[32] Ye T., Mittal R., Udaykumar H.S. & Shyy W. (1999). An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, **156**, 209–240. 21

[33] Zhang L.P. & Wang Z.J. (2004). A block LU-SGS implicit dual timestepping algorithm for hybrid dynamic meshes. *Computers & Fluids*, **33**, 891–916. 21, 22, 38

[34] French A. (2004). Solution of the euler equations on cartesian grids. *Applied Numerical Mathematics*, **49**, 367–379. 21

[35] Hirsch C. (1990). *Numerical Computation of Internal and External Flows, Computational Methods for Inviscid and Viscous Flows*, vol. 2. Wiley. 22

[36] Beam R.M. & Warming R.F. (1976). An implicit finite difference algorithm for hyperbolic systems in conservation law form - Application to

Eulerian gas-dynamics equations. *Journal of Computational Physics*, **22**, 87–110. 22

[37] MacCormack R.W. & Baldwin B.S. (1975). A numerical method for solving the Navier-Stokes equations with application to shock-boundary layer interactions. In *AIAA paper 1-75*. 22

[38] Vahdati M., Morgan K., Peraire J. & Hassan O. (1989). A cell-vertex upwind unstructured grid solution procedure for high-speed compressible viscous flow. In *International Conference on Hypersonic Aerodynamics*, 12.1–12.22, Royal Aeronautical Society, London. 23, 29, 36

[39] Crumpton P.I., Moinier P. & Giles M.B. (1997). An unstructured algorithm for high Reynolds number flows on highly stretched meshes. *Numerical Methods in Laminar and Turbulent Flow (ed. C. Taylor and J. T. Cross)*, 561–572. 29

[40] Sørensen K.A., Hassan O., Morgan K. & Weatherill N.P. (2002). Agglomerated multigrid on hybrid unstructured meshes for compressible flow. *International Journal for Numerical Methods in Fluids*, **40**, 593–603. 29, 31, 39

[41] Malan A.G. (2002). *Investigation into the Continuum Thermodynamic Modeling of Investment Casting Shell-Mould Drying*. Ph.D. thesis, University of Wales Swansea. 29, 40

[42] Swanson R.C. & Turkel E. (1992). On central-difference and upwind schemes. *Journal of Computational Physics*, **101**, 297–306. 31

[43] Lin F.B. & Sotiropoulos F. (1997). Assessment of artificial dissipation models for three-dimensional incompressible flow solutions. *Journal of Fluids Engineering*, **119**, 331–340. 31

[44] Watson D.F. (1992). *Contouring, A guide to the analysis and Display of Spatial Data*. Pergamon Press, New York. 34, 50, 85

[45] Murman S.M., Aftosmis M.J. & Berger M.J. (2003). Implicit approaches for moving boundaries in a 3-D Cartesian method. In *AIAA*, vol. Paper 2003-1119, Reno. 38

[46] Tannehill J.C., Anderson D.A. & Pletcher R.H. (1997). *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis, Washington, 2nd edn. 39

[47] Taflove A. & Brodwin M.E. (1975). Electromagnetic scattering problems. *IEEE Transactions on Microwave Theory and Techniques*, **23**, 623. 40

[48] Zienkiewicz O.C. & Taylor R.L. (2000). *The Finite Element Method: Volume 3 - Fluid Dynamics*. Butterworth-Heinemann, Oxford, 5th edn. 40

[49] Briggs W.L., Henson V.E. & McCormick S.F. (2000). *A Multigrid Tutorial*. Society for Applied and Industrial Mathematics, Philadelphia, 2nd edn. 40, 46, 48

[50] Charlton E.F. (1997). *An Octree Solution to Conservation-laws over Arbitrary Regions (OSCAR) with Applications to Aircraft Aerodynamics*. Doctor of Philosophy Thesis submitted to the University of Michigan, Michigan. 41, 42, 80

[51] Hannemann V. (2001). Structured multigrid agglomeration on a data structure for unstructured meshes. In *Numerical Methods for Fluid Dynamics VII*, 329–337, Oxford. 42

[52] Mavriplis D.J. & Venkatakrishnan V. (1996). A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes. *International Journal for Numerical Methods in Fluids*, **23**, 527–544. 42

[53] Blazek J. (2001). *Computational Fluid Dynamics: Principles and Applications*. Elsevier Science, Oxford, 1st edn. 48, 50

[54] Wesseling P. (1992). *An Introduction to Multigrid Methods*. John Wiley and Sons. 49

[55] Roache P. (1998). *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, New Mexico. 53, 77

[56] Belov A., Martinelli L. & Jameson A. (1995). A new implicit algorithm with multigrid for unsteady incompressible flow calculations. In *AIAA Paper*, vol. 95-0049. 54

[57] Slater J.W. (2005). *Tutorial on CFD Verification and Validation*. NPARC Alliance, http://www.grc.nasa.gov/WWW/wind/valid/tutorial/tutorial.html. 75

[58] AIAA (1998). Guide for the verification and validation of computational fluid dynamics simulations. Tech. Rep. AIAA G-077-1998, AIAA. 75

[59] Burden R. & Faires J. (2001). *Numerical Analysis*. Brooks/Cole, seventh edn. 77, 85

[60] Aftosmis M.J. (1997). Solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries. In *28th Computational fluid Dynamics*, vol. Lecture Series 1997-02, von Karman Institute for Fluid Dynamics. 80

[61] OpenCASCADE (2005). http://www.opencascade.org/, version 5.2 edn. 80