UNIVERSITY OF SOUTHAMPTON

AUTOMATED ADAPTATION OF SPATIAL GRIDS FOR FLOW SOLUTIONS AROUND MARINE BODIES OF COMPLEX GEOMETRY

by Alexander Mitchell Wright

Thesis submitted in candidature for
the degree of Doctor of Philosophy

Ship Science
School of Engineering Sciences
Faculty of Engineering and Applied Science
December 2000

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF APPLIED SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING SCIENCES, SHIP SCIENCE

Doctor of Philosophy

AUTOMATED ADAPTATION OF SPATIAL GRIDS FOR FLOW SOLUTIONS AROUND MARINE BODIES OF COMPLEX GEOMETRY

by Alexander Mitchell Wright

Marine bodies have complex geometry and are subject to various physical phenomenon, such as the free surface and turbulence. These require a variety of cell topologies which, when combined, often restrict the geometrical discretisation of the domain. The approach taken has been to use control volumes of arbitrary definition to allow the benefits of mesh adaptivity as well as a mix of cell topologies.

Three dimensional unstructured meshes are constructed using a data structure that lowers memory requirements from previous methodology and allows efficient and complex mesh adaption. A three dimensional Euler flow solver has been developed that utilises this mesh definition with artificial compressibility. The generalised fluxes for arbitrarily shaped moving meshes have been mathematically derived for a flux vector splitting scheme. The solver operates upon a face structure using a first order upwinding spatial discretisation. Validation has shown that results are convergent. Case studies have included flow over a arc hump, a foil at an angle of attack and a truncated structure.

Meshes have been assessed with respect to both geometrical and flow solution properties, and numerical quantification of the parameters relevant to the solution accuracy has been defined. Facial skew is the most influential geometrical property while the local pressure gradient has been used to assess local solution accuracy. Quality quantification parameters have been used to control a variety of mesh adaptation techniques. Control volume splitting, merging and vertex manipulation have been used to improve solution accuracy with respect to computational efficiency. In addition, the cellular splitting algorithm has been used to create a multigrid scheme that increases the computational efficiency still further.

Acknowledgments

In order to accomplish this work I have had to rely upon a number of people to whom I am truly grateful. Firstly I would like to thank the members of staff (past and present) in the Department of Ship Science and the School of Engineering Sciences in general. Particular thanks must go to my supervisor, Dr. Steve Turnock, whose enthusiasm, patience and overall guidance has been greatly appreciated; but please Steve, no more "really good ideas"!

Secondly I would like to thank my friends, both in Southampton and elsewhere, who have kept me sane with the occasional visit to the pub and have remained patient when I have said "just one more month" for the fifth month in a row.

Most importantly, I would like to state my eternal appreciation of my parents for their never ending support and belief in me as a person. Thank you; without your help I would not be in the privileged position that I am now.

Finally, thank you Sarah for putting up with a grumpy old person for a very long time. Your patience and support have always been there for me at the right time.

Right. Now I'm off to try and re-find a life.

Contents

Nomenclature xx

1 Introduction **1**

1.1 Aims and Objectives 1

1.2 Background 2

1.2.1 Marine vehicles 2

1.2.2 Development of fluid mechanics 4

1.2.3 Computational modelling of fluid flow 7

1.3 Layout of Thesis 11

2 Grid and Data Structure **13**

2.1 Introduction 13

2.2 Geometrical Entities 13

2.3 Mesh Generation 15

2.3.1 Structured, unstructured and hybrid meshes 16

2.3.2 Cell centred and cell vertex schemes 22

2.4 Data Storage Approach 24

2.5 Data Structure Implemented 28
2.5.1 Arbitrary cell topology 28
2.5.2 Hybrid memory structure 34
2.5.3 Entity fission and fusion 35
2.5.4 Surface definition and tracking 35
2.6 Summary 38

3 Finite Volume Flow Solver **40**
3.1 Introduction 40
3.2 Euler Equations 40
3.3 Finite Volume Discretisation 43
3.4 Incompressible Flows 45
3.4.1 Artificial compressibility 46
3.5 Summary 52

4 Numerical Implementation **53**
4.1 Time Integration 53
4.1.1 Runge Kutta time stepping 54
4.1.2 Pseudo-time 55
4.2 Spatial Discretisation 57

4.2.1 Flux vector splitting 58

4.2.2 First order discretisation 61

4.3 Validation 62

4.3.1 Flow over a circular arc hump 63

4.3.2 NACA 0012 foil 73

4.4 Summary 80

5 Grid Quality Assessment **81**

5.1 Introduction 81

5.2 Error Analysis of Numerical Solution 82

5.2.1 Error definition 85

5.2.2 Error distribution for current scheme 85

5.3 Properties of Influence 88

5.3.1 Facial properties 89

5.3.2 Cellular properties 92

5.3.3 Flow solution properties 94

5.4 Numerical study of flow solver 96

5.4.1 Error inducement 96

5.4.2 Error analysis 97

5.5 Validation of quality assessment 99

5.5.1 Flow over 10% hump 99

5.5.2 Flow around 2D foil 108

5.6 Discussion of numerical results 114

5.7 Summary 116

6 Mesh adaption **118**

6.1 Introduction 118

6.1.1 Importance of mesh adaption 118

6.1.2 Methods and control of adaption 119

6.2 Literature review 120

6.2.1 Control of adaption 120

6.2.2 Method of adaption 123

6.3 Control volume splitting 124

6.3.1 Geometrical Entity Based Approach 125

6.3.2 Geometrical Position Based Approach 129

6.3.3 Benefits of position based approach 132

6.4 Control volume merging 133

6.4.1 Initial method of merging control volumes 133

6.4.2 Influences and effects 134

6.5 Entity stretching and convection 135

6.6 Validation and verification
of adaptation algorithms 138

6.6.1 Arbitrary splitting of volumes 139

6.6.2 Geometrical entity convection and manipulation 144

6.7 Summary 153

7 Validation test cases **155**

7.1 SPAR buoy 155

7.1.1 Model definition 155

7.1.2 Results 157

7.1.3 Discussion of results 163

7.2 Summary 165

8 Conclusions and further work **166**

8.1 Additional Observations 167

8.2 Further work 168

References **173**

A Forms of the Euler Equations **183**

A.1 Conservative Form of Euler Equations 183

A.2 Non-Conservative Form of Euler Equations 186

A.3 Quasi-linear form of Euler equations 187

A.4 Characteristic Form of Euler Equations 188

A.5 Transformation Between Forms 190

B Derivation of Characteristics of Euler Equations **193**

B.1 Artificial Compressibility Formulation 194

B.2 Moving Mesh Formulation 202

C Least Squares Theory **206**

D Multigridding **208**

D.1 Theory 208

D.1.1 Von Neumann error analysis 208

D.1.2 Stability and Efficiency 210

D.1.3 Two grid and full multigrid models 211

D.1.4 Restrictor and Prolongator Operators 213

D.1.5 Multigrid cycle 214

D.1.6 Convergence 215

D.2 Implementation 216

D.3 Results 218

D.4 Discussion 221

D.5 Summary 224

List of Figures

1.1 Levels of abstraction from reality to computational model 8

2.1 The basic geometrical entities 14

2.2 Physical and computational domains for structured grid topology 17

2.3 Grid structure for seven block DTMB Model 5415 grid 18

2.4 Unstructured grid for DTMB Model 5415 grid 19

2.5 A surface and volume hybrid mesh for a turbine blade 22

2.6 CHIMERA mesh around NASA Space Shuttle geometry 23

2.7 A graphical representation of a stack and a list 24

2.8 An example of a simple hash table 26

2.9 The representation of a simple tree structure 27

2.10 Definition of a hanging node 29

2.11 Cellular and control volume based meshes 29

2.12 Control volume face definition 31

2.13 Polygon control volume 31

2.14 Connectivity of geometrical mesh 32

2.15 Data structure of geometrical mesh 36

2.16 Tree data structure of flow solution nodes 37

2.17 Typical results from a modern surface capturing free surface code 38

3.1 Arbitrary Volume 43

3.2 Discrete Control Volume 44

3.3 Influence of freestream Mach number on rectangular wing solution time 45

4.1 Convergence history for incompressible flow over a hump in a channel using $\omega = 0.5, 0.6, 0.7, 0.8, 1.0$ and 1.1 56

4.2 Convergence history for incompressible flow over a hump in a channel using global and local timestepping. 56

4.3 Wire frame of hump mesh 64

4.4 Hump meshes utilised to test effects of accuracy, stability and convergence 64

4.5 Convergence history for 38×10 mesh on 10% arc hump 67

4.6 Convergence history for 268×67 mesh on 10% arc hump 67

4.7 Convergence history for all meshes on 10% arc hump 68

4.8 Pressure coefficient upon 10% arc hump surface for 38×10 mesh 69

4.9 Pressure coefficient upon 10% arc hump surface for 268×67 mesh 69

4.10 Pressure coefficient upon 10% arc hump surface for all meshes . . 70

4.11 Comparison of finest mesh 10% arc hump flow with potential flow code 70

4.12 Streamwise velocity variation over 10% arc hump for 38 × 10 mesh 71

4.13 Streamwise velocity variation over 10% arc hump for 268 × 67 mesh 71

4.14 Wire frame of foil domain 73

4.15 123 × 28 and 593 × 33 meshes around NACA 0012 foil 74

4.16 Convergence history for coarse mesh around NACA0012 foil at 4 degrees angle of attack 77

4.17 Convergence history for all meshes around NACA0012 foil at 4 degrees angle of attack 77

4.18 Pressure coefficient for NACA0012 foil at 4 degrees angle of attack, for all meshes 78

4.19 Pressure coefficient for NACA0012 foil at 4 degrees angle of attack, compared to experimental data 78

4.20 Pressure contours around NACA0012 foil at 4 degrees angle of attack for densest mesh 79

5.1 Division of error for mesh size variation 83

5.2 Face definition in local frame of reference 86

5.3 Wave-like solution passing across interfaces 90

5.4 Types of facial distortion induced 97

5.5 Error in force for 10% arc hump with mesh density variation . . . 101

5.6 Variation in average local error with mesh discretisation for 10% arc hump 101

5.7 Error in X direction force for 10% arc hump with mesh density and quality variation 102

5.8 Error in Z direction force for 10% arc hump with mesh density and quality variation 102

5.9 Distribution of local error for 58 × 15 mesh over 10% arc hump . . 103

5.10 Distribution in local error for 58 × 15 mesh with distortion 103

5.11 Variation in differential of flow properties (P, u, v respectively) over 10% arc hump 104

5.12 Variation in solution error with facial aspect ratio for all distorted meshes 105

5.13 Variation in solution error with facial skew for all distorted meshes 105

5.14 Variation in solution error with facial deviation from planarity for all distorted meshes 106

5.15 Variation in solution error with misalignment of facial normal for all distorted meshes 106

5.16 Relative importance of geometrical properties for all distorted mesh solutions 107

5.17 Error in force for NACA 0012 foil at 4° with mesh density variation 109

5.18 Variation in average local error with mesh discretisation for NACA 0012 foil 109

5.19 Distribution of local error for 58 × 15 mesh over NACA 0012 foil . 110

5.20 Distribution in local error for 58 × 15 mesh with distortion 110

5.21 Variation in differential of flow properties (P, u, v respectively) over NACA 0012 foil 111

5.22 Variation in solution error with facial aspect ratio for distorted meshes 112

5.23 Variation in solution error with facial skew for distorted meshes . 112

5.24 Variation in solution error with facial deviation from planarity for distorted meshes 113

5.25 Variation in solution error with misalignment of facial normal for distorted meshes 113

6.1 Cellular description used to define mesh quality 122

6.2 Volumes with unacceptable quality in a mesh around a Wigley hullform 128

6.3 Adapted areas of Wigley hullform mesh 128

6.4 Volumes with extreme quality in the Wigley hullform mesh after adaptation 128

6.5 Node and face definition 136

6.6 Typical hexahedral control volume 139

6.7 Children of volume with splitting plane normal (1, 0, 0) 139

6.8 Children of volume with splitting plane normal(0.577, 0.816, 0) . . 140

6.9 Children of volume with splitting plane normal(0.707, 0.707, 0) . . 140

6.10 Children control volumes with new IDs 142

6.11 Parent control volume and 4th generation child 142

6.12 Convergence history for adapted 58 × 15 mesh on 10% arc hump . 143

6.13 Pressure coefficient on 10% arc hump for original and adapted mesh 143

6.14 Total grid quality of 58 × 15 mesh 146

6.15 Total grid quality of 58 × 15 mesh with mesh distortion 146

6.16 Total grid quality of distorted 58 × 15 mesh after adaption 146

6.17 Areas of distorted 58 × 15 mesh to which Algorithms 6.5 and 6.7 were applied 147

6.18 Comparison of forces generated by the hump with the original mesh, the distorted mesh and with the adapted distorted mesh . . 148

6.19 Comparison of forces generated by foil with the original mesh, the distorted mesh and the two adaptions of the distorted mesh. . . . 149

6.20 Geometrical grid quality for original 343 × 25 mesh 150

6.21 Geometrical grid quality for 343 × 25 mesh after distortion 150

6.22 Geometrical grid quality for distorted mesh after adaption via Algorithm 6.4 151

6.23 Geometrical grid quality for distorted mesh after adaption via Algorithms 6.4, 6.5 and 6.7 151

6.24 Distribution of error through the domain for the mesh adapted with Algorithm 6.4, and with all algorithms respectively. 152

7.1 Wire frame of domain around truncated square section prism . . . 155

7.2 Original meshes surrounding structure in the XY and XZ planes . 156

7.3 Control volumes created during solution of flow 158

7.4 Volumes of domain space with pressure differential values over 0.1 and 0.25 for unadapted and adapted meshes respectively 159

7.5 Pressure coefficient around structure for all meshes at base and tip respectively 160

7.6 Streamwise velocity at three vertical positions for $42 \times 36 \times 24$ and adapted $26 \times 26 \times 16$ meshes respectively 161

7.7 Vertical velocity profile for $42 \times 36 \times 24$ and adapted $26 \times 26 \times 16$ meshes respectively 162

7.8 Convergence history for unadapted and adapted flow solutions . . 163

8.1 Incorporation of more intuitive selection process in splitting algorithm 171

A.1 Arbitrary volume, containing sources and surface fluxes 184

A.2 Characteristic surface 189

A.3 Zones of dependence and influence 189

A.4 Relation between the conservative, primitive and characteristic variables 192

D.1 Two grid multigrid coupling 213

D.2 The two basic types of multigrid cycle possible 215

D.3 Wire frame of domain and node - node connectivity on initial coarse mesh 219

D.4 Coarse and fine meshes, showing the original control volumes and the children bearing the same global ID 219

D.5 Convergence history for channel initially at rest with different solution strategies. 220

D.6 Coarse mesh control volume definition with fine mesh faces 222

List of Tables

4.1 Node distributions for meshes of flow over a 10% hump 63

4.2 CPU time usage for meshes over 10% arc hump 66

4.3 Forces imposed upon 10% arc hump for all meshes 66

4.4 Node distributions for meshes of flow over a NACA 0012 foil . . . 75

4.5 Artificial compressibility factors for a domain with a stagnation point 75

4.6 Forces imposed upon NACA 0012 foil at 4° angle of attack for all meshes 76

5.1 Break down of discretisation and distortion errors 87

5.2 CPU time usage for mesh quality analysis, in comparison to flow solver CPU time 116

6.1 Grid quality of Wigley hullform mesh before and after adaptation 127

6.2 CPU time usage for adapted mesh over 10% arc hump 144

6.3 Forces imposed upon 10% arc hump for adapted and original meshes 144

6.4 Weighting of geometrical parameters for quality analysis of meshes 145

6.5 Statistical definition of geometrical quality in 10% arc hump meshes 147

6.6 CPU times for different adaption techniques for NACA 0012 foil mesh 153

7.1 CPU time usage for different meshes 158

D.1 CPU time usage per iteration for different solution strategies . . 221

D.2 Memory requirements for different meshes 221

D.3 Required total CPU time for different solution strategies 223

Nomenclature

a	convection speed
$\vec{A}$	Jacobian of flux vector with respect to conservative variables with components A, B, C
$\mathcal{B}$	artificial bulk viscosity factor
c	speed of sound
e	internal energy per unit mass
E	total energy per unit mass
f^*	numerical flux
$\vec{f_e}$	external force vector
$\vec{\mathbf{F}}$	scalar flux vector with components **f**, **g**, **h**
$\mathcal{F}n$	non dimensional Froude number
g	gravity acceleration
G	Amplification factor for Von Neumann stability analysis
h	enthalpy per unit mass
H	stagnation enthalpy per unit mass
I	$\sqrt{-1}$
$\vec{k}$	wave direction
K	projection of Jacobian matrix on propagation direction $\vec{k}$
$\mathcal{K}$	artificial compressibility factor
l	Representative length of body
L	Transformation matrix between characteristic and primitive variables
M	Transformation matrix between conservative and primitive variables
$\vec{n}$	normal vector
p	static pressure
P	Transformation matrix between characteristic and conservative variables
$\mathcal{P}$	preconditioning matrix for artificial compressibility
Q	source term vector
$\mathcal{R}e$	non dimensional Reynolds number
s	entropy per unit mass
t	time
T	temperature
U	conservative state vector
$\vec{v}$	velocity with Cartesian components u, v, w
V	volume
V	primitive variables vector
w_f	work done by external force
W	characteristic variables vector
x, y, z	Cartesian co-ordinates

α	Runge-Kutta time integration coefficient
β	Constants of proportionality for geometrical grid quality
γ	specific heat ratio
λ	eigenvalue
Λ	eigenvector
μ	viscosity
ν	kinematic viscosity
ξ, η, ζ	curvilinear co-ordinates
ρ	density
τ	pseudo time
ψ	piezometric pressure (incorporating hydrostatic component)

Subscripts

A	artificial compressibility value
cv	contravarient value
g	grid/mesh value
m	moving mesh value
ND	non-dimensional value
t	partial derivative with respect to time
x, y, z	components in the x, y, z direction
x, y, z	partial derivatives with respect to x, y, z
∞	free stream condition
ξ, η, ζ	partial derivatives with respect to ξ, η, ζ

Superscripts

m	real time step number
n	pseudo-time step number
T	transposed (vector or matrix)

Symbols

$\frac{D}{Dt}$	substantial/total derivative
$\frac{\partial}{\partial t}$	partial derivative
∇	grad operator $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$
Δ	finite step

1 Introduction

1.1 Aims and Objectives

Progress in the ability to model dynamic ship hull and offshore structure performance using computational fluid dynamics (CFD) is presently restricted by a number of key aspects. These include the presence of incompressibility, turbulence and time dependence. The problem is further exacerbated by the presence of the free surface which imposes severe computational requirements in processor time and memory usage.

The stringency of this requirement restricts the computational mesh density, particularly in structured meshes, and hence the achievable accuracy for a given amount of computational effort. In addition, the complexity of the surface curvature of a realistic ship hull and the deforming free surface makes high quality mesh generation difficult. The flow over a moving hull, with turbulence, strong flow rotationality and separation often results in poor performance of even the most sophisticated turbulence models, numerical schemes and mathematical models. These effects are compounded by the complex shapes of realistic structures that result in low quality numerical meshes. The result of such poor meshes is to lower the accuracy and stability of a given numerical solution.

The initial aim is therefore to investigate methods of mesh adaptation and their use to maximise mesh efficiency as well as automatically controlling mesh definition. The second aim of this work is to analyse spatial mesh quality in a quantitative manner and determine its effect upon solution accuracy.

The objectives to attain these aims are

- Develop a description of the physical domain which does not place limits upon geometrical topology or mesh structure

- Implement an accurate numerical method for the solution of the unsteady, incompressible Euler equations, capable of operating upon an arbitrary cell topology
- Define suitable grid quality measures in a statistically quantifiable manner
- Develop methods of adapting an arbitrary topology mesh and implementing said methods in conjunction with grid quality measures
- Validate the approach by exploiting the unstructured and arbitrary mesh environment to increase solution accuracy while maintaining computational efficiency
- Incorporate the above achievements in a framework capable of development towards accurate free surface modelling

1.2 Background

The calculation of the hydrodynamic forces imposed upon a marine vehicle or structure is still a topic of research, even after centuries of study. To fully understand the problem the defining aspects of fluid flows and the marine environment must be understood. In addition to this knowledge, a recognition of previous developments in the attempt for greater and more accurate prediction is also desirable.

1.2.1 Marine vehicles

Marine vehicles operate in a unique environment that provides challenging conditions to be modelled. They operate at the interface of two mediums of greatly different density. Thus free surface waves can be radiated, diffracted and propagated. With the density difference there are also compressibility changes; air is compressible while water is incompressible. The complexity of the environment is enhanced with the inclusion of viscosity and the presence of high flow rotationality.

Viscous effects are of particular interest in accurately modelling flow around

the stern of ships and in their wake where separation is likely to occur [1, 2, 3], as well as in steep/breaking waves.

Rotationality is of particular interest to sailing vessels, where the ship hull is often at a small angle of yaw, thus producing lift. While potential flow panel methods can simulate lifting bodies by the attachment of a wake sheet to trailing edges, this approach cannot be used in such cases [4].

In addition to these ambient environmental conditions there are the time dependent external influences of water currents and waves as well as wind. Water currents can have a considerable influence upon the sea spectrum obtained in certain conditions; it changes the wave number, frequency and amplitude as well as providing non zero mean values [5].

The accurate modelling of the water-air interface in isolation is a challenge, and when a man-made structure is added another set of difficulties is added.

These bodies are free to move in all six degrees of freedom; three translational and three rotational, all of which have the ability to cross-couple. Thus any model that aims to predict vessel responses has to take into consideration such non-linear terms.

Of greater interest to the work undertaken here is the geometric shape of the vessel. A basic hull form can be described as a ellipsoid of mathematical form (Wigley hull), and most 'stationary' offshore structures defined by truncated cylinders and hexahedra. These are however not realistic configurations by themselves. Offshore structures are made of many individual components, connected in a complex three dimensional configuration. Ship hulls regularly have complex two dimensional curvature as well as numerous appendages, such as bilge keels, rudders, shafting, bosses and propulsion devices.

Modelling of the general marine environment is one with higher emphasis upon the mathematical model used, while the prediction of flow around realistic marine vehicles has greater dependence upon the computational discretisation of the domain continuum.

The use of CFD within the marine industry has been slow to commence, but

is now being used in a wider and wider range of applications, from vortex induced vibration (VIV) modelling of deep sea risers to predicting the wave-making drag and sail performance of racing yachts [4, 6] .

1.2.2 Development of fluid mechanics

There are three main methods of calculating such forces; experimental testing, theoretical calculations and computational modelling.

The foundations of fluid mechanics were created by Newton with the publication of the second volume of his *Principa* in 1687. In this work he established the relationships between force, momentum and acceleration on a rational basis. The theory promoted by Newton to calculate the hydrodynamic force upon a surface was his sine squared law which was used widely by naval architects due to its ease of use.

In the eighteenth century the discrepancies of this theory were noted, by d'Alembert in 1777, and also in 1781 by Euler. Euler progressed with a formula for resistance that attempted to take account of shear stress distribution, noting that his results were in agreement with d'Alembert.

It was in the eighteenth century that fluid mechanics really developed through the analysis of d'Alembert, Euler and Bernoulli. Bernoullis' importance to fluid dynamics is based upon his 1837 publication, *Hydrodynamica* (which incidentally is the first use of the term "hydrodynamics"). The initial form of the now common place Bernoulli's equation relating velocity and pressure is detailed here, but in a somewhat obscure format . It was Euler, a pupil of Bernoulli, that conceived of pressure as a point property, which led in turn to Euler's equation. These three men detailed what is now known as potential flow: inviscid and incompressible (with irrotationality unknowingly assumed). This led to d'Alemberts paradox of flow around a two dimensional cylinder producing zero drag. [7]

It was not until the nineteenth century that the importance of viscosity in the prediction of drag was fully realised. Viscosity is "the molecular resistance which fluid particles exhibit against displacement in relation to each other and with respect to the surface of solid objects " [8]. In 1845 an Englishman, G.

Stokes presented a hypothesis relating the viscosity coefficient and the bulk viscosity coefficient of a fluid, which gives values for shear stresses produced within fluid flow. Stokes and the Frenchman, Navier, independently derived the scalar equations defining momentum transfer by a fluid, incorporating viscosity, compressibility and the time dependant nature of fluid flow that are currently in use today - the Navier Stokes equations (Equation 1.1-a to 1.1-c). [9]

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho uu}{\partial x} + \frac{\partial \rho uv}{\partial y} + \frac{\partial \rho uw}{\partial z} = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + \frac{1}{3}\mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial x \partial y} + \frac{\partial^2 w}{\partial x \partial z}\right) \quad (1.1\text{-a})$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial \rho vu}{\partial x} + \frac{\partial \rho vv}{\partial y} + \frac{\partial \rho vw}{\partial z} = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + \frac{1}{3}\mu\left(\frac{\partial^2 u}{\partial y \partial x} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 w}{\partial y \partial z}\right) \quad (1.1\text{-b})$$

$$\frac{\partial \rho w}{\partial t} + \frac{\partial \rho wu}{\partial x} + \frac{\partial \rho wv}{\partial y} + \frac{\partial \rho ww}{\partial z} = -\frac{\partial p}{\partial z} + \mu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + \frac{1}{3}\mu\left(\frac{\partial^2 u}{\partial z \partial x} + \frac{\partial^2 v}{\partial z \partial y} + \frac{\partial^2 w}{\partial z^2}\right) \quad (1.1\text{-c})$$

Other notable developments in the field of fluid mechanics within the nineteenth century were by Reynolds and Froude. In 1883 Reynolds determined the point at which fluid flow changes from laminar to turbulent flow - the *critical* Reynolds number - and the effect this has upon the overall drag produced. He went on to separate the flow of fluid into mean and rapid fluctuations in his 1895 paper, which led to the concept of Reynolds stresses used today for turbulence modelling. The basic scaling law detailed in Equation 1.2 relating inertial and viscous forces is named after Reynolds, thus the term "Reynolds number" [10, 11, 12].

$$\mathcal{R}e = \frac{|\vec{v}|l}{\nu} \quad (1.2)$$

William Froude developed the free surface scaling law which is the ratio between dynamic and static forces (lift or frictional drag against gravity or weight)

as noted in Equation 1.3. He achieved this through experimental testing, towing wooden planks upon the surface of water [13, 14].

$$\mathcal{F}n = \frac{|\vec{v}|}{\sqrt{gl}} \tag{1.3}$$

It is important to note that in the late nineteenth and early twentieth centuries the major developments in fluid mechanics, such as that carried out by Reynolds and Froude, were accomplished via experimental testing and then transformed into theory, due to the still highly complex (and largely unknown) nature of the effect of viscosity at the time. The most pertinent point to be derived from the work of Reynolds and Froude is that for objects moving on the interface between water and air, the scaling similarity laws determine that in practice it is not possible for a vessel and its scale model to have the same Reynolds number *and* the same Froude number. Wind tunnel testing attempts to model objects at the same $\mathcal{R}e$, while towing tank tests model at $\mathcal{F}n$. Thus any scale model testing will require an empirical conversion factor, due to the incorrect $\mathcal{R}e$.

In the last century the major theoretical advance was the discovery and definition of the boundary layer. As well as developing lifting-line theory and other important theories upon which modern fluid dynamics are based, Prandtl allowed practical engineering solutions to fluid dynamic problems after his publication of 1904 [7] detailing the concept of the boundary layer. It was not until 1908 that Blasius' equation allowed the first practical use of the theory. The thin (in comparison with the body) boundary layer has a marked effect upon the viscous drag of the body, dictating separation, momentum losses and velocity profiles. Outside the boundary layer the flow velocity is mostly unaffected by viscous shearing effects.

The successful modelling of fluid dynamics requires the knowledge of mass, momentum and energy transfer as well as the inclusion of the physical effects of viscosity, compressibility and rotationality, as detailed by the Navier Stokes equations 1.1-a to 1.1-c. These are however a coupled system of non-linear partial differential equations, and are extremely difficult to solve by an analytical method with current techniques except for a limited number of special cases. The work of Reynolds and Froude allowed the use of scaled data to model accurately a full

sized body, but discovered also that marine vehicles only have the same viscous to inertia forces and dynamic to static forces at one specific body size - i.e. only a full scale model will have the correct viscous *and* wave resistance. The work of Prandtl and others allowed the removal of viscosity from the majority of the domain for practical purposes, and highlighted its importance within the boundary layer next to the body's surface.

This gradual development of theoretical fluid dynamics since the eighteenth century has resulted in the study and practice of fluid dynamics being a 'two approach' operation in this century, involving theory and experimentation. The introduction of the digital computer, combined with numerical algorithms for the solution of physical problems, has introduced a fundamentally new approach - *computational fluid dynamics* (CFD).

1.2.3 Computational modelling of fluid flow

CFD can be used for a number of different purposes, which can be broken down into two main areas; research and design. As a research tool it can be used to carry out numerical experiments, to help interpret physical experimental results and to help understand highly complex flows. As a design tool it can be used to predict loadings and motions as well as aiding detailed and parametric design studies.

The definition of a computational approach requires several steps leading from an initial mathematical model to a final numerical solution, as shown in Figure 1.1.

The first step is the selection of a level of approximation to the physical problem to be solved, dependent upon the accuracy required as much as the computational power available. This includes the decision to include viscosity, rotationality, compressibility, heat transfer and so on. The second step is the choice of discretisation method of the mathematical model and involves the spatial discretisation and the equation discretisation. The decisions involved with defining the mathematical model are detailed in Chapter 3 , but due to the developmental nature of this work it was felt that viscous effects should not be considered. This is due to the complexity required to model such flow charac-

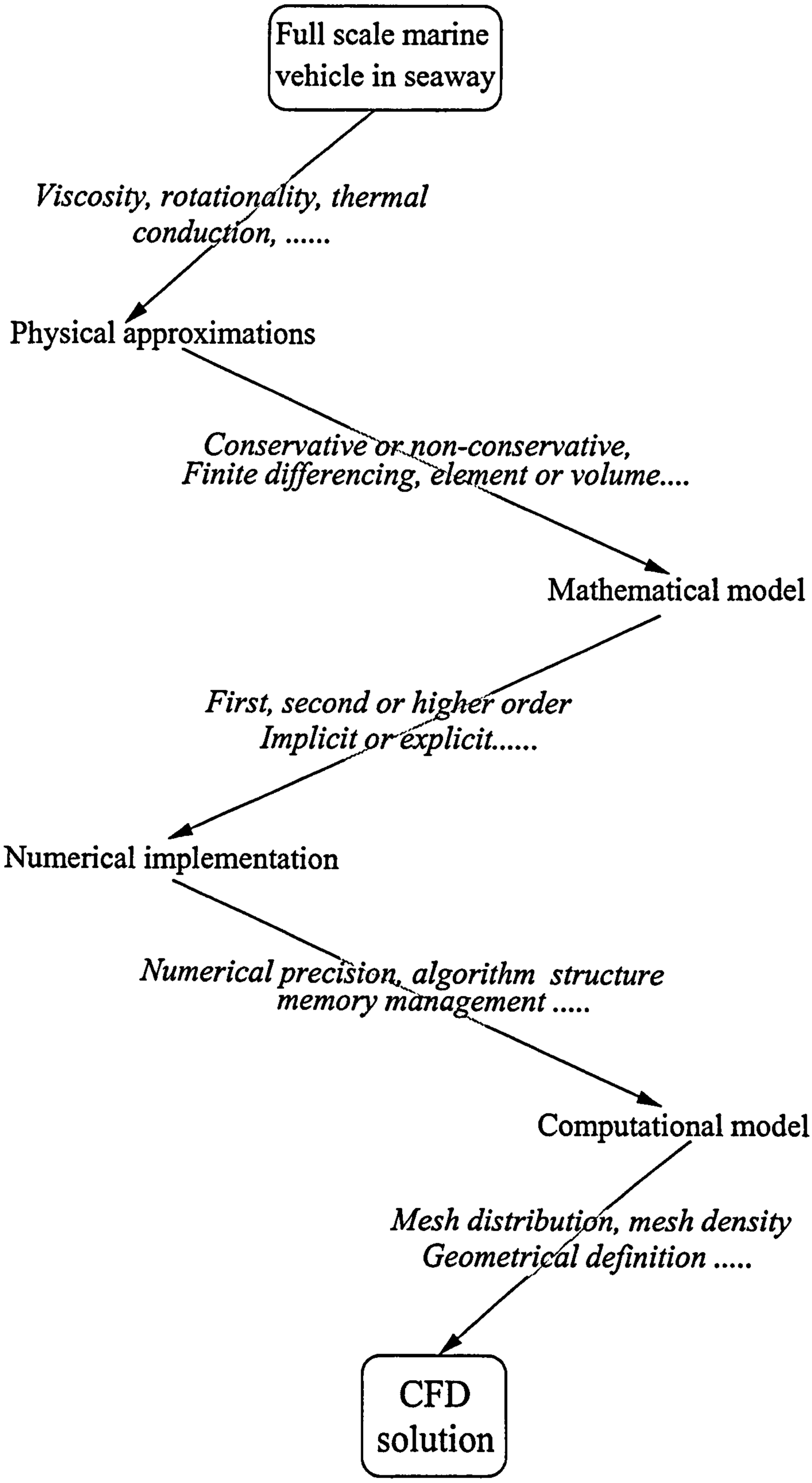


Figure 1.1: Levels of abstraction from reality to computational model

teristic in conjunction with the extra depth of knowledge required to create the relevant numerical schemes [15] as well as the added problems of modelling at full scale Reynolds number and the extra computational power required for such calculations. Presently realistic viscous flow calculations are achieved via a Reynolds Averaged Navier Stokes (RANS) code, where there are numerous difficulties in the construction and choice of turbulence model [16], or Large Eddy Simulations (LES) where the computational requirements for a relatively simple flow are in the order of 1×10^7 degrees of freedom which uses approximately 12 000 CPU hours on a Cray T3E super computer for a $\mathcal{R}e$ of 1×10^6 [17]. This is mainly due to the requirement that the problem has to be treated as temporally unsteady.

The splitting of the physical domain into discrete elements that model the mathematical behaviour of the fluid is a subject that has gained in importance as the problems being tackled have increased in complexity. The type of grid chosen can make or break the numerical solution. Because of this grid generation is a very active research activity in itself [18, Section 5],[19]. The different types of meshing available are described in Chapter 2, but to date there has been no clear winner. Each grid strategy has its own advantages and disadvantages, being particularly suited to a certain type of flow or geometrical configuration.

There are three basic types of numerical schemes used to discretise the mathematical operators; finite difference, finite element and finite volume. The finite difference method is based on the properties of Taylor Series expansions and is the easiest method to apply. It is however dependent on mesh points being positioned on families of non-intersecting lines [9], and hence places limits upon the physical discretisation topology. The finite element method applies a set flow field shape function to each discrete element. The elements can be of arbitrary size and shape and the technique relies heavily upon transformation between co-ordinate systems and numerical integration. It is however dependent upon the mesh having a generic topology. The finite volume method solves the integral form of the governing equations and can take full advantage of an arbitrary mesh, being unrestricted in mesh topology [15]. It is for this reason that the finite volume method has been chosen for this work, allowing the flow solver to be developed independent of the discrete meshes used.

In addition to the overall type of numerical scheme chosen the form of the numerical equations needs to be defined, choosing between time dependence or

time independence, implicit or explicit, and the order of accuracy. All of these topics will be covered in more detail in the relevant Chapters, but earlier work (Rycroft [20]) states that the application of higher order upwind schemes on arbitrary meshes is a non-trivial task and needs to be studied in depth to establish a robust accurate scheme. The final level of approximation used to obtain a computational solution is that of the computational algorithm and the manner of use of the computer resources - i.e. memory management, hardware architecture etc.

Since the introduction of CFD technology it has been realised that the solution to a given problem is highly dependent upon the discrete mesh that the flow solver uses. More and more resources have been allocated to the production of high quality meshes with minimal user effort [21, 22]. The problem is further confused by the fact that the user has only their prior experience and knowledge to define the specific areas of complex flow before solution, so the mesh may not be optimal. This is enhanced for unsteady problems with moving boundaries [23]. While a robust and accurate mathematical model is required for acceptable solutions, recent publications have strongly suggested that the key issues are in the overall framework of the solver and method of working with the physical domain shape and size [23, 24, 25, 26].

The problems to be considered in this work are of the flow of water around realistic marine structures. Therefore the flow will be viscous, incompressible and time dependent as well as in the vicinity of free surface waves. The structures and bodies being studied will consist of more than one basic geometrical shape, as well as singularities and surfaces of complex curvature and intersection. These bodies are also free to move in response to the imposed hydrostatic, hydrodynamic and aerodynamic forces. In response to these requirements the approach used in this work has been to create a mesh generation process and flow solver within a framework that can adapt and develop according to the solution requirements.

As stated previously in this chapter viscosity has been neglected, as have time dependent free surface flows, due to the developmental nature of the work. The flow solver created models incompressible rotational flow with the allowance for moving boundaries. The framework of the code has been created so that future developments can easily incorporate unsteady flow modelling and viscous effects. To model such flows efficiently the geometric mesh must be able to

model high curvature, geometrical singularities and a widely varying flow regime, both spatially and temporally. In the boundary layer the flow will be viscosity dominated, with high flow gradients, while the outer flow field will consist of low mass and momentum transfer. In addition there will be areas far from the body surface, such as the ship wake, that requires detailed viscous and geometrical modelling. The solution devised for coping with such conditions is one of arbitrary control volume shape and structure. This geometrical mesh is free to adapt to the local (and global) flow regime, thus producing a semi-automated and highly efficient meshing strategy. The adaption of the mesh is controlled by assessing the local mesh quality and flow characteristics and comparing them to those of the domain as a whole. The arbitrary control volume shape can accommodate geometrical singularities, moving boundaries and surface-surface intersections easily while the adaptive meshing will balance the variation of flow characteristics.

1.3 Layout of Thesis

This thesis details the work undertaken in the creation of a flow solver that can assess the relative quality of a localised mesh section, and adapt such an area to maximise the efficiency of the solver as a whole. The structure of the thesis follows the hierarchy of the solver as a whole, starting with the basic concepts of the geometrical entities and concluding with validated case studies of adapted solutions. As a result topics such as the numerical implementation of the flow solver itself have been split between several chapters.

Chapter 2 details the various methods of storing geometrical and flow solution data, and the method chosen for this work that most efficiently allows the storage of the required information. Chapter 3 describes the three dimensional flow solver, detailing the use of artificial compressibility and Chapter 4 describes the manner in which this theory has been discretised spatially and temporally.

The parameters by which accuracy and quality of both mesh and flow solution are assessed are presented in Chapter 5. The Chapter then details the manner in which they have been utilised in the current work. Chapter 6 continues this key section of the work by detailing the various methods of mesh

adaptation that have been implemented to counter the mesh deficiencies highlighted. Following this, Chapter 7 describes the validation case used to prove the increased efficiency of this work; a SPAR buoy.

The concluding Chapter 8 summaries the achievements of this work, its deficiencies and some of the issues addressed. Further work aimed at refining the current solver and the full modelling of viscous free surface flows around ship structures are listed and commented upon.

Each Chapter builds upon relevant published work in its introduction, as well as containing a summary of the work achieved and the salient points addressed in its conclusion. Relevant figures and tables are presented adjacent to the referencing text.

2 Grid and Data Structure

2.1 Introduction

As stated in Section 1.2.3 and Figure 1.1 the physical domain must be discretised into finite volumes to allow a computational solution to be derived. It has been decided that arbitrarily shaped and sized control volumes will be the most efficient in the modelling of complex marine structures hence an unstructured meshing approach is to be used. In addition, the arbitrary nature of the entity topology and the allowance for adaptation requires a flexible and robust connectivity of the basic geometrical entities. The fast and efficient searching of the grid relies heavily upon the underlying data structure within which the geometrical entities are stored. Previous work (Rycroft [20]) created a complete connectivity data structure and investigated manners of improving efficiency of data storage, and it is this work that has been used as a starting point. Topics investigated in this work are: which geometrical entities are required to be defined, what level of connectivity is required between these entities and the manner in which they are stored.

2.2 Geometrical Entities

The discretisation of the physical domain into a grid is a process whereby the physical continuum is divided into discrete entities. The methods used and the solutions obtained are entirely geometrical entities, thus any measurement of this grid must be in terms of geometric definitions. The base entities of a mesh, in order of hierarchy, are nodes, edges, faces and cells. That is, the most basic entity is a node. A series of nodes together can create edges; a series of edges defines a face; and a series of faces defines a cell. Figure 2.1 shows these four entities. Solutions are limited to a finite volume of space whose extent are defined by specified bounding surfaces. It is within these surfaces that the grid needs to be created.

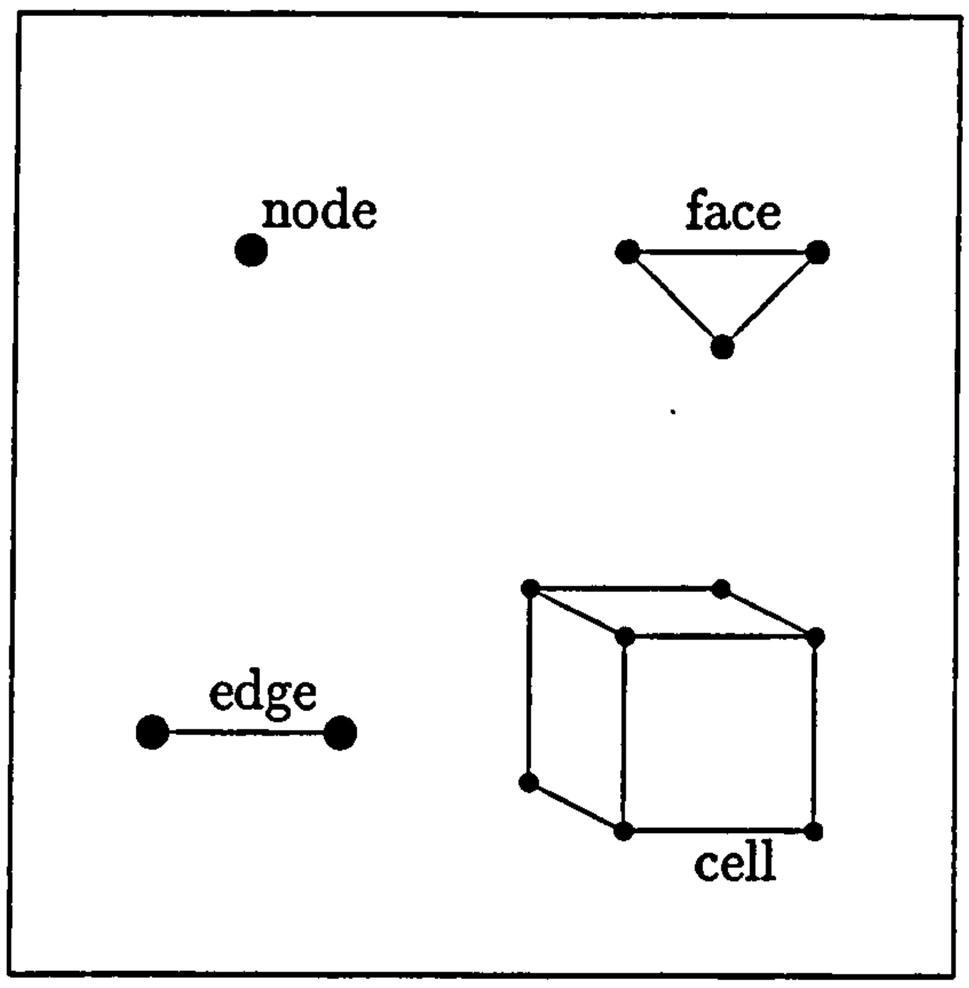


Figure 2.1: The basic geometrical entities

The majority of CFD solvers operate from a Eulerian point of view (fixed co-ordinate system with fluid moving through it), but a number operate within a Lagrangian framework (volume moving with the fluid) or a joint Lagrangian-Eulerian scheme[1]. The result of such schemes is a velocity associated with the mesh entities. It is important to note that for a numerical solution to be obtained the mesh must adhere to the Geometric Conservation Law (GCL) [27], which states that -

$$\frac{\partial}{\partial t}\int_V \mathcal{K}_1 dV + \oint_B (\vec{v}\mathcal{K}_1 - \vec{v_g}\mathcal{K}_1 + \mathcal{K}_3)\cdot dS = \int_V \mathcal{K}_2 dV \tag{2.1}$$

where V is cell volume, $\vec{v}$ the fluid velocity, $\vec{v_g}$ the boundary velocity and S the vectorised area of surface B enclosing the volume. $\mathcal{K}_1$, $\mathcal{K}_2$ and $\mathcal{K}_3$ are tensors that can be altered to define the mass or momentum conservation equations.

This can be broken into two separate aspects, volume conservation and surface area conservation. If there is constant fluid density and a zero velocity field then

$$\frac{\partial V}{\partial t} = \oint_B \vec{v_g}\, dS \tag{2.2-a}$$

and if there is a uniform flow in an arbitrary direction, $\vec{n}$, on a non-moving mesh then

$$\oint_B \vec{n}\cdot dS = 0 \tag{2.2-b}$$

[1]termed ALE - Arbitrary Lagrangian Eulerian

The volume conservation law (VCL), Equation 2.2-a, requires that the volumetric increment of a moving cell must be equal to the sum of the changes along the surfaces that enclose the volume, while the surface conservation law (SCL) states that the cell volume must be closed by its surfaces. Contravention of the surface conservation law will lead to a misrepresentation of the convective velocities and numerical violation of the volume conservation will produce extra sources or sinks in the physical media (Hindman [28], Demirdzic & Perić [29]).

2.3 Mesh Generation

When CFD began to be a recognised field of computational science a quarter of a century ago it was expected that grid generation would be a "solved" problem, with standardised solutions. However, the fact is that grid generation is a major pacing item in regard to the widespread use of CFD in design applications [30]. Cosner in 1995 noted that " a satisfactory grid can be developed to model nearly any configuration of interest. The issues at present focus on operational concerns such as ... quality" [31].

Traditional approaches to mesh generation have focused on the generation of hexahedral meshes which conform to a transformation between Cartesian and curvilinear co-ordinate systems.

The adoption of the finite volume and element methods have permitted tetrahedral, and more generally, polyhedral meshes to be utilised. The advantages of such methods are the greater flexibility that the mesh generation processes provide over traditional techniques, and the increased automation that may be installed. In addition, such unstructured meshes are intrinsically adaptive, a feature which has been subsequently exploited to dynamically manipulate the mesh throughout the solution process.

Recent advances in the construction of unstructured meshes, motivated by increased geometrical flexibility and viscous flow resolution, have lead towards hybrid meshes, constructed using more than one cell topology .

2.3.1 Structured, unstructured and hybrid meshes

Structured Meshes

Meshes in which the connectivity between geometrical entities is implicit are termed structured meshes. This implicit connectivity is generated by the direct transformation from the physical Cartesian domain (x, y, z) to the computational domain(i, j, k). The two main groups of meshes belonging to this family are Cartesian and curvilinear meshes.

Cartesian meshes are constructed directly in the physical space, with vertices aligned with the Cartesian axes. The geometries are identical for all cells, and do not have to be discretely evaluated for each vertex, hence alleviating the need to store co-ordinate data whilst permitting efficient solver algorithms. This method can be expanded for complex geometries using a cut-cell approach, where the geometry is superimposed on a Cartesian mesh. Cells intersecting the solid boundary are cut and the region inside the geometry discarded. Clark et at. [32] and Quirk [33] have demonstrated the ability of this technique to accurately model flow over complex shapes.

Curvilinear meshes are produced by a transformation between the physical domain, described by Cartesian co-ordinates (x, y, z) and a computational domain, described by the generalised curvi-linear co-ordinates (ξ, η, ζ), where the boundaries of both domains coincide. The transformation allows the introduction of localised mesh clustering, such that the cells are non-regular in shape and orientation. This in turn requires that vertex co-ordinate information is stored. The connectivity of the mesh is however maintained uniformly. There are three basic topologies allowed in such meshes; the 'C', 'H' and 'O' meshes. The type of mesh that conforms most accurately with the physical domain tends to produce the best quality mesh. Figure 2.2 shows these meshes for the physical domain of an aerofoil.

The use of a singular transformation for complex, multi-object physical domains result in unsatisfactory physical meshes typically containing highly skewed cells. This considerably limits the complexity of the physical domain for which this method may be used. The multi-block approach extends the structured method by decomposing the domain into a series of sub-domains, termed 'blocks'.

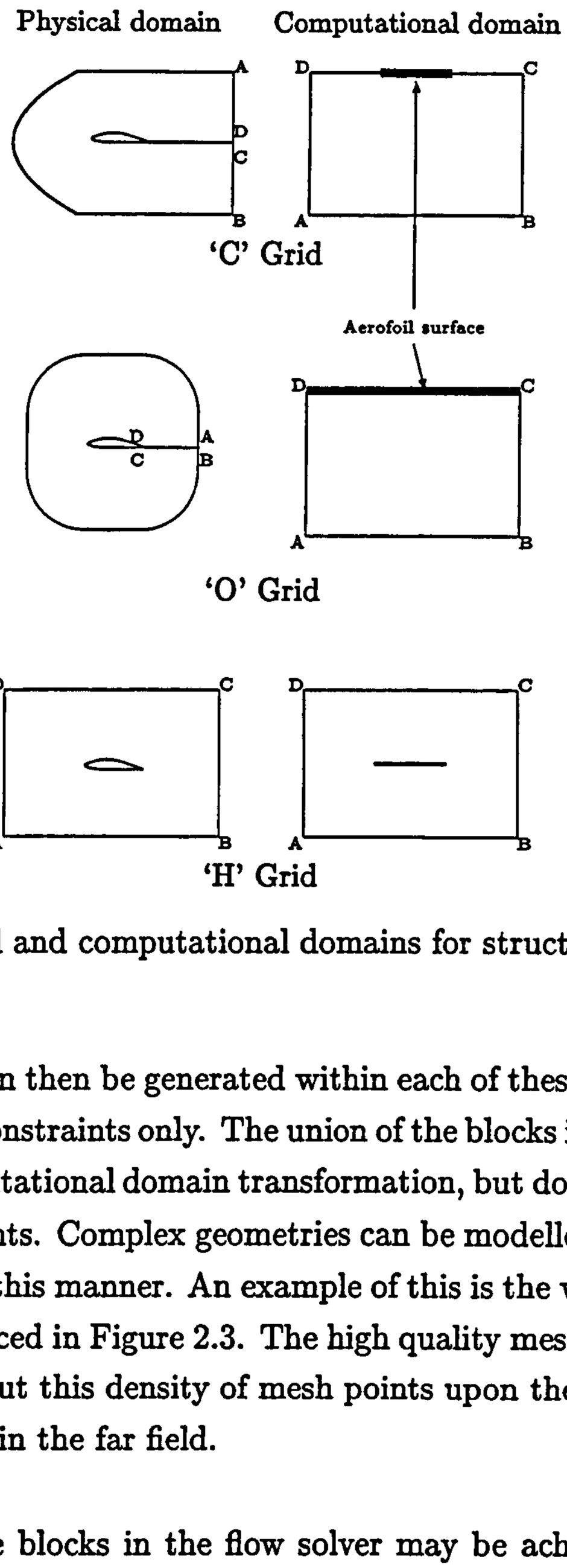


Figure 2.2: Physical and computational domains for structured grid topology

A structured mesh can then be generated within each of these blocks in isolation, conforming to local constraints only. The union of the blocks is not constrained by the physical to computational domain transformation, but does require additional connectivity constraints. Complex geometries can be modelled by decomposition and local meshing in this manner. An example of this is the work of Beddhu et al [34], which is reproduced in Figure 2.3. The high quality mesh on the hull surface can clearly be seen, but this density of mesh points upon the surface leads to an excess of data points in the far field.

The union of the blocks in the flow solver may be achieved in one of two ways. The first is to ensure mesh consistency on the block faces, while the second does not require mesh compatibility, but instead relies upon interpolation routines to communicate information between blocks. Smoothness across the block boundaries is often a major hindrance upon overall grid quality. An increasingly popular solution to this is to allow the blocks to overlap, and hence use the numerical interpolation method of information transfer. Lin et al [35] and Masuko [36] are two examples of the use of this method to model complex marine geome-

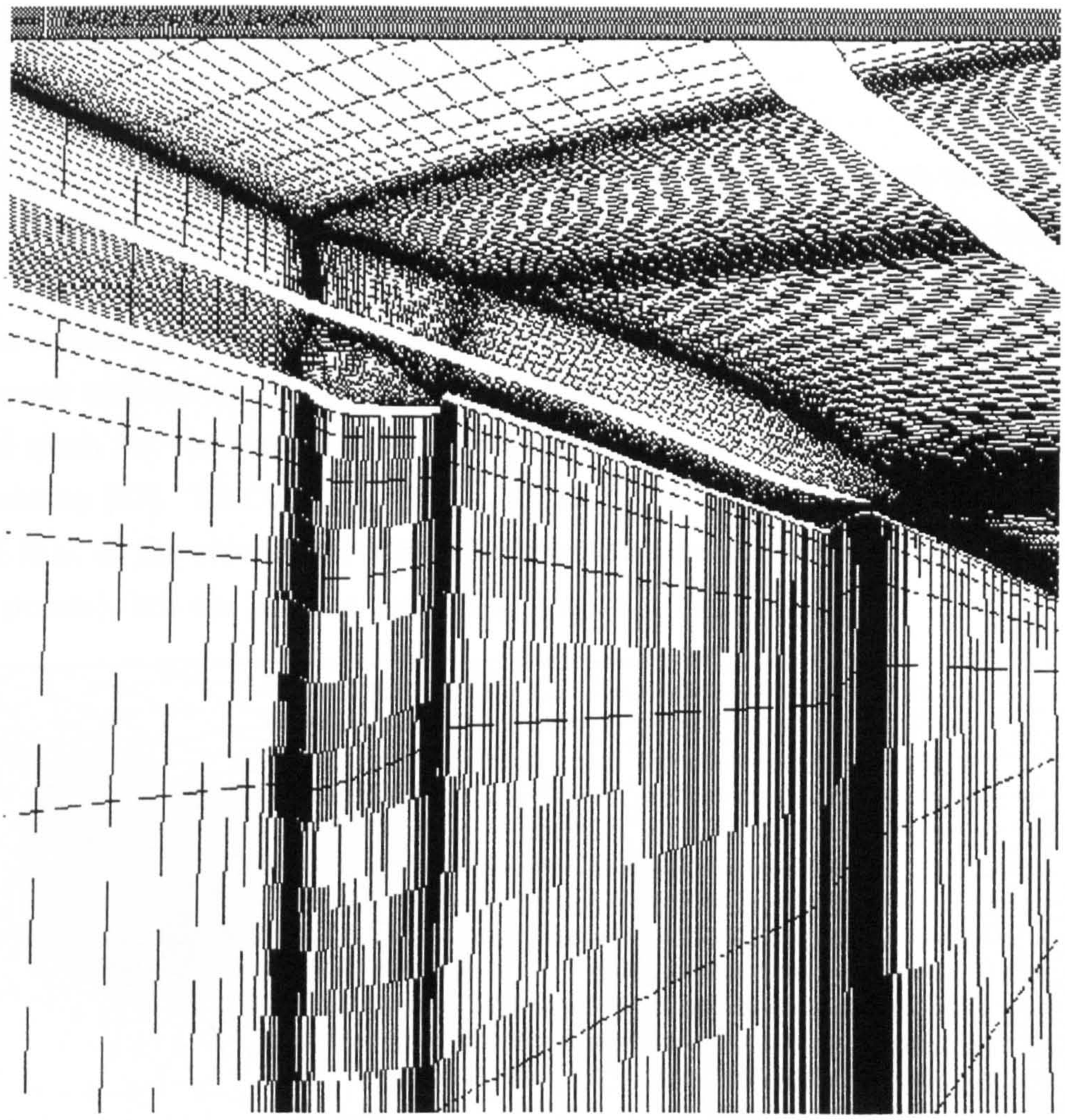


Figure 2.3: Grid structure for seven block DTMB Model 5415 grid. Reproduced with permission from [34] Copyright ©1998 The Author

tries. The drawback of such methods is the presence of redundant data points and the complex interpolation required between points if more than two meshes overlap at a given position [35]. A further alternative to these meshes are octree meshes where the initial mesh is subdivided, in a binary fashion, in the vicinity of interesting flow features. Such methods do capture flow features more accurately while retaining a degree of the efficiency and low memory requirements

of structured meshes. However the user is still limited by the manner in which extra data points are added and the inclusion of hanging nodes [19, Chp 21].

Unstructured Meshes

In unstructured meshes there is no implicit knowledge of the mesh connectivity. The connectivity between geometrical entities is required to be explicit, resulting in an increase of required memory and lowering in efficiency of data management. These additional costs are balanced by a vast improvement in the flexibility of the mesh and its ability to adapt. As a comparison to Figure 2.3, an unstructured mesh for the same hull form is shown in Figure 2.4, again reprinted with permission [37]. The total number of elements in the mesh is approximately a third that of the structured mesh (507 744 tetrahedra compared to 1.44 million grid points), but the surface mesh distribution is comparable.

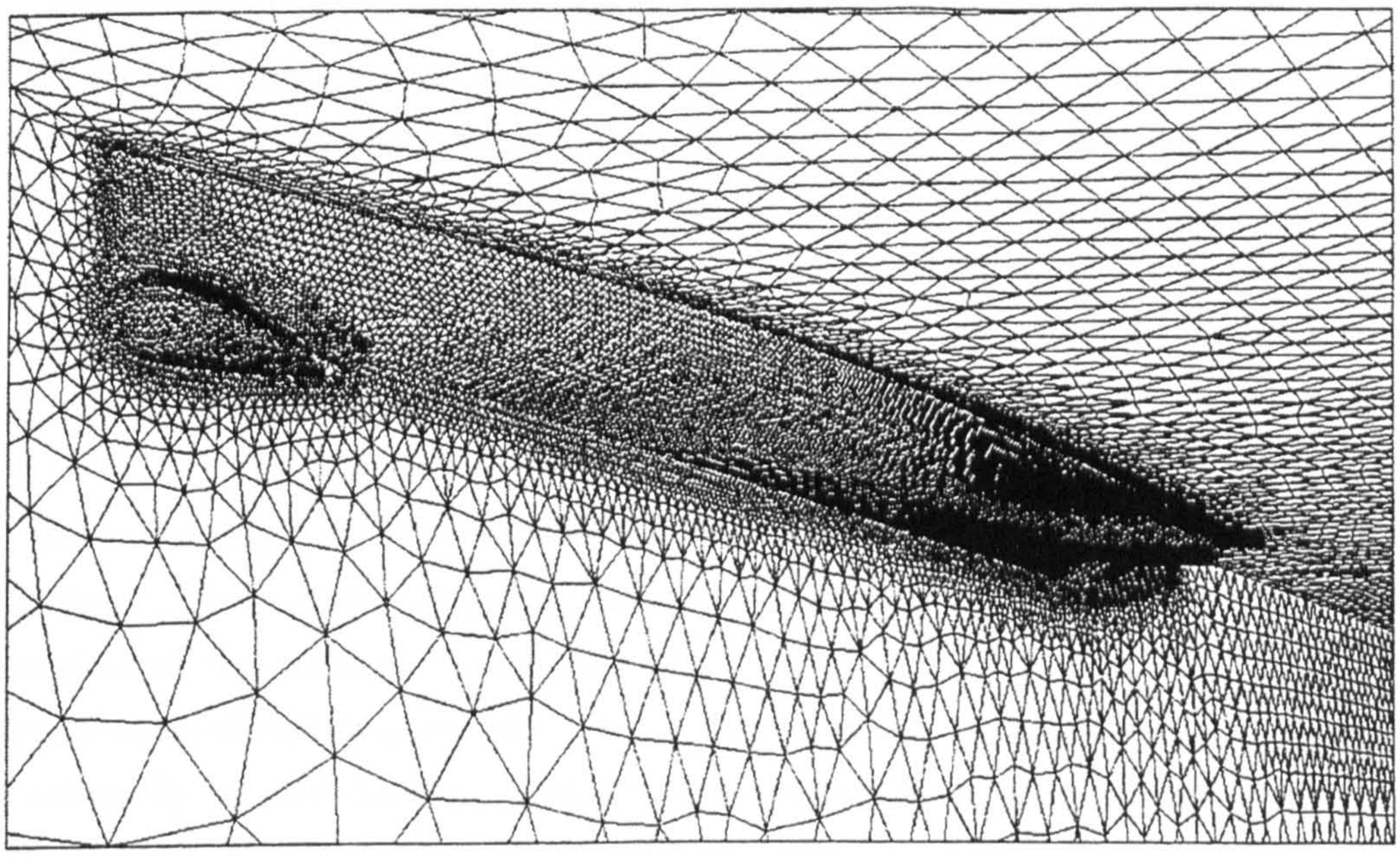


Figure 2.4: Unstructured grid for DTMB Model 5415 grid. Reproduced with permission from [37] Copyright ©1998 The Author

Unstructured meshes have become synonymous with triangular and tetrahedral control volumes since their introduction. However, as the only definition of an unstructured mesh is the lack of implicit connectivity, any topology of control volume can be used.

The forms of creating tetrahedral unstructured meshes are amply detailed

elsewhere (see Rycroft [20]), and hence will not be discussed in this work.

The extra computational cost of the unstructured approach, as previously stated, is present in two forms. Firstly there is the extra storage required for the explicit connectivity between control volumes and geometrical entities, which is normally achieved using matrices describing the mapping of a particular cell topology to its vertices. Each matrix is thus a list of nodal identities or, vice-versa, a list of cells to which a vertex is connected. The additional memory cost can therefore be taken as the cost of these matrices. The 'real' cost however is much harder to clearly define, and is to be expected to be smaller. The use of tetrahedra meshes, and unstructured meshes in general, permits a much more efficient placement of points, and will normally attain the same solution resolution for far fewer control volumes or a higher resolution for the same solution degrees of freedom.

The second disadvantage of unstructured meshes is the increased data placement/retrieval time, due to non-direct memory addressing (e.g. operating on a vertex via the surrounding faces) and poor cache use. The inefficient cache use results from non-contiguous memory storage and non-optimisation of the algorithm with respect to the hardware. In comparison, structured data are arranged in a manner which can be exploited to maximise the efficiency of cache usage by grouping the entities that will be accessed simultaneously.

An additional problem, which is often overlooked, concerns the solver implementation. The correct implementation of higher order algorithms requires an expansion of the numerical stencil. The connectivity of a structured approach permits an easy and natural extension of the present stencil. Such a stencil may not naturally be switched to an unstructured mesh for which local interpolations are necessary, as a rule, to reconstruct the technique [20]. The data structure must accommodate searching within the mesh such that adjacent geometrical entities may be identified at low computational cost.

The final difficulty incurred with tetrahedral based volumes occurs in viscous problems. High aspect ratio cells are a necessary by-product when attempting to gain accurate resolution of the viscous terms, due to the high velocity gradients within the boundary layer, in the free stream direction. However the generation of such volumes is not easily attained. Indeed, Aftosmis et al [38] state that

there is no improvement in solution accuracy using stretched triangulations over hexahedral elements and thus offer a much reduced efficiency. The natural development from this is the use of hexahedral volumes in areas of viscous flow and tetrahedra in the outer domain; a hybrid mesh.

Hybrid Meshes

Hybrid meshes are a sub-set of unstructured meshes and are defined through the use of more than one cell topology. The use of mixed cell topology can improve efficiency as a result of utilising cells with a sparser connectivity matrices compared to tetrahedra and/or to optimise the cell topology with the flow domain, as described above. An example of the type of mesh that can be created using such a method is shown in Figure 2.5.

Weatherill [39] uses a combination of quadrilaterals and triangles to generate meshes over complex 2D shapes. Unstructured 'micro-blocks' were either inserted in areas of poor grid quality or structured blocks were embedded in to the global mesh. Shaw and Peace [40] developed this approach for three dimensions, introducing pyramid volumes at the interface of hexahedra and tetrahedra. Sharov and Nakahashi [41] use a tetrahedral background mesh as a basis for marching a triangular front away from the structured mesh surfaces. The intersections of the two regions are evaluated to form a non-overlapping mesh. Nakahashi and Kano, in a more recent paper [42], have developed this method further by incorporating mesh adaptation, using local gas density as the adaption criterion.

There is a form of crossover with multi-block meshes in hybrid meshes, with separate areas utilising the most appropriate local gridding technique, and connecting with other blocks at set boundaries. As with standard multi-block meshes, over-lapping blocks can be used. An example of this is the work by Gomez [43] on the modelling of the NASA space shuttle. Figure 2.6 illustrates the mesh that Gomez created. As before though, costly interpolation routines are required in the over-lapping regions.

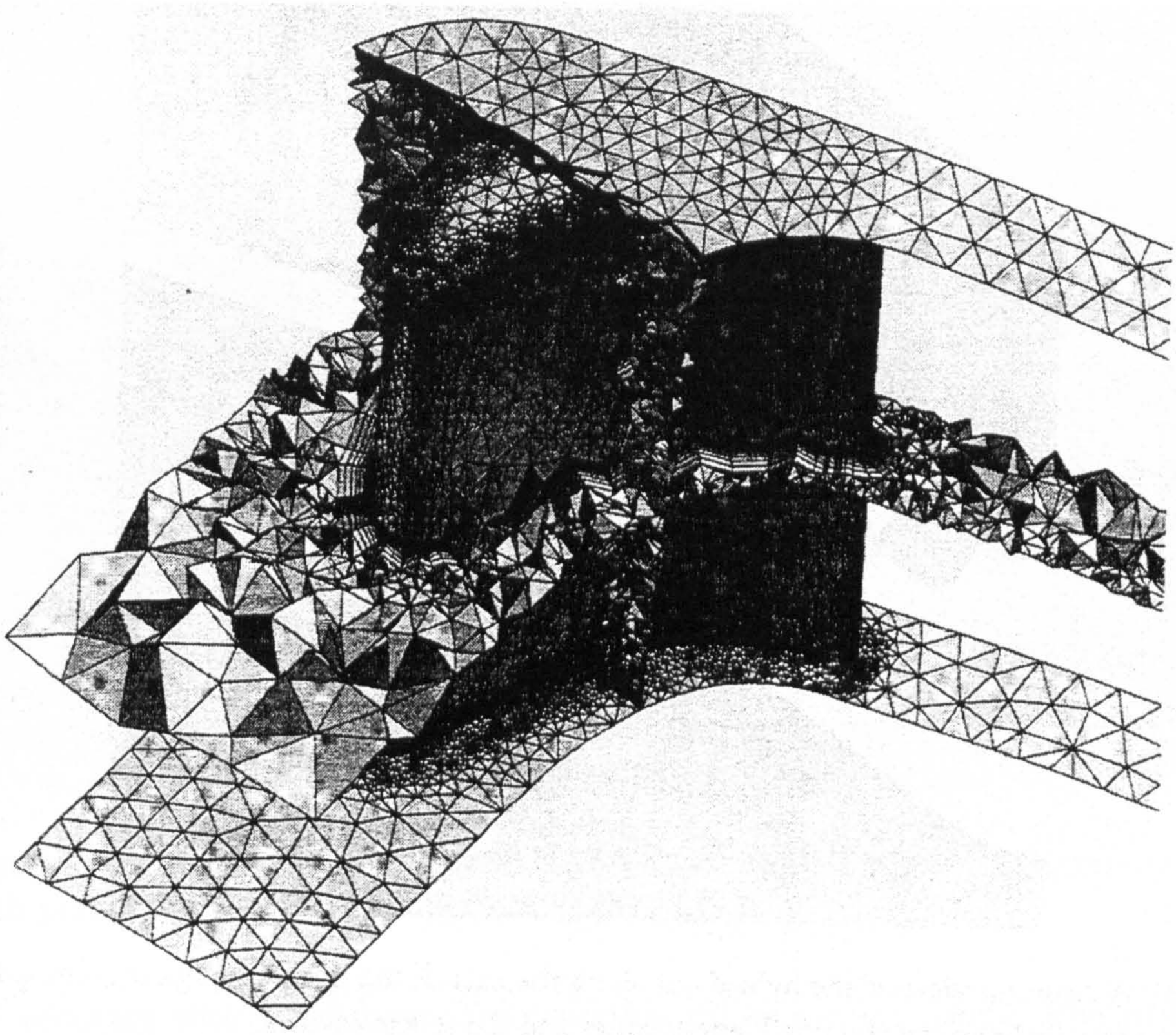


Figure 2.5: A surface and volume hybrid mesh for a turbine blade. Reproduced with permission from [19], Copyright ©1999 CRC Press

2.3.2 Cell centred and cell vertex schemes

The most commonly used method of mesh generation is to store the state vector information at the cell centre, as no extra geometrical construction and calculations are required. The main disadvantage of such methods is that boundary conditions require an extrapolation of the variables onto the boundary face. This causes an extra computational cost and, more importantly, in areas of high flow gradients the interpolation can introduce numerical inaccuracies.

When the flow variables are stored at the vertices of the mesh a larger flexibility of control volume shape is possible; cells surrounding the vertex may be amalgamated, creating overlapping control volumes, or a dual mesh can be created on top of the original mesh. The second option is currently a more common approach (Rycroft [20]). One of the advantages of the cell vertex approach is

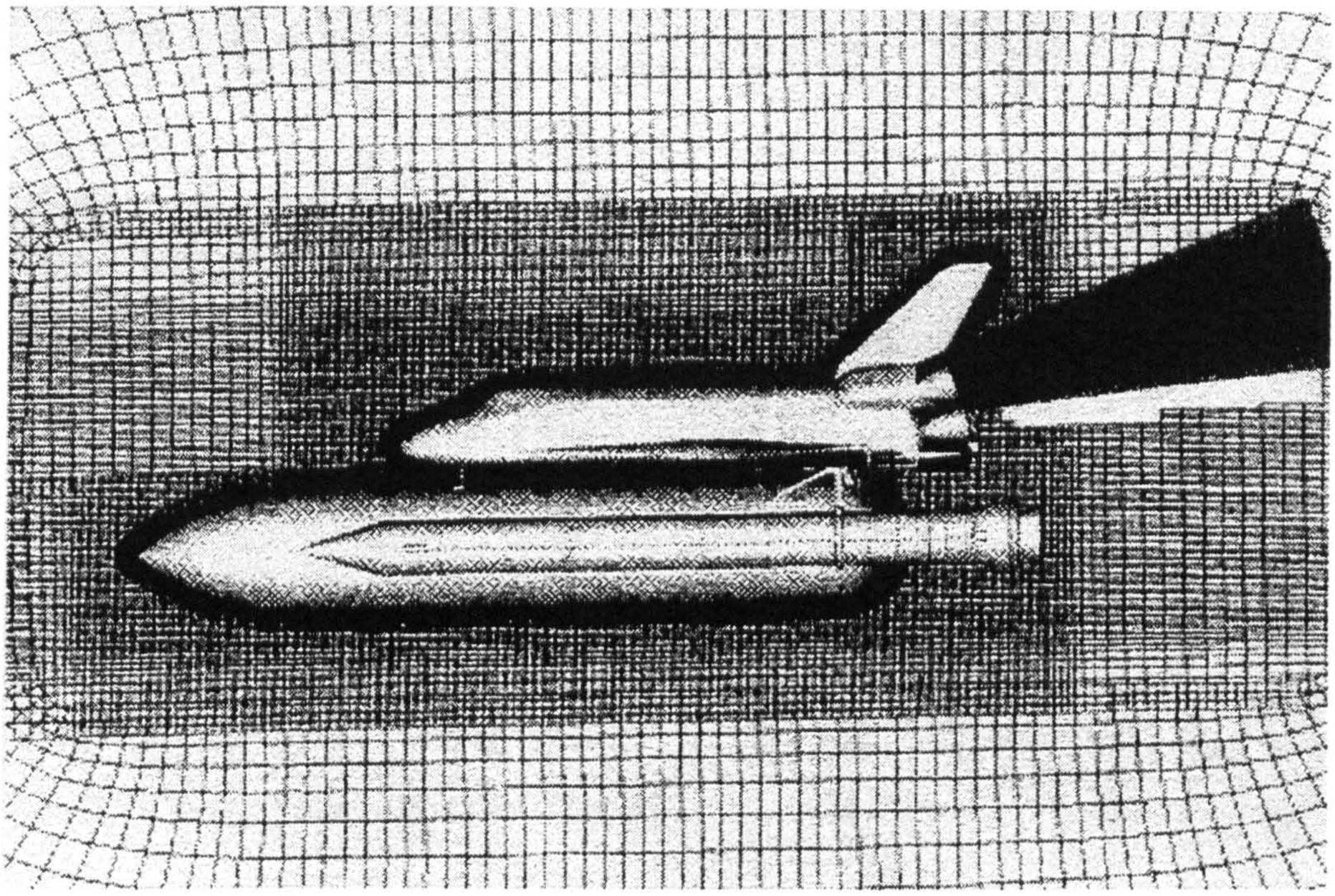


Figure 2.6: CHIMERA mesh around NASA Space Shuttle geometry. Reproduced with permission from [43], Copyright ©1994 AIAA

the accuracy with which boundary condition values can be defined, due to the positioning of vertices on the boundaries. A second advantage stems from the fact that loss of accuracy due to skewed cells has been found by others to be higher for cell centred schemes than cell vertex schemes [44][45]. The derived conclusion of this is that non-uniform meshes would benefit from a cell vertex scheme (Rycroft [20], page 58).

As accurate surface pressures are of prime importance in marine applications (free surface deformation and hull resistance for example) and taking into account the requirement of adaptable and robust meshes a cell vertex scheme has been chosen in this work. This scheme will operate upon unstructured arbitrary control volumes; that is each control volume will have an arbitrary number of faces and be identified in an unstructured manner. Further details of this method of control volume definition are detailed by Rycroft [20] and Wright and Turnock [46].

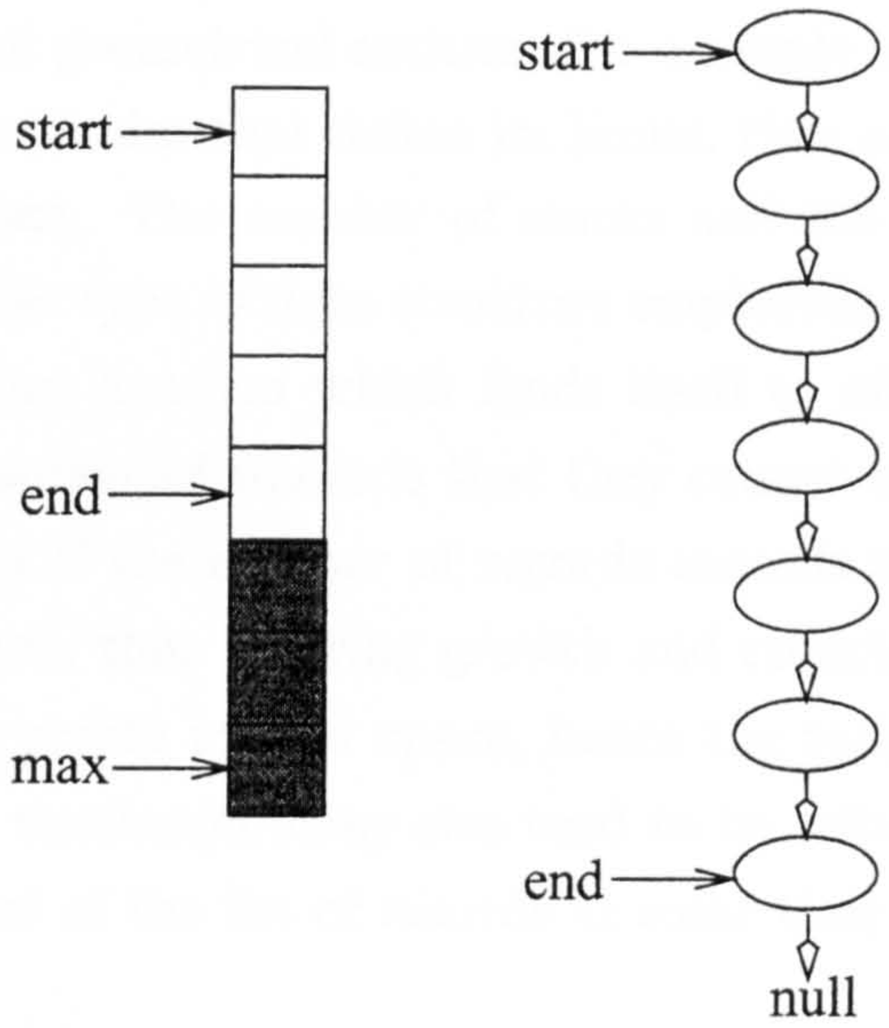


Figure 2.7: A graphical representation of a stack and linked list memory structure

2.4 Data Storage Approach

The term *data structure* defines the relationships between the data items used by the computer program - in this case the geometrical entities. The algorithm of the program requires a variety of operations to be executed upon these entities and hence the manner in which their storage is organised will greatly influence the efficiency of the computer code as a whole. The decision to use a certain type of organisation of data structure depends upon many factors but the most relevant are the type of operations that are to be performed upon the data and the amount of computer memory which is available [19, Chp 14].

There are essentially two forms of data storage; sequentially stored *stacks* or *linked lists* where arbitrary sections of memory are joined by the storage of the memory position of the following record. These two forms of storage can be used to create linear lists, hash tables and tree structures. A graphical representation of these two structures can be seen in Figure 2.7.

Stacks are one dimensional arrays of contiguous memory, and can be used to associate data with a given index in a similar manner as the implicit location of data within a structured mesh. Stacks are however limited to one dimension and can hence only be used to describe the information in a one dimensional list, such as those of nodes. Links between separate stacks can be generated by storing the

stack indices of relevant geometrical entities. For example a face entity may store the stack indices of the nodes that define its limits, thus allowing nodal data to be accessed via the face. The number of stacks and the type of inter-links is clearly prescribed by the type of data structure employed. Stacks store elements in a contiguous memory location which lends itself to efficient memory cache use. The main disadvantage of stacks is that they cannot be manipulated easily, requiring a global reset if the number of records exceeds the stack size. Stacks can be created oversized, thus allowing growth and reduction in the number of records, but this does create unused space, hence the range of allowable ratios of actual array size to maximum array size tend to be critical. There is also the unavoidable global reset of the list of records at some time in the process.

Linked lists are individual locations in memory which are linked, linearly *or* non-linearly, by memory pointers. Each link in the list is treated individually, hence the order of storage can be manipulated and individual links can be removed and added. These manipulation operators can be achieved without changing the memory storage locations, the pointers are merely re-arranged. The clear advantage to this system is its adaptability and ease and speed of manipulation. A disadvantage is the inherent non-contiguous nature of the storage which often leads to a reduction in the efficiency of cache management. In addition, if distributed memory architectures are being considered, the complexity of partitioning the data structure is substantially increased [20]. A third disadvantage, notable for large problems, is that more memory space per record is required due to the memory address inclusion. The final disadvantage, and with the most serious implications for overall algorithm efficiency, is that when global searches are initiated members of the list cannot be accessed directly; the operator is required to trace a path through the entire list, greatly reducing efficiency. This final consideration is often negated by the fact that direct addressing is not needed in many practical applications.[19, Chp 14].

These two forms of storage can be used to create linear lists, hash tables or tree structures. Linear lists are just that; one dimensional lists of data, most often in the form of a stack. Hash tables allow searching to be efficiently accomplished upon a list of data. For each given record the relevant entry in the table will contain all the entities that meet the appropriate criterion. For example, on a two dimensional mesh containing triangular elements, a hash table could be created that lists the nodes on a face, with the indexing value being the node

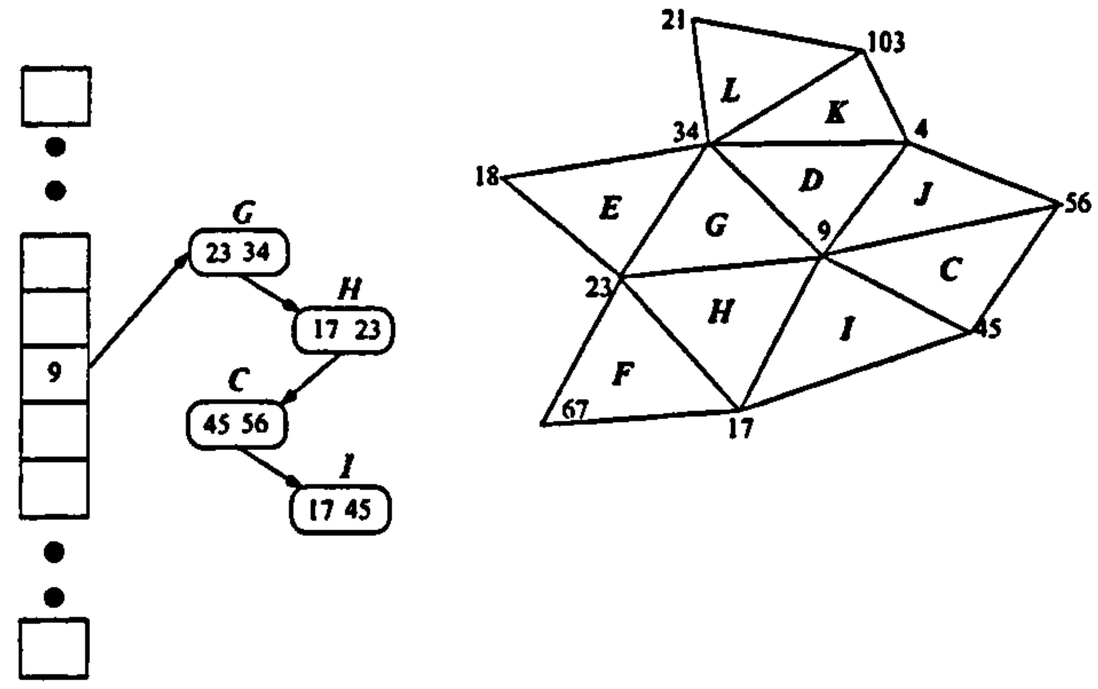


Figure 2.8: An example of a simple hash table, storing node ID connected to a triangular element

with the lowest (numerical) ID. The entry in the table will be the node ID, and the faces connected to that element where it is the lowest ID would be listed in some manner, as shown in Figure 2.8. There are many different forms and levels of complexity of such tables, and can be made highly specific to a given situation, using both sequential stacks and linked lists.

Tree structures are better suited for simple searches to ensure that an entity exists and range searches to find all data with set limits. A tree is a finite set whose elements are called nodes. The root node is the node from which all others can be accessed, and a leaf node is one without any further sub-divisions. Figure 2.9 demonstrates such a structure. Binary trees are ones where each node is linked at most to two branches, and are the most commonly used, but are by no means the only type. A node may have as many branches as it requires; quadtrees have four branches and octrees eight. These structures are often employed for two and three dimensional decomposition of a domain. A tree can be created via the use of linked lists or, in the case of an ordered tree, a stack. A binary tree will be equivalent to a doubly linked list, with pointers to the children. It should be noted however that pointers to parents can be employed equally as easily - the only disadvantage is the extra memory storage requirements.

Ideally for the final data structure used, a balance is sought which maintains the versatility of the dynamic allocation found in linked lists with the efficiency of data handling attained with stacks. Linear lists provide easily handled and retrieved data lists, while hash tables and tree structures allow searches to be efficient, as well as rapid sorting of the data. The case shown in Figure 2.8 shall be taken as an example. If N is the total number of nodes in the mesh, F the

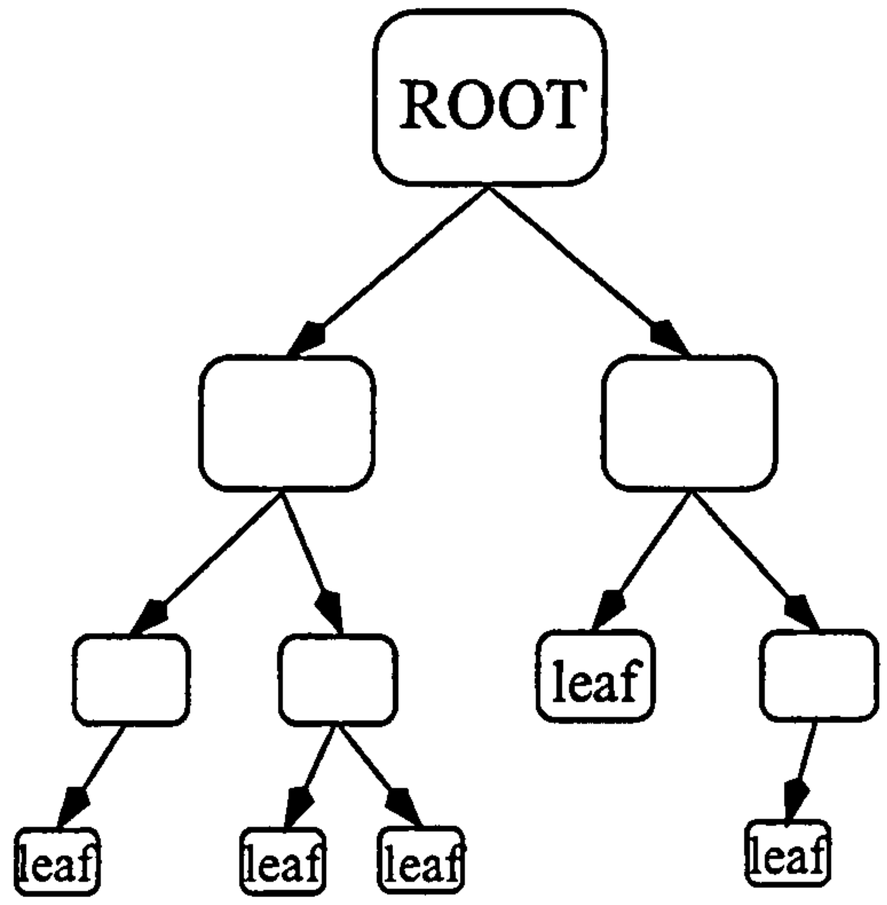


Figure 2.9: The representation of a simple tree structure

number of faces, and **R** the total number of records listed then -

- Memory requirements = $\mathbf{N}PTR + \mathbf{F}(PTR + 2INT)$
- Operations for search = $\mathrm{O}(1 + \mathbf{R}/\mathbf{N})$
- Operations for adding record = $\mathrm{O}(1)$

where PTR and INT are the memory requirements for a pointer and integer respectively. The average value of $(\mathbf{R}/\mathbf{N})$ is generally small - ≈ 6 for a two dimensional mesh. If the data was stored in an array however, with **F** entries, each of which contain the three nodes that define the face, then -

- Memory requirements = $\mathbf{F}(3INT)$
- Operations for search = $\mathrm{O}(3\mathbf{F})$
- Operations for adding record = $\mathrm{O}(\mathbf{F})$

For a linked list of the same structure as the array then the memory requirements would increase to $\mathbf{F}(3INT + PTR)$ while the number of operations required for adding a record would drop to $\mathrm{O}(1)$.

The hash table demonstrated is clearly more efficient at retrieving the correct data but it is dedicated to only one type of search. More complex hash trees can

have increased generality and thus allow a greater number of searches to be undertaken, but at the price of increased memory requirements.

2.5 Data Structure Implemented

The data structure implemented was developed from previous work within the Fluid Structural Interaction Research Group at the University of Southampton (see Rycroft [20], Turnock and Rycroft [47]). A cell vertex scheme has been utilised because of the increased accuracy in surface pressure modelling, as well as the fact that non-uniform meshes benefit from a cell vertex scheme [20, page 58]. The previous scheme has been proven to work for hybrid meshes of arbitrary topology but the memory overhead was high ($\approx 600\%$ with respect to a structured hexahedra mesh) and parts of the data structure were redundant for large sections of the computational process. The level of geometrical information, the nature of the mesh, the mesh connectivity and the nature of storage within the computer architecture have all been considered in a drive to improve the efficiency of the structure.

2.5.1 Arbitrary cell topology

Mesh adaption, as required if an optimal grid is to be developed during the solution phase, is one of the main drivers in the use of unstructured data types, and is often intrinsic in the data structure used. Unfortunately the use of hybrid meshes, with mixed cell topologies reduces the adaptive capabilities of the approach. Weatherill [39], as well as Shaw and Peace [40], explicitly define a limited number of control volume topologies for which their algorithms will operate successfully. This clearly limits the adaption process such that adjacent cells conform to a set connectivity matrix. A significant problem, particularly for cell centred schemes, is that of 'hanging nodes'. Figure 2.10 details a typical hanging node. If two hexahedra are adjacent to each other and one hexahedron is split into four smaller hexahedra, then there is a 'hanging node' on the interface. This presents a disjoint in the mesh, where one control volume on an interface uses the node, but the other does not. Fluxes evaluated on the adapted quadrilateral faces are not intrinsically equal to the flux through the face of the second hexahedron,

and this may lead to a lack of conservation.

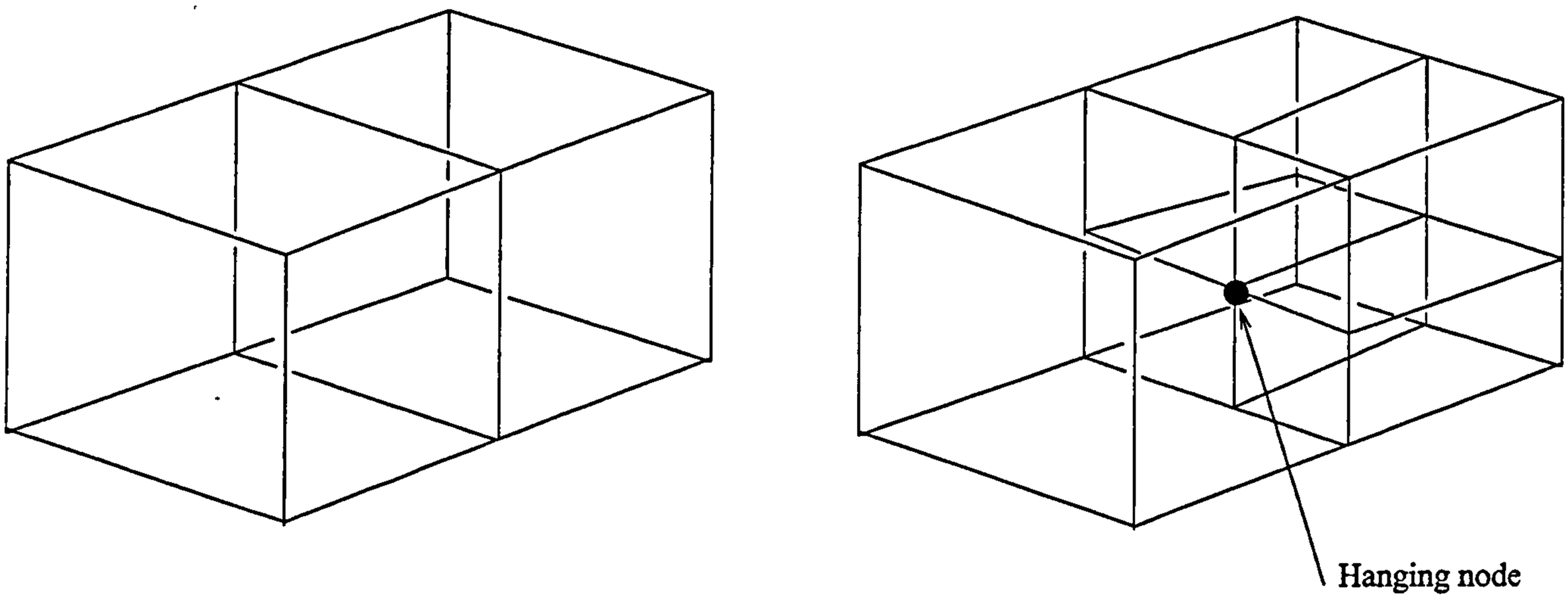


Figure 2.10: Definition of a hanging node between control volumes

Ideally a data structure should be able to process control volumes, regardless of their topology. Rycroft [20] used the full range of geometrical entities, with complete sequential connectivity between them. Because of this complete connectivity, the ordering of individual elements by another entity was not important. As specified previously in Section 2.2 there are four basic entities from which a mesh is created - nodes, edges, faces and cells - hence there were links from nodes to edges, from edges to nodes and faces, from faces to edges and cells and from cells to faces.

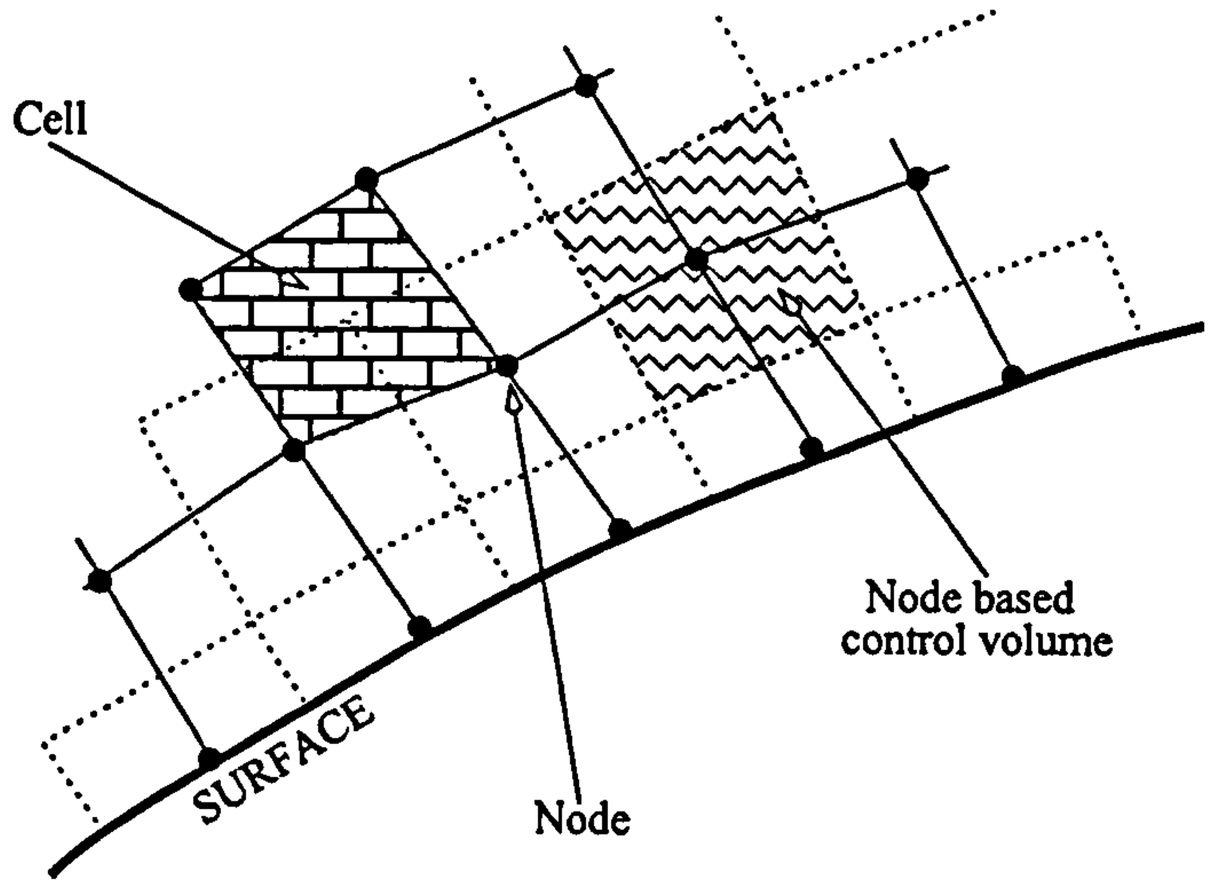


Figure 2.11: Cellular mesh, with over lying node-centred control volume mesh

It should be reiterated that the cells are not the control volumes used in the solver, due to Rycroft's use of a node centred scheme, rather they are the cells of

the initial mesh. The control volumes are created from the overlying node centred dual mesh, as shown in Figure 2.11. It can therefore be observed that the cells are generally redundant during the solution phase, used only for overall connectivity. The flow solver devised by Rycroft operated in an edge based algorithm, passing fluxes between nodes, the edges of the cell centred mesh defining the interface between node based control volumes.

Entity definition

For the present work it was decided to operate purely with the overlying dual mesh. This decision was made because no justifiable benefit of the cell centred mesh could be established. As a results, the interfaces of the control volumes are actual faces of the mesh. Therefore it was decided that the flow solver would operate using a face based algorithm.

With the use of pointers to the nodes lying either side of the face, as well as node to node pointers, edges have been made entirely redundant. The use of node to node pointers also allows complete mesh connectivity, and the face to node pointers allow spatial magnitudes to be defined. These two factors also result in cellular definition being superfluous. Hence the scheme devised has negated the requirement for edge and cell definition. The only connectivity required is face to node and node to node.

Due to the unstructured nature of the data, and the arbitrary polyhedral shaping of each control volume the algorithm utilises triangular faces (henceforth termed *subfaces*) the union of which defines the faces of the control volume, as shown in Figure 2.12. The face area is a summation of the triangular segment areas and the facial normal is an area weighted average of the segment normals. This allows all geometrical properties to be calculated and the flux terms to be evaluated.

From these faces control volumes can be created, with the Global Conservation Law of surface area conservation being implicitly adhered to. The volume of the cells can be calculated from the triangular face segments, in an extension of the facial area calculation. Each triangular segment is defined as the base of a tetrahedron, with the apex being the control volume data point (i.e. the grid

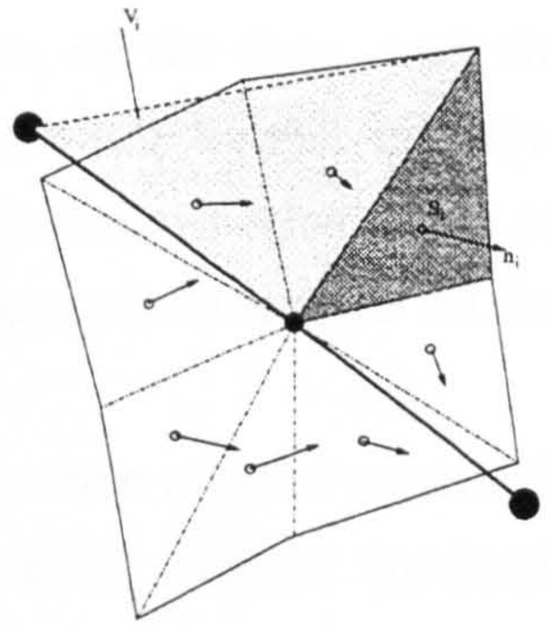


Figure 2.12: Control volume face definition

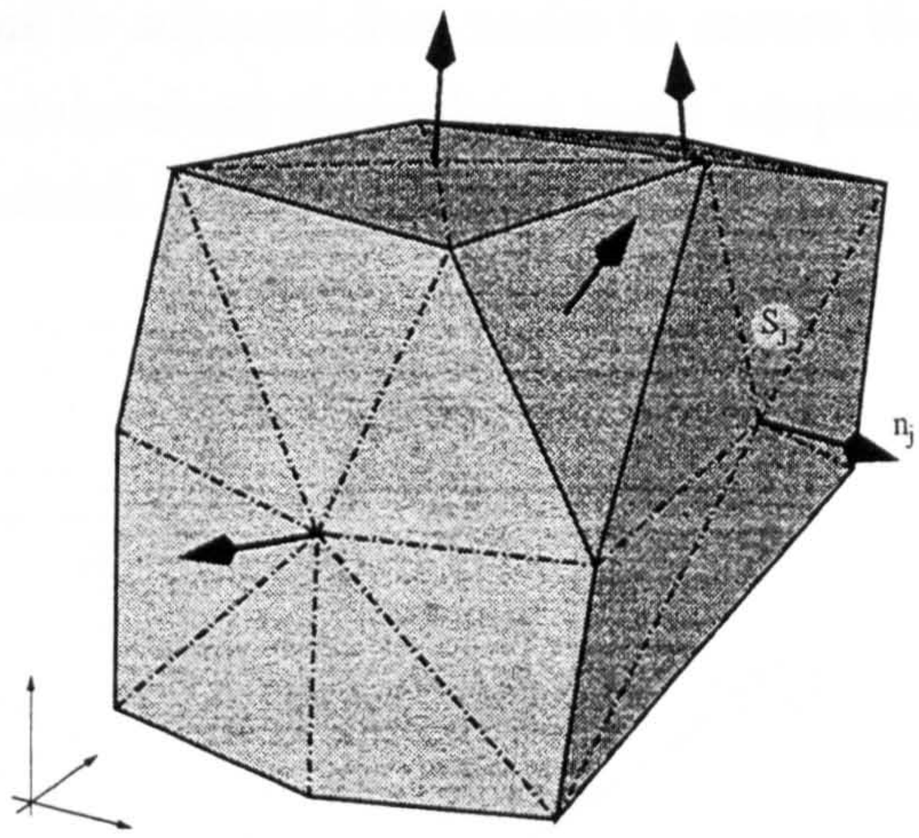


Figure 2.13: Polygon control volume

node). The summation of the volumes of these tetrahedra equals the volume of the cell. Figure 2.12 shows the planar area of a triangular segment and the volume subtended to it from one of the grid nodes adjacent to the face. It should be noted that this triangular segment subtends the grid node on the other side of the face, and hence a tetrahedral volume will have to be calculated for it as well.

If the mesh is constructed from a range of topologies there is no limit on the number of faces connected to a given vertex. Thus the control volumes associated with such a mesh may be complex polygons, as shown in Figure 2.13.

Mesh connectivity

Because a cell vertex scheme has been chosen, the flux to/from a control volume requires that the face knows which control volumes it is attached to.

To allow the construction of the face, and the calculations of areas and volumes, the faces have a connection to the 'construction' nodes that define the limits of the face. These construction nodes are in the position of cell and facial centres of the initial mesh. Figure 2.14 details all these connections graphically.

The construction nodes in turn require connection to the grid nodes (henceforth termed flow nodes) to allow correct interpolation of flow properties at boundaries but more importantly to aid the local remeshing process, and the subsequent reallocation of entity connections. In conjunction with this the flow nodes have connections to adjacent flow nodes to ensure the geometrical Global Conservation Law is maintained throughout mesh adaptation as well as aiding the efficiency of the remesh procedures.

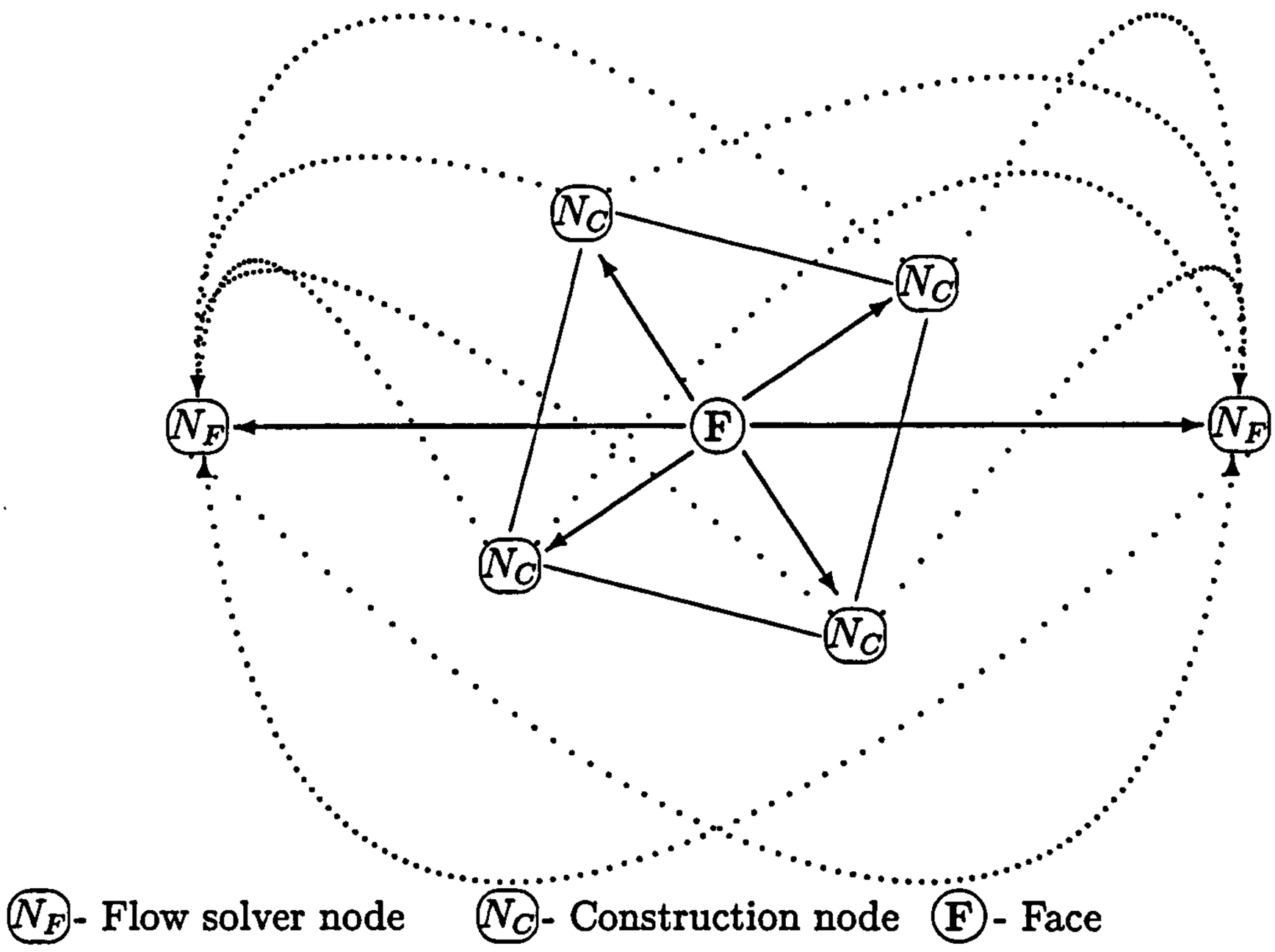


Figure 2.14: Connectivity of geometrical mesh

If the mesh were to be stationary and not subject to re-meshing during the solution then the node to node connections (including the construction node to node connections) can be removed. This would leave only face to node pointers defining the entire mesh connectivity.

Additional benefits of this structure are principally the ability to conduct a number of global searches efficiently without extra hash tables being created.

Other benefits are the allowance for storage of extra information such as local geometrical properties and the reduced logic requirements for implementing boundary conditions due to the explicit inclusion of the face entity.

The in-house mesh generator *FLEXIMESH*, detailed in Ship Science Report 101, [48] and by Rycroft [20] was altered to produce files of correct format in both ASCII and binary format. Machine portable binary files were not generated due to the developmental nature of the code and the relatively limited number of architectures that the code was tested upon.

Memory requirements

As an example to compare the varying levels of memory required by different approaches a structured mesh of hexahedra with N^3 points shall be used. As before , *PTR* signifies the memory requirement of a pointer, *INT* the memory space of an integer and *FLT* the memory for a float. The memory cost of all three types is normally identical.

Structured mesh The only data requiring storage are the spatial co-ordinates of the data points -

$$\text{MEMORY} = 3N^3 FLT \tag{2.3}$$

Unstructured hexahedhra mesh In addition to the data point co-ordinates the cell to node connectivity is also required -

$$\text{MEMORY} = 3N^3 FLT + 8(N-1)^3 PTR \tag{2.4-a}$$

$$\text{OVERHEAD} = 8(N-1)^3 PTR \tag{2.4-b}$$

Rycroft's arbitrary topology mesh The complete connectivity list devised would require each cell to list six faces, each one of those faces upon the cell to list four edges and those edges to list two nodes (hence $6 \times 4 \times 2$ connections)-

$$\text{MEMORY} = 3N^3 FLT + 48(N-1)^3 PTR \tag{2.5-a}$$

$$\begin{aligned} \text{OVERHEAD} &= 48(N-1)^3 PTR \\ &= 600\% \text{ of standard unstructured mesh} \end{aligned} \tag{2.5-b}$$

New arbitrary topology mesh Because this mesh operates upon node centred control volumes the equivalent mesh was deemed to be one with $(N-1)^3$ control volumes; this specifies N^3 construction nodes and $(N-1)^3$ flow solver nodes. Each face will list six nodes (four construction nodes and two flow solver nodes), each construction node will list the surrounding flow solver nodes (eight construction nodes per control volume) and each flow solver node will list the surrounding flow solver nodes. In addition to this the spatial co-ordinates of the flow solver nodes will also be required -

$$\begin{aligned} \text{MEMORY} &= (3N^3 + 3(N-1)^3)FLT \\ &+ (6 \cdot 3N(N-1)^2 + 8(N-1)^3 \\ &+ 2 \cdot 3(N-2)(N-1)^2)PTR \end{aligned} \tag{2.6-a}$$

$$\begin{aligned} \text{OVERHEAD} &= 3(N-1)^3 FLT \\ &+ (18N(N-1)^2 + 8(N-1)^3 \\ &+ 6(N-1)^2(N-2))PTR \\ &\simeq 437.5\% \text{ of standard unstructured mesh} \end{aligned} \tag{2.6-b}$$

Assuming floats and pointers have the same memory cost, then the memory overhead of the arbitrary topology compared to a purely structured mesh is ≈1167% (overhead of Equation 2.6-b divided by memory requirements of Equation 2.3). While there is still a substantial memory overhead compared to a purely structured mesh there has been a clear improvement for the previous arbitrary topology structure. It worth noting also that if the node to node connections are removed then the overhead drops to 262.5% of the unstructured hexahedral mesh. This would be possible if mesh adaptation were not required or if more complex search algorithms were used during adaptation. As stated in Chapter 2.3.1, the true memory cost is likely to be considerably lower. Exact studies of this behaviour are detailed in Chapter 6.

2.5.2 Hybrid memory structure

In conjunction with the development of the basic data structure, and inter- connectivity, code development has concentrated on determining the most efficient manner of accessing the data. A combination of one dimensional lists of data

(stacks) and non-contiguous linked lists have been utilised with memory pointers to produce a structure that allows efficient access to data as well as minimising memory overheads. The manner of accessing the data structure is shown in Figure 2.15. Memory pointers are denoted by arrows, and stacks by segmented boxes. It should be noted that the stack of memory allocated to storing facial data is over-sized, to allow for the creation and destruction of faces during grid adaptation without costly global reallocation.

The use of linked lists for the storage of all entities was deemed to be essential when considering the level and regularity of adaptation to be undertaken. The faces and nodes have also been given direct access from the array of pointers for ease of use. Direct access to the faces is required during the solver algorithm (which is facially based) and direct access of the nodes greatly benefits the remeshing process as described in Chapter 6. The construction nodes never require sufficient direct access to warrant a stack of pointers. The vast majority of construction node use is via the faces, which point directly to the node in question.

2.5.3 Entity fission and fusion

A generic tree data structure has been devised for the flow node section of the data structure to allow flow solver node creation and destruction by remeshing. As a node is split, it is 'deactivated' (denoted by the graphical representation of a node being covered), and pointers to its 'children' created. These children are not direct members of the stack accessed by the overall grid structure, but via the parent node. Figure 2.16 demonstrates how this process can be repeated for several 'generations' of children. Faces do not need this structure, they are merely maintained in the linear linked list, with new faces added to the end, or removed as necessary.

2.5.4 Surface definition and tracking

As stated in Chapter 1.1, the long term goal of this work is the accurate modelling of free surface flows around realistic marine structures. This requires two additional topics that have not yet been discussed; the free surface model and

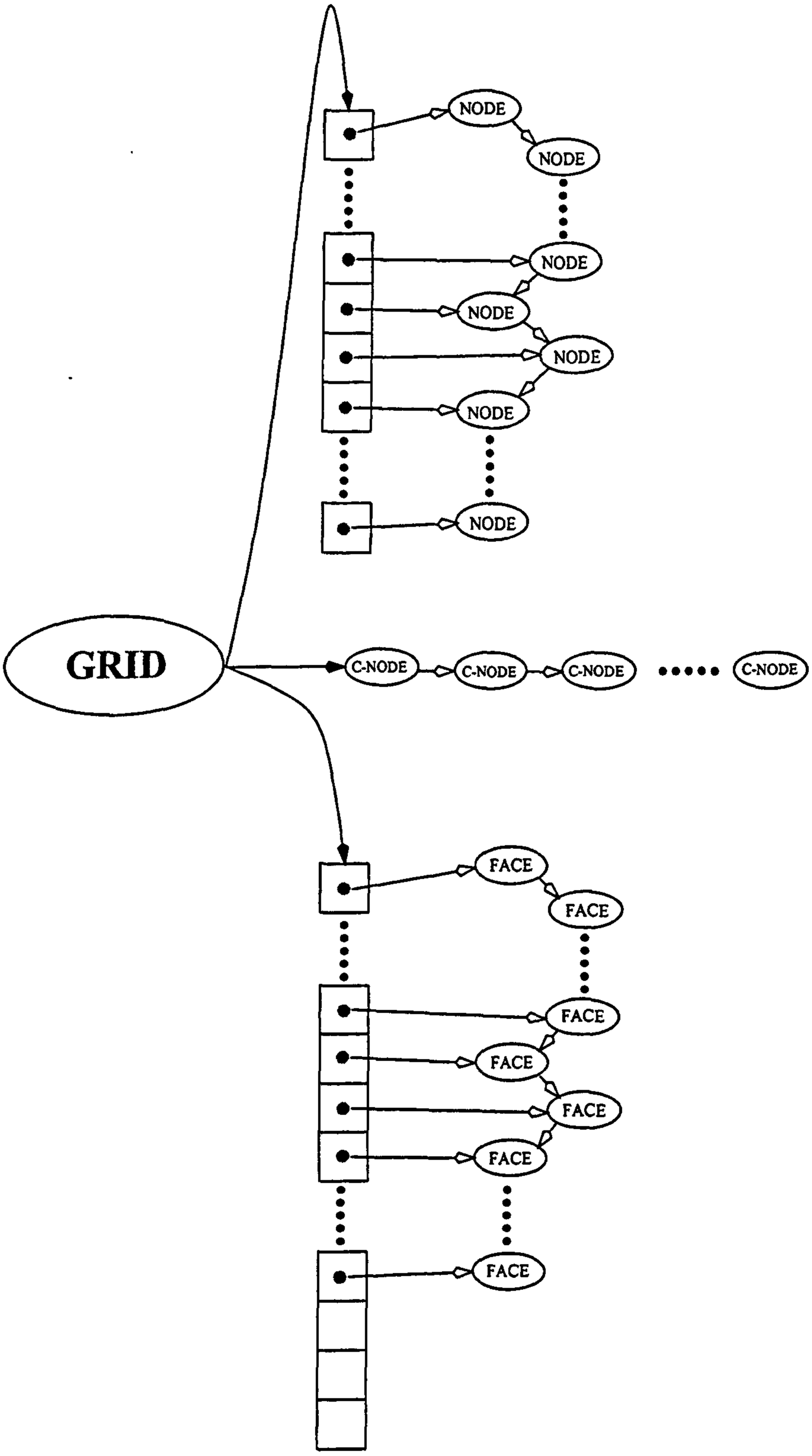


Figure 2.15: Data structure of geometrical mesh

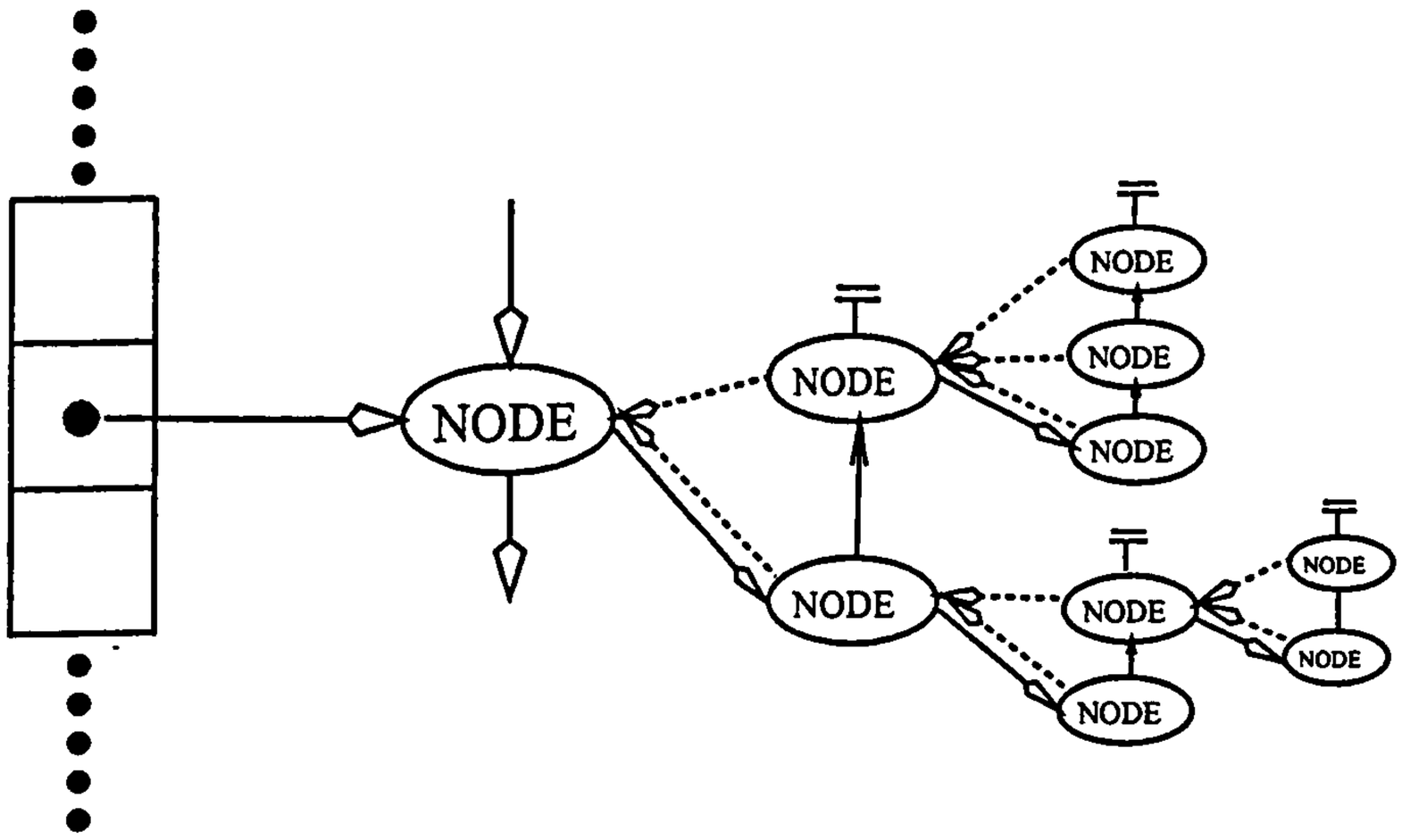


Figure 2.16: Tree data structure of flow solution nodes

surface definition (be it the air/water boundary or the surface of a structure). The subject of the free surface definition is complex, and is discussed in detail elsewhere [24, 25, 26, 49, 50], hence will not be discussed in this thesis. The only concepts that are required are the use of *surface capturing* and *surface tracking.* Surface capturing analyses the entire domain, both water and air, and defines the surface by a separate function, often density based. If air has a given unique function value, as does water, the interface is where the function equals the average of these two values. Surface tracking models only the water, with free surface deformation defined by the kinematic and dynamic free surface boundary conditions [51].

The major advantage of surface capturing techniques over surface tracking methods are their capability to model steep and breaking waves, as can be seen in figure 2.17 as well as the ability to maintain a high mesh quality due to lack of movement in the mesh. Such methods do however require much larger memory storage, due to the increased mesh size, and the interface position tends to suffer from numerical dissipation and fails to define a sharp boundary position[24, 26].

Even without the use of a surface tracking free surface algorithm there is a requirement for accurate surface definition. As the computational mesh adapts there is a requirement for knowledge of the underlying surface definition, and if the bodies are required to move relative to each other then the tracking of the

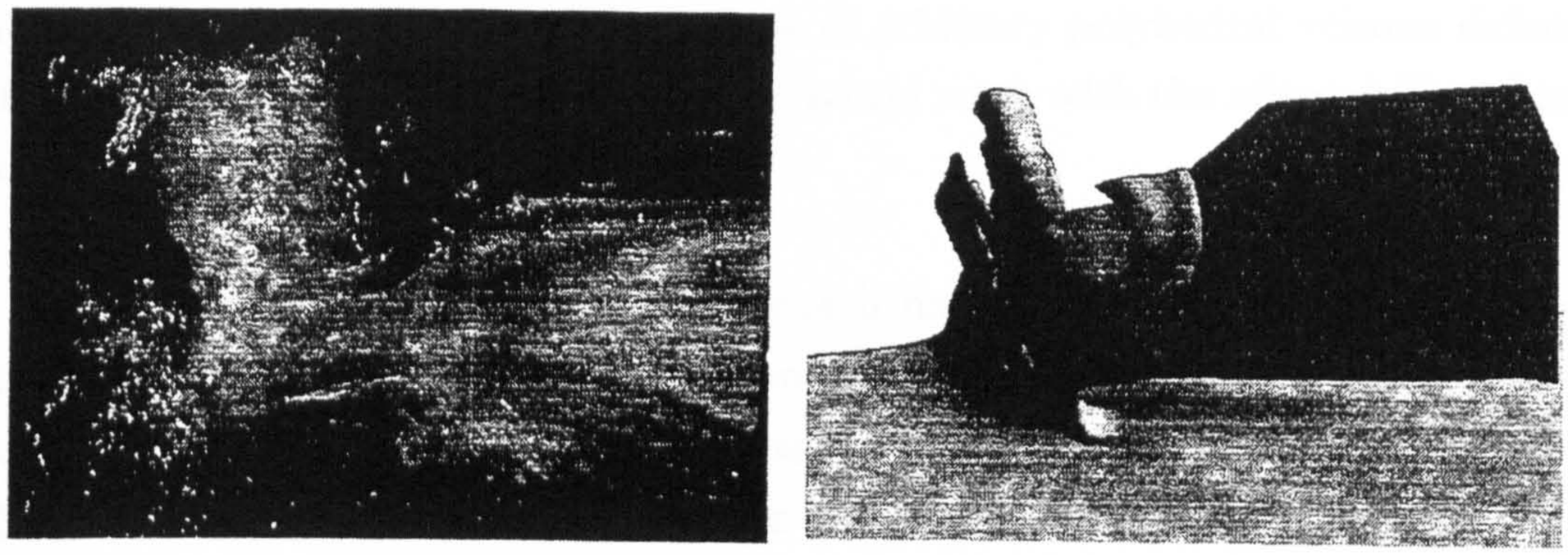


Figure 2.17: Typical results from a modern surface capturing free surface code. This particular case study has only been verified qualitatively. Reproduced with permission from [52]Copyright ©1999 The Author

position of a partical upon a surface is required. As with the computational grid this has been achieved via an advanced hybrid data structure, with computational mesh nodes linked to the relevant surface patch. This work is not central to the subject of this thesis, and hence will not be discussed further. Full details can be seen in Turnock and Wright [53].

2.6 Summary

Structured mesh generation requires complex manipulation in order to mesh realistic geometrical configurations, such as geometrical singularities and diverse geometrical features. Triangular based unstructured meshes enable efficient mesh construction but do not provide sufficiently accurate solutions for viscous flow regimes. Hybrid meshes overcome such problems by the use of different control volume topologies in different domain zones. These meshes however reduce the inherent adaptability of unstructured meshes via the restriction of cell types. The

approach chosen for this work is the use of arbitrary polyhedral volume definition which incorporates the benefits of a hybrid mesh with the adaptability of an unstructured mesh.

A node based control volume approach has been adopted because of the importance of surface pressure definition in marine fluid flows. In order to accommodate the arbitrary polyhedra a data structure has been created that defines the mesh via geometrical entity definition and connectivity. This structure is 37% more efficient with memory use than previous such structures.

Increased ease of adaptation has been accommodated by storing entities in linked lists and using a tree structure to allow rapid cellular fission and fusion. Direct accessing of the node and face entities is achieved via the use of stacks of pointers to the linked lists.

This grid definition and data structure has been used in the flow solver which is described in Chapters 3 and 4.

3 Finite Volume Flow Solver

3.1 Introduction

As stated in Section 1.2 there are three techniques commonly in use today to solve discrete computational problems; finite difference, finite element and finite volume. The finite volume method discretises the integral form of a conservation law. Introduced into the field of numerical fluid dynamics independently in 1971 and 1972 for the solution of two dimensional Euler flow, it presently has a wide range of applications and two distinct advantages. The method can take full advantage of an arbitrary mesh, allowing control volume shape and location manipulation, and the adoption of direct discretisation of the integral formulation ensures that the basic entities (mass, momentum and energy) will also remain conserved [9, page 237].

This Chapter introduces the governing equations of fluid dynamics and the method of discretising space into finite volumes is described. Section 3.4 details the method of artificial compressibility, and how standard compressible flow techniques can be used to create an efficient incompressible flow solver.

3.2 Euler Equations

The Euler equations express a set of conservation laws for the quantities mass, momentum and energy. They are a reduced form of the full Navier-Stokes equations, describing inviscid flow where all viscous and thermal conductivity terms are neglected. This form of the governing equations is an improved approximation of fluid behaviour over the potential flow equations as it models rotationality.

Analytically, the primary advantage of the Euler equations over those models containing viscous terms is that the system of partial differential equations reduces from second to first order.

The time dependent Euler equations, in conservative form, may be written in vector notation as -

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = \mathbf{Q}, \tag{3.1}$$

where $\mathbf{U}$ is termed the state vector, $\vec{\mathbf{F}} = (\mathbf{f}, \mathbf{g}, \mathbf{h})$ are the flux vectors, and $\mathbf{Q}$ is a vector of source terms.

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uH \end{bmatrix} \mathbf{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho vH \end{bmatrix} \mathbf{h} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho wH \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} 0 \\ \rho f_{ex} \\ \rho f_{ey} \\ \rho f_{ez} \\ \rho W_f \end{bmatrix} \tag{3.2}$$

ρ is the density, u, v and w are the velocity components in the x, y, and z Cartesian co-ordinate directions, E is the specific total energy, p is the static pressure and H is the specific stagnation enthalpy. In the source term vector $\vec{f}_e$ is the external force and W_f the work performed by the external force ($W_f = \rho \vec{f}_e \cdot \vec{v}$).

The Euler equations have still to be supplemented by the constitutive laws. The thermodynamic laws define the internal energy e or the enthalpy h as a function of only two other thermodynamic variables chosen between density ρ, pressure p, temperature T, entropy s or any other intensive variable.

For an ideal gas (compressible flow such as air), the equation of state is written as

$$\frac{p}{\rho} = RT, \tag{3.3}$$

where R is the gas constant per unit mass and is equal to the universal gas constant divided by the molecular mass of the fluid. As a result the internal energy and enthalpy are only functions of temperature and have the following relationship with the ratio of specific heat coefficients,

$$\gamma = \frac{C_p}{C_v}, \tag{3.4}$$

where C_p is the specific heat coefficient under constant pressure and C_v the specific heat coefficient under constant volume.

$$e = C_v T = \frac{1}{\gamma - 1}\frac{p}{\rho} \tag{3.5}$$

$$h = C_p T = \frac{\gamma}{\gamma - 1}\frac{p}{\rho} \tag{3.6}$$

$$H = E + \frac{p}{\rho} = h + \frac{\vec{v}^2}{2} \tag{3.7}$$

$$c^2 = \gamma R T = \frac{\gamma p}{\rho}. \tag{3.8}$$

In the absence of heat sources the entropy equation for continuous flow variations reduces to

$$T\left(\frac{\partial s}{\partial t} + \vec{v} \cdot \vec{\nabla} s\right) = 0, \tag{3.9}$$

expressing that entropy, s, is constant along a flow path. Hence the Euler equations describe isentropic flows in the absence of discontinuities.

The equations form a set of coupled, non-linear partial differential equations, hyperbolic in nature, and dictated by the convection of wave-like properties. The different forms in which they can be represented are described in Appendix A.

The approach detailed within this work uses the conservative formulation in preference to a non-conservative form for two reasons. Numerical experiments and comparisons show that non-conservative formulations are generally less accurate than conservative ones, particularly in the presence of strong gradients. The second reason, of less importance to the type of flow being modelled, is that numerical source terms which tend to emerge in flows with discontinuities can give rise to large errors and incorrect shock intensities with a non-conservative formulation. Therefore, in order to obtain the correct discontinuities, it has been shown that it is necessary to discretise the conservative form of the flow equations [9, page 240][54].

3.3 Finite Volume Discretisation

The finite volume method discretises the integral form of a conservation law by applying it locally to discrete volumes. The summation of the local conservation laws results in the cancellation of fluxes across internal faces, re-establishing global conservation.

The integral form of the equations expresses an average change over a volume due to the flux through the surface of the volume, thus placing no restrictions upon the topology of the discrete volumes. The advantages of this method are:

- there are no mesh topology nor placement of control point restrictions as with the Finite Difference method;
- the conservative law can be solved directly in physical space, removing the requirement of transformation to and from curvilinear co-ordinates; and
- the spatial co-ordinates of geometrical entities are used only in the calculation of cell volume and face area.

Due to these extra degrees of freedom the determination of the fluxes on control volume interfaces can be calculated by a variety of techniques, resulting in a versatile methodology. This versatility is a main reason for the popularity of the method, and a key factor in its selection for this work.

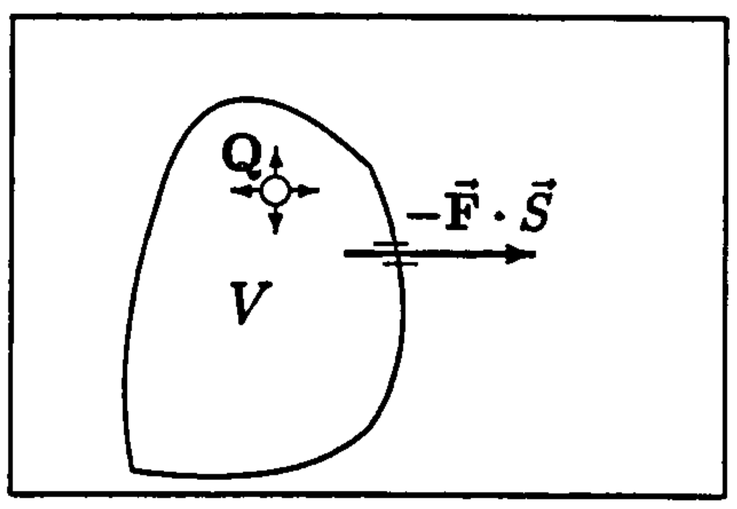


Figure 3.1: Arbitrary Volume

The integral conservation law, written for a discrete volume is

$$\frac{\partial}{\partial t}\int_V \mathbf{U}dV + \oint_S \vec{\mathbf{F}} \cdot d\vec{S} = \int_V \mathbf{Q}dV, \tag{3.10}$$

where V is a volume or arbitrary shape and size, as shown in Figure 3.1, and $\vec{S}$ is the outward facing surface contour. Therefore the time variation of the conservative variables U depends purely upon the surface values of the fluxes and the summation of the source vector. Due to this, Equation 3.10 can be discretised, as shown in Equation 3.11, for a discrete volume V_j.

$$\frac{\partial}{\partial t}(\mathbf{U}_j V_j) + \sum_i \left(\vec{\mathbf{F}} \cdot \vec{S}\right) = \mathbf{Q}_j V_j \tag{3.11}$$

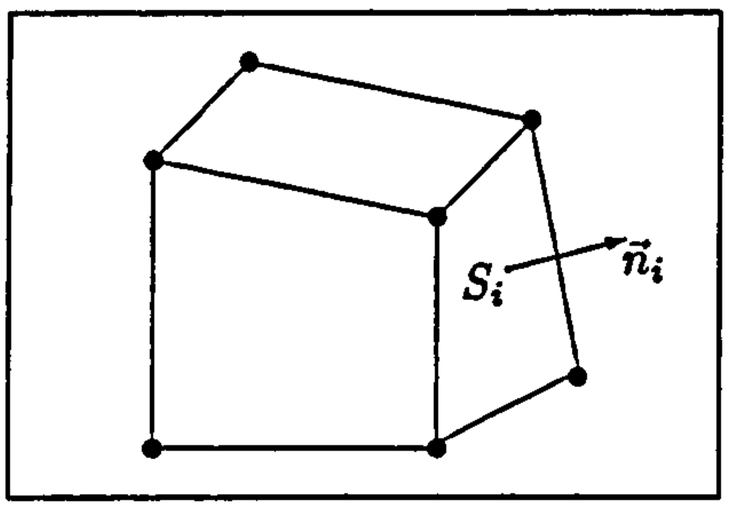


Figure 3.2: Discrete Control Volume

The first term in this formulation defines an average change in the conserved properties within the volume - i.e. the solution is piecewise constant. The following constraints upon the volumes must be satisfied in order to ensure conservation:

- The domain volume must equal the sum of the discrete control volumes;
- Overlapping of adjacent control volumes may only occur if the internal surface is shared;
  and
- Interface fluxes must be evaluated by a control volume independent formulae; this ensures that fluxes on internal faces will cancel.

Due to its generality the finite volume method can handle any type of mesh, structured or unstructured. Differences between these meshes are discussed by Wright and Turnock [46]. The only geometrical definition that distinguishes a scheme is where the state variables are stored. They can either be stored at the vertices of a mesh, in a 'Cell Vertex' scheme, or at the centres of cells, in a 'Cell Centred' scheme, as detailed in Section 2.3.2

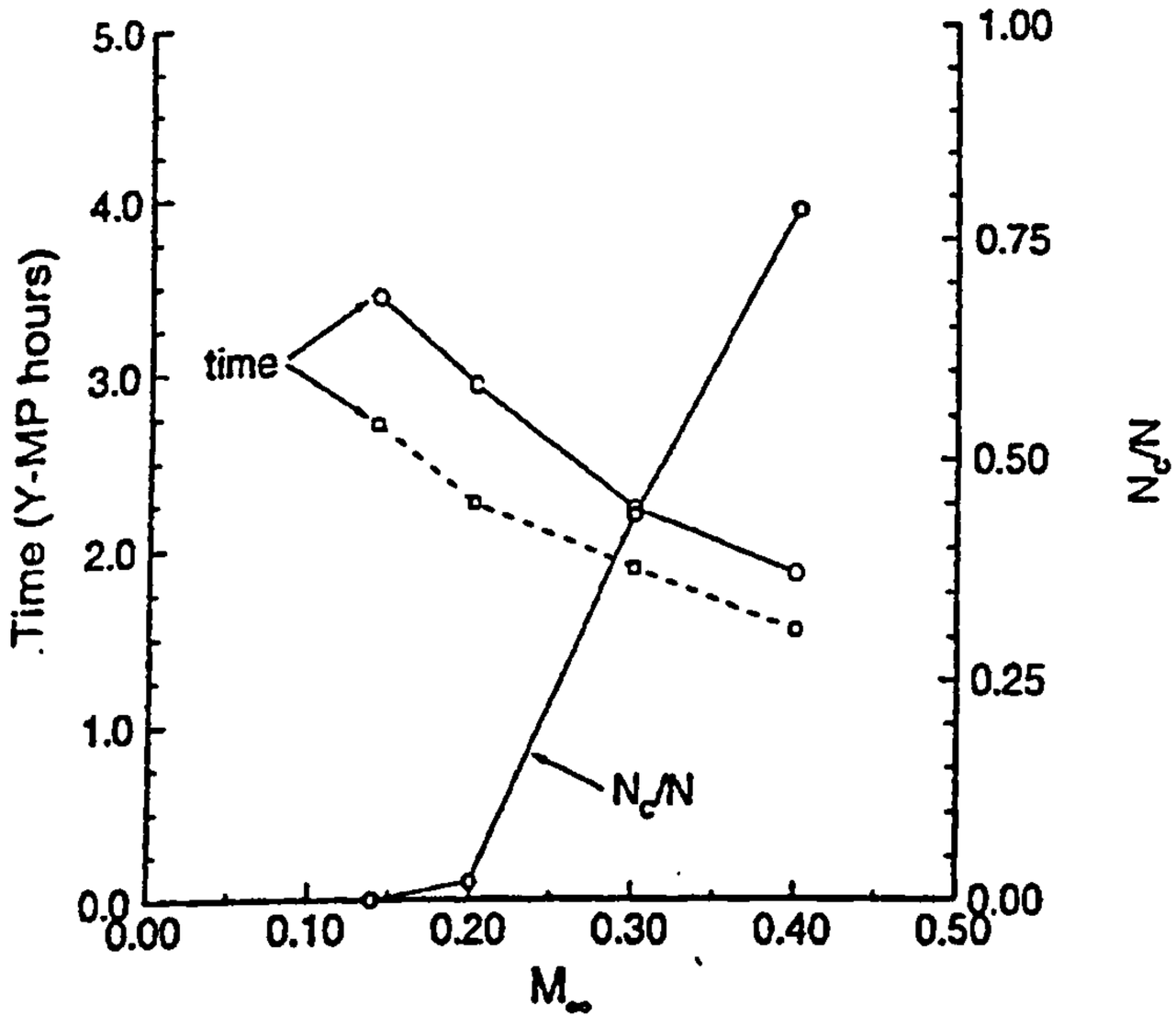


Figure 3.3: Influence of freestream Mach number on rectangular wing solution time. Reproduced with permission from [55], Copyright ©1996 AIAA

3.4 Incompressible Flows

A major problem in the prediction of steady state incompressible flow using the Euler equations is the lack of a time dependant derivative in the continuity equation. This is due to the density having a constant value. In turn, a time-independent constraint must be imposed upon the momentum equations to ensure a divergence free velocity field. An additional problem is that the eigenvalues resulting from the system of Euler equations become infinite in the limit of incompressible flow; this is because for an incompressible fluid the speed of a pressure wave is infinite. Thus the standard methods for computing compressible flows cannot be used. A study by Milholen et al [55] shows the degree to which computational time required to gain a solution as the Mach number decreases. Figure 3.3 is reprinted from this paper, showing the variation of time for the lift coefficient to converge and the number of data points with a Mach number greater than 0.3.

3.4.1 Artificial compressibility

Development of concept

There are methods widely used that take the divergence of the momentum equations and solve implicit equations at each time step for the fundamental variable fields (pressure and velocities) such that continuity is conserved (Hino [56], Miyata et al [57] and Tahara et al [58]). An equation for the pressure can be obtained by taking the divergence of the momentum equation and introducing the divergence free velocity condition leading to a Poisson equation for the pressure for a given velocity field. For typical marine problems with constant viscosity the temperature is decoupled and the basic variables are velocity and pressure. The velocities are calculated from the momentum equations, then corrected to satisfy the continuity equation via the Poisson equation for the pressure. One popular method of this technique is the SIMPLE algorithm, as used by Muzaferija and Perić [26, 52]. This method can be expensive due to the requirement of solving the equations by an iterative method and also calculating the divergence of the momentum equations in a curvilinear co-ordinate system [15, 59].

An alternative method is that suggested by Chorin in 1967 [60]. In the original paper Chorin describes the problems with the current methods, and highlights problems with each one. The incompressible Euler equations are listed as -

$$\begin{aligned} \nabla \cdot \vec{v} &= 0 \\ \frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} &= -\frac{1}{\rho} \nabla p, \end{aligned} \tag{3.12}$$

where $\vec{v}$ is the velocity vector, p is the pressure, and ρ is the density.

Chorin suggested the addition of an artificial time dependent term to the continuity equation. The Euler equations now took the form -

$$\begin{aligned} \frac{1}{\rho_o} \frac{\partial p}{\partial t} + \mathbf{c}^2 \nabla \cdot \vec{v} &= 0 \\ \frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} &= -\frac{1}{\rho} \nabla p \end{aligned} \tag{3.13}$$

ρ_o is the true density and $\mathbf{c}$ is an arbitrary real parameter.

Chorin put forward the hypothesis that if, as the calculation progresses, the solution of Equation 3.13 converges to a steady solution (i.e. one which does not depend on t) then it is a steady solution of Equation 3.12 and does not depend upon **c**, the artificial speed of sound.

Chorin also stated that it was necessary for the artificial Mach number to be below unity for the set of equations to hold.

The rest of Chorin's original 1967 publication is concerned with the creation of a suitable finite difference scheme and verification that the method is applicable. Due to the relatively early publication of the method little computational validation could be undertaken, and indeed the full scope of possibilities for the method remained unrealised.

The next paper of note that develops Chorin's method is Rizzi and Eriksson [61].The method is detailed in a similar manner as by Chorin, but the main application of the method was to rotational flow. This form of flow was chosen because of the widespread presence of vorticity in realistic flows. With potential models this requires prior knowledge of the position of the wake sheet; a complicated procedure for all but the most trivial of situations. The problems of solving an incompressible flow are highlighted, with the emphasis being upon the effect of wave speed upon the stability of the hyperbolic set of equations. The use of artificial compressibility is described as a pseudo-temporal evolution for the pressure which is hyperbolic and which converges to the true steady state value.

The method is described in some detail by Rizzi and Eriksson, the aspects discussed being hyperbolicity in conjunction with curvi-linear co-ordinates and also the value used for the artificial speed of sound, **c**. By matrix manipulation the only variable that the numerical stability was dependent upon was the ratio of $\frac{\mathbf{c}^2}{|\vec{v}|^2}$. From this the condition that $\mathbf{c}^2 = max\,(0.3, r(\vec{v}\cdot\vec{v}))$ with $1 \leq r \leq 5$ has been adopted.

The CFL condition for local stability is also discussed, as are discontinuities. There are several discontinuity situations discussed, but none are given any special significance. More detail of the boundary conditions are given however, and are advanced as being of prime importance. Solid wall and far field boundary conditions are discussed. The far field boundary conditions are specified as a form

of Engquist's hierarchical series of absorbing boundary conditions (Engquist and Majda [62]). This is basically a form of characteristic boundary equations.

The inclusion of artificial viscosity is promoted, due to the need to counteract non-linear aliasing associated with central differencing, to control the cascading of energy from large scale to small scale and also to ensure the existence of a steady state. Rizzi and Eriksson use a linear fourth order differencing term for each of the three spatial directions.

Rizzi and Eriksson verified their model with two dimensional irrotational and rotation and three dimensional rotational case studies. All are discussed thoroughly.

Ramshaw and Mousseau [63] develop the method of artificial compressibility further with the introduction of an accelerated solution via the use of artificial viscosity. Due to sound waves being compressive in nature the most effective method of damping is bulk viscosity. The pressure gradient term of the standard momentum equations ($-\frac{1}{\rho}\nabla p$) is replaced by $-\frac{1}{\rho}\nabla q$, with q as defined in Equation 3.14.

$$q = p - \rho\mathcal{B}\nabla \cdot \vec{v} \tag{3.14}$$

Thus the constant $\rho\mathcal{B}$ is an artificial viscosity term. The value of $\mathcal{B}$ was given a value to minimise the time needed to reach steady state. It is claimed that an artificial speed of sound value of $\mathbf{c^2} = 0.2\frac{\Delta^2}{\Delta t}$ where $\Delta^2 = \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)^{-1}$ gave very good results. The scheme devised by Ramshaw and Mousseau was validated on simple two dimensional test cases; mainly a variety of channel flows. Improvements to the residual reductions are charted, and the conclusions drawn are that variations of this technique should be worthy of consideration in a wide range of computational flow methods.

A more recent publication that develops the concept of artificial compressibility is Farmer, Martinelli and Jameson [51]. The technique is used within a finite volume multigrid Euler scheme for the solution of three dimensional fully non-linear ship wave problems. The adapted Euler equations, equivalent to set of Equations 3.13, are defined as -

$$\frac{\partial \mathbf{U}}{\partial t} + \mathcal{P}\left(\nabla \cdot \vec{\mathbf{F}}\right) = 0, \tag{3.15}$$

where $\mathbf{U}$ is the conservative vector and $\vec{\mathbf{F}} = (\mathbf{f}, \mathbf{g}, \mathbf{h})$ are the flux vectors. The

preconditioning matrix, $\mathcal{P}$, incorporates the artificial compressibility factor, $\mathbf{c}$, as used in set of Equations 3.13.

The choice for $\mathbf{c}$ is taken as $\mathbf{c}^2 = \mathcal{K}(u^2 + v^2 + w^2)$. $\mathcal{K}$ is a constant of the order of unity. This formulation allows the artificial speed of sound to be large in areas of high velocity and low pressure to improve accuracy, while much smaller in regions of lower velocities. This choice of $\mathbf{c}$, it is stated, also greatly improves the behaviour of the boundary condition. Farmer et al discuss the effects of artificial compressibility upon upstream and downstream characteristic boundary conditions. At outflow boundaries the equation characteristics are required to be sufficiently upstream dominant, hence allowing zero gradient extrapolation. The majority of the rest of the publication is concerned with the finite volume scheme used and the coupling with the free surface algorithm.

Results obtained with this formulation were on the whole accurate, with only a little overprediction of the free surface height. It is noted that the pressure residual from the continuity equation gives a good indication of the error of divergence of mass. It is concluded that the method of solution created is accurate and up to ten times more efficient than methods reported in previous literature [51].

The method is presently in widespread use and has been adopted by the majority of projects developed in a recent EPSRC directed programme in marine CFD [64].

Theory

From the formulation of the Euler equations it can be seen that the density term can be made to appear on either the left hand side or the right hand side of the mass conservation equation. That is the conservative term in the mass conservation can be $\frac{p}{\rho}$ or it can be the pressure term alone. This results in changes to the other conservative variables and to all the flux vectors. That is -

$$\frac{p_t}{\rho} + \mathbf{c}^2 u_x + \mathbf{c}^2 v_y = 0, \qquad (3.16\text{-a})$$

or

$$p_t + \mathbf{c}^2 \rho u_x + \mathbf{c}^2 \rho v_y = 0. \tag{3.16-b}$$

This results in the conservative momentum terms being either u and v or ρu and ρv respectively. In Ni's 1981 paper [65] detailing his numerical scheme the latter approach was taken; that is the conservative vector was $\mathbf{U} = (p, \rho u, \rho v)^T$. However Chorin's method when defined by Rizzi and Eriksson [61], places the density upon the left hand side of the equation. With the density being a constant this causes the Jacobian to be of a simpler nature. Farmer et al [51] also utilise this formulation of the Euler equations.

Therefore the definition of the governing equations used in this work is as Equation 3.15 -

$$\frac{\partial \mathbf{U}}{\partial t} + \mathcal{P}\left(\frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} + \frac{\partial \mathbf{h}}{\partial z}\right) = 0 \tag{3.17}$$

$$\mathbf{U} = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix} \mathcal{P} = \begin{bmatrix} \mathbf{c}^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} u \\ u^2 + p \\ uv \\ uw \end{bmatrix} \mathbf{g} = \begin{bmatrix} v \\ uv \\ v^2 + p \\ vw \end{bmatrix} \mathbf{h} = \begin{bmatrix} w \\ uw \\ vw \\ w^2 + p \end{bmatrix}$$

(Rizzi and Eriksson [61])

These values can, and for this work have been, non-dimensionalised by dividing lengths and speeds by a representative distance L and the magnitude of the free stream velocity $|\vec{v}|$ respectively. Pressure is non dimensionalised by $\rho_\infty |\vec{v}|^2$.

By analogy with the equations of motion for a compressible fluid, the pressure can be related to the artificial speed of sound/ compressibility factor, $\mathbf{c}$, by an artificial density, ρ_A. These parameters are related as follows -

$$p = \mathbf{c}^2 \rho_A \tag{3.18}$$

When the temporal derivatives tend to zero the set of equations satisfy precisely the incompressible Euler equations, with the consequence that the correct pressure may be established using the artificial compressibility formulation, as stated by Equation 3.18. It is evident from this that the scheme is not time accurate, due to the non-physical representation of the domain during convergence. A logical conclusion is that local timestepping can be utilised to increase the solver efficiency.

It should be stated that the matrix $\mathcal{P}$ plays an important role in determining convergence rates and stability. $\mathcal{P}$ may be viewed as a device to create a well posed system of hyperbolic equations that are to be integrated to steady state in a manner similar to standard finite volume formulations. The artificial speed of sound can in turn be viewed as a relaxation parameter (Farmer et al [51]). The value of the artificial speed of sound is taken as

$$\mathbf{c}^2 = \mathcal{K}(u^2 + v^2 + w^2), \tag{3.19}$$

where $\mathcal{K}$ is a constant of order unity. Rizzi and Eriksson [61] stated that "when the ratio of $\mathbf{c}^2$ to $|\vec{v}|^2$ is greater than unity the pressure waves dominate .... and the system is better conditioned". Thus it can be assumed that the artificial speed of sound is a locally varying constant, and the constant, K, must be greater than 1 yet small enough "so that convective and acoustic effects occur on similar time scales" (Ramshaw and Mousseau [63]).

If the altered Euler equations, 3.17, are written in a quasi-linear form then the eigenvalues are as stated in Equation 3.20. The full derivation of these values is given in Appendix B.

$$\mathbf{\Lambda} = \begin{bmatrix} \vec{v}\vec{n} & -- & -- & -- \\ -- & \vec{v}\vec{n} & -- & -- \\ -- & -- & \vec{v}\vec{n} + \mathbf{a} & -- \\ -- & -- & -- & \vec{v}\vec{n} - \mathbf{a} \end{bmatrix} \tag{3.20}$$

$$\text{where} \quad \mathbf{a}^2 = \vec{v}\vec{n}^2 + \mathbf{c}^2(n_x^2 + n_y^2 + n_z^2), \tag{3.21}$$

and n_x , n_y and n_z are local normal components in a Cartesian framework.

An important consideration is that the speed of propagation is not the speed of sound, **c**, as with standard compressible flow, but instead **a**. A second con-

sideration is that as stagnation points are reached then $\vec{v}$, c and $\mathbf{a}$ will all tend towards zero with the given formulation. From Chapters 4.1 and 4.2 it can be seen that the speed of propagation term, $\mathbf{a}$, is inversely related to the time step and flux terms. Thus at stagnation points the solution would become highly unstable [66]. Following Rizzi and Eriksson [61] the artificial speed of sound has been modified as follows -

$$\mathbf{c}^2 = \text{MAX}\left(\mathcal{K}(u^2 + v^2 + w^2), \mathbf{c}^2_{\mathbf{MIN}}\right) \tag{3.22}$$

The implementation of this theory in practise, and numerical values used in the pre-conditioning matrix are detailed in Section 4.3

3.5 Summary

The Euler equations, defining incompressible, rotational inviscid flow have been presented, and the theory of discretising a domain into finite volumes for their solution detailed. The technique of artificial compressibility that allows standard time marching techniques to be applied to incompressible flows has also been presented. Chapter 4 develops this theory into a numerical model suitable for computational modelling.

4 Numerical Implementation

This Chapter describes the numerical implementation of the theory defined in Chapter 3, detailing the methods of time integration, spatial discretisation and the validation of this method for two representative case studies.

4.1 Time Integration

The full Euler equations have hyperbolic characteristics, which lends them to time marching procedures where numerical waves are naturally convected. Both explicit and implicit schemes can be used. Explicit methods the matrix of unknown variables at the new time is a diagonal matrix, while the right hand side of the system is dependent only on the flow variables at the previous times. This therefore leads to a trivial matrix inversion and a solution with a minimal number of aritmetric operations. With implicit methods the matrix is not diagonal because more than one set of variables are unknown at the same time level.

Due to their use of purely known data points explicit schemes can be easily implemented whereas implicit schemes require a more complex solution method, typically some form of iterative or direct matrix solver. Implicit schemes are often used however because of stability considerations. Explicit schemes are always conditionally stable but implicit schemes have less conditions, and often are unconditionally stable. That is, the time step in explicit schemes is closely linked to the size of the mesh, usually resulting in a very small time step. Implicit schemes on the other hand can have much larger timesteps and hence reach a converged solution sooner.

Due to their relative ease to program, and ability to allow more efficient algorithms, an explicit scheme has been used in this work. An additional reason for the use of an explicit scheme is because of the perceived difficulties of implementing an efficient matrix solver upon an arbitrary polyhedral topology [20].

4.1.1 Runge Kutta time stepping

As stated in Section 3.4.1, artificial compressibility destroys time accuracy, so a time accurate discretisation scheme is not required, nor is one which uses small time steps. Hence one stage explicit methods, such as Euler's method, are not used; instead a multistage method has been implemented, as used by Rizzi and Eriksson [61] and Farmer et al [51]. An explicit fourth order Runge-Kutta scheme is used, as detailed in Equation 4.1.

$$\begin{aligned} U^0 &= U^{n+1} \\ U^1 &= U^0 + \alpha_1 \frac{\Delta t}{V_\Omega} R\left(U^0\right) \\ U^2 &= U^0 + \alpha_2 \frac{\Delta t}{V_\Omega} R\left(U^1\right) \\ U^3 &= U^0 + \alpha_3 \frac{\Delta t}{V_\Omega} R\left(U^2\right) \\ U^4 &= U^0 + \alpha_4 \frac{\Delta t}{V_\Omega} R\left(U^3\right) \\ U^{n+1} &= U^4, \end{aligned} \tag{4.1}$$

where R(U) is the flux integral over the volume, as defined by Equations 4.18 and 4.20, and the coefficients α_i are given as

$$\alpha_1 = 0.15 \quad \alpha_2 = 0.3275 \quad \alpha_3 = 0.57 \quad \alpha_4 = 1.0. \tag{4.2}$$

The maximum allowable time step to maintain a stable scheme requires that the Courant-Friedrichs-Levy (CFL) number for this explicit time marching was taken to be $2\sqrt{2}$. This method was presented by Jameson et al [67] in 1981. A representative distance over which a wave is permitted to travel is found from a volume to area ratio. This combines with a combination of the numerical wave solution propagation speeds to give the allowable time step, as shown in Equation 4.3 below.

$$\Delta t = \omega \frac{CFL \cdot V}{\lambda_x S_x + \lambda_y S_y + \lambda_z S_z}, \tag{4.3}$$

where ω is a safety factor and S_x, S_y and S_z are summations of the directionalised areas over the surface of the control volume V, as defined in Equation 4.4. The directionalised speeds of propagation (λ_x etc) are defined in Equation 4.5.

$$\vec{S} = \frac{1}{2}\sum_{i=0}^{n} |S_i \vec{n}_i| \tag{4.4}$$

$$\begin{aligned} \lambda_x &= |u| + a \\ \lambda_y &= |v| + a \\ \lambda_z &= |w| + a \end{aligned} \tag{4.5}$$

The safety factor, ω, is present to maintain stability during the evolution of the solution. Instabilities tend to occur in areas of poor grid quality, especially highly skewed control volumes, because the timestep limiter allows more than one numerical wave to pass through the volume in a given timestep. Initial studies of the safety factor for the flow over a hump are presented in Figure 4.1. This flow solution has a very simple geometry, yet clearly shows that as ω increases a larger number of instabilities are present in the convergence history. For all other solutions a safety factor of $\omega = 0.65$ was deemed suitable.

In the Euler equations when using artificial compressibility (Equations 3.17) time conservation is lost due to the inclusion of the $\frac{\partial p}{\partial t}$ term in the mass conservation equation. Thus a global timestep, which maintains time accuracy between individual control volumes, is redundant. The scheme is accelerated towards steady state via local time stepping, where each control volume uses the local maximum time-step allowable. Figure 4.2 demonstrates the increased efficiency of such a scheme. These convergence histories were obtained for flow over a hump, as described in detail in Section 4.3.

4.1.2 Pseudo-time

Because the method of artificial compressibility destroys time accuracy neither an implicit nor an explicit time integration scheme will allow unsteady flow calculations. To solve incompressible unsteady flow the normal technique is to use a pressure correction method (Miyata et al [24]). To keep the efficiency of the artificial compressibility method a dual time approach can been adopted. This approach has been described by Gaitonde [68]; an implicit discretisation is used in real time, but at each real time step the solution is marched to steady state

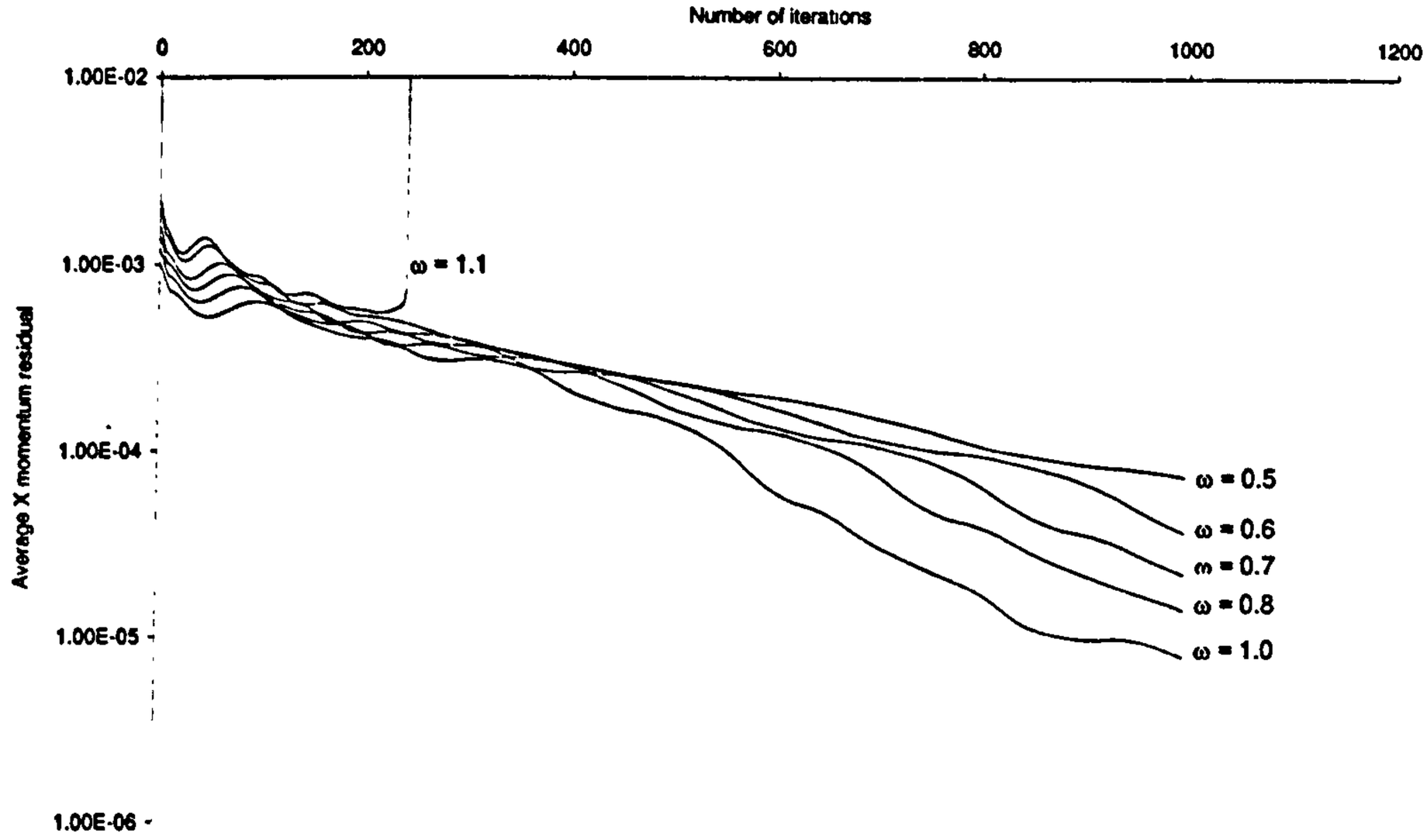


Figure 4.1: Convergence history for incompressible flow over a hump in a channel using $\omega = 0.5, 0.6, 0.7, 0.8, 1.0$ and 1.1

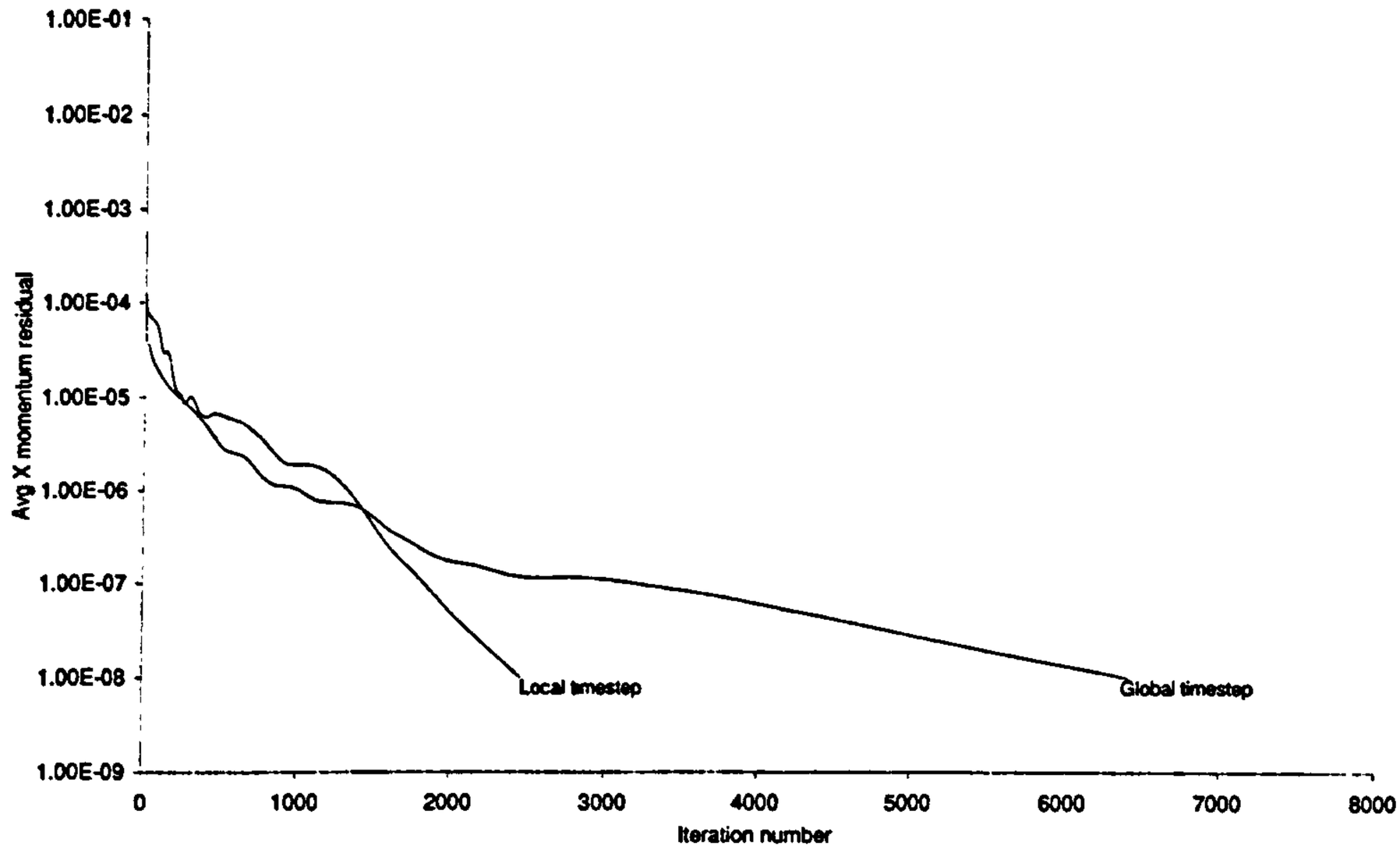


Figure 4.2: Convergence history for incompressible flow over a hump in a channel using global and local timestepping.

through a pseudo-time via the explicit Runge Kutta method defined by Equation 4.1. To differentiate between real time and pseudo-time, t shall be used for real time and τ for pseudo-time. The superscript m is used to denote the real time step and n the pseudo-time step. The result of this is that the Runge Kutta scheme of Equation 4.1 develops into

$$\begin{aligned} U^0 &= (U^{m+1})^n \\ U^k &= U^0 + \alpha_k \frac{\Delta\tau}{V^{k-1}} R^* \left(U^{(k-1)}\right) \\ (U^{m+1})^{n+1} &= U^4, \end{aligned} \tag{4.6}$$

with

$$R^*(U^k) = \frac{3V^{m+1}U^k - 4V^m U^m + V^{m-1}U^{m-1}}{2\Delta t} + R^{m+1}(U^k). \tag{4.7}$$

With the introduction of this extra dimension, the stability criterion for the pseudo-time step, $\Delta\tau$, becomes

$$\Delta\tau = min\left[\omega \frac{CFL \cdot V}{\lambda_x S_x + \lambda_y S_y + \lambda_z S_z}, \frac{2\Delta t}{3}\right] \tag{4.8}$$

It should be noted that the volume terms are dependent upon the real time position (denoted by the superscript m) due to the presence of moving boundaries such as a deforming free surface or the movement of a vessel. Due to the implicit nature of the real time stepping procedure the step size, Δt, can be chosen purely upon accuracy considerations.

4.2 Spatial Discretisation

The simplest spatial discretisation schemes are based on central differencing and have a symmetry with respect to a change in sign of the Jacobian eigenvalues which does not discriminate between downstream and upstream influences. As a result the propagation of disturbances along characteristics is not numerically modelled. Upwind schemes introduces the physical properties of the flow into the numerical discretisation. A problem with second- (and higher) order central schemes is the generation of oscillations in the locality of disturbances which require artificial viscosity to be damped. The construction of schemes which take into account the physical properties of the fluid and equations aim to prevent such undesired oscillations. The first level of introduction of physical characteristics

is termed flux vector splitting, and introduces information as to the sign of the eigenvalues. The flux terms are split and discretised directionally, according to the direction of the associated propagation speeds.

The higher form of modelling the physical characteristics, flux vector differencing, has not been utilised in this work for a number of reasons. The most obvious is the developmental nature of the work, hence overly complex models have been avoided. Flux vector splitting has been proven to produce good results within the artificial compressibility framework [69, 70] and the principal research topic is refinement of the spatial grid, not the numerical flow model.

4.2.1 Flux vector splitting

In characteristic form, the Euler equations are written as Equation 4.9; see Appendix A or Wright and Turnock [71] for details.

$$\frac{\partial \mathbf{W}}{\partial t} + \Lambda \frac{\partial \mathbf{W}}{\partial x} \tag{4.9}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix}$$

The eigenvector, Λ, can be split into a positive and negative matrix, such that

$$\begin{aligned} \Lambda &= \Lambda^+ + \Lambda^- \\ |\Lambda| &= \Lambda^+ - \Lambda^- \end{aligned} \tag{4.10}$$

or

$$\begin{aligned} \lambda &= \lambda^+ + \lambda^- \\ |\lambda| &= \lambda^+ - \lambda^- \end{aligned} \tag{4.11}$$

Λ^+ has only positive eigenvalues, and Λ^- has only negative eigenvalues.

Thus to gain directionality in the conservative form of the Euler equations the eigenvalues must be incorporated in the Jacobian via the transformation matrices between the characteristic and conservative forms. If $\mathbf{K}$ is taken to be the generalised Jacobian of the flux components for the conservative form of the Euler equations (i.e. $\mathbf{K} = n_x A + n_y B + n_z C$ where $A = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ etc.), then eigenvalues can be found from the solution of $det|\mathbf{K} - \lambda \mathbf{I}| = 0$. This can be solved by either solving for $det|\mathbf{K} - \lambda \mathbf{I}| = 0$ or by solving for the eigenvalues of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ separately, and then combining the results. Either way, the resultant eigenvalues are as shown in Equation 3.20.

If $\mathbf{P} = \frac{\partial \mathbf{U}}{\partial \mathbf{W}}$, the transformation matrix between forms of the equations, then the directionalised eigenvalues can be incorporated within the Jacobian, as shown in Equation 4.12.

$$
\begin{aligned}
\mathbf{W_t} + \mathbf{\Lambda W_x} &= 0 \\
\Rightarrow (\mathbf{P^{-1} U_t}) + \mathbf{\Lambda}(\mathbf{P^{-1} U_x}) &= 0 \\
\Rightarrow \mathbf{P P^{-1} U_t} + \mathbf{P \Lambda P^{-1} U_x} &= 0 \\
\Rightarrow \mathbf{U_t} + (\mathbf{P \Lambda P^{-1}}) \mathbf{U_x} &= 0
\end{aligned}
$$

$$
\Rightarrow \mathbf{K} = \mathbf{P \Lambda P^{-1}} \tag{4.12}
$$

It should be noted that the system of equations maintains hyperbolicity if the eigenvalues are real and the norms of the transformation matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ are uniformly bounded for arbitrary real n_x, n_y and n_z.

The matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ can be gained from the asymmetry of the $\mathbf{K}$ matrix; there exists a set of right eigenvectors and a set of left eigenvectors (r^j and l^j respectively) of $\mathbf{K}$ such that the rows of $\mathbf{P^{-1}}$ are equal to the left eigenvectors and the columns of $\mathbf{P}$ are equal to the right eigenvectors.

$$\begin{aligned} \text{i.e.} \quad (l_i)^j \mathbf{K}_{ik} &= \lambda_j (l_k)^j \ , \ \textit{summation in } i \\ \text{and} \quad \mathbf{K}_{ik} (r_k)^j &= \lambda_j (r_i)^j \ , \ \textit{summation in } k \end{aligned} \tag{4.13}$$

From this an upwind formulation can be obtained with the Jacobians

$$\begin{aligned} \mathbf{K}^+ &= \mathbf{P}\mathbf{\Lambda}^+\mathbf{P}^{-1} \\ \mathbf{K}^- &= \mathbf{P}\mathbf{\Lambda}^-\mathbf{P}^{-1} \\ \text{with} \quad \mathbf{K} &= \mathbf{K}^+ + \mathbf{K}^- \end{aligned} \tag{4.14}$$

Hence one has

$$\begin{aligned} & \mathbf{F} \\ = \ & \mathbf{F}^+ + \mathbf{F}^- \\ = \ & \mathbf{K}^+\mathbf{U} + \mathbf{K}^-\mathbf{U} \\ = \ & \mathbf{P}\mathbf{\Lambda}^+\mathbf{P}^{-1} + \mathbf{P}\mathbf{\Lambda}^-\mathbf{P}^{-1} \end{aligned} \tag{4.15}$$

As stated by Equation 4.13, the matrices $\mathbf{P}$ and $\mathbf{P}^{-1}$ can be gained from these eigenvalues combined with the matrix $\mathbf{K}$. By Equations 4.12 and 4.15, the split flux vectors $\mathbf{F}^+$ and $\mathbf{F}^-$ can be found. The derivation of these matrices, for both stationary and moving meshes, can be seen in Appendix B

A problem of using such a flux vector splitting scheme is that its stability limits are defined by the eigenvalues of $|\mathbf{F}|$, and *not* by the Jacobian eigenvalues of K, as specified in Equation 4.3 [15]. A more detailed description of the derivation of the true stability limit via Von Neumann analysis is presented in Appendix D. The method of calculating the maximum allowable time step has not been altered as this non-linear stability limit is much more time consuming to calculate and previous numerical experiments have shown the stability limit corresponds to ≈ 0.85 of the Jacobian limit [15]. This value has been successfully incorporated within ω, the safety factor.

4.2.2 First order discretisation

If the flux terms of the conservative form of the Euler equations (Equation 3.17) are split according to the direction of the characteristics, then

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^+}{\partial x} + \frac{\partial \mathbf{F}^-}{\partial x} = 0 \tag{4.16}$$

Taking the one dimensional problem as a case study, if $\mathbf{F}^+$ is discretised with a backward difference scheme and $\mathbf{F}^-$ discretised with a forward difference then the concept of the upwind scheme will be realised. For simplicity the subscript i defines the discrete spatial position while the superscript n defines the timestep.

$$\begin{aligned}
\mathbf{U}_i^{n+1} - \mathbf{U}_i^n &= -\frac{\partial t}{\partial x}(\mathbf{F}_i^+ - \mathbf{F}_{i-1}^+)^n - \frac{\partial t}{\partial x}(\mathbf{F}_{i+1}^- - \mathbf{F}_i^-)^n \\
\Rightarrow \Delta \mathbf{U}_i^n &= -\frac{1}{2}\frac{\partial t}{\partial x}\{\mathbf{F}_{i+1} - \mathbf{F}_{i-1}\} + \frac{1}{2}\frac{\partial t}{\partial x}\{|\mathbf{F}_{i+1}| - 2|\mathbf{F}_i| + |\mathbf{F}_{i-1}|\} \\
\Rightarrow \Delta \mathbf{U}_i^n &= -\frac{\partial t}{\partial x}(\mathbf{F}_{i+\frac{1}{2}}^* - \mathbf{F}_{i-\frac{1}{2}}^*) \qquad (4.17)
\end{aligned}$$

$$\text{where} \quad \mathbf{F}_{i+\frac{1}{2}}^* = \frac{\mathbf{F}_{i+1} + \mathbf{F}_i}{2} - |\mathbf{A}|\frac{\mathbf{U}_{i+1} - \mathbf{U}_i}{2} \tag{4.18}$$

This scheme is 1^{st} order accurate in both time and space. A higher order scheme (e.g. MUSCL) is not being implemented at this stage for a number of reasons. The developmental stage of the code incorporated with the unstructured nature of the spatial grid and the unstable nature of higher order schemes makes the implementation of such a scheme more difficult. The use of adaptive meshing, as described by Wright and Turnock [46], to some extent negates the increased accuracy of a higher order scheme by allowing local mesh refinement.

Implementation

The calculation of the individual split eigenvalues is accomplished in the same manner as that by Steger and Warming (1986) [72]-

$$\lambda^{\pm} = \frac{\lambda \pm \sqrt{\lambda^2 + \varepsilon^2}}{2} \tag{4.19}$$

The ε is added because of the lack of continual differentiability at zeros of the eigenvalue (i.e. sonic and stagnation points). It should be noted that for the case of artificial compressibility only stagnation points are of interest due to the flow being sub-(*pseudo*)sonic at all times.

Thus we have a first order scheme where the conservative vector and flux vector states at each of the nodes surrounding a face are combined to produce the flux through the face. This is purely one dimensional, but can be expanded into three dimensions by the adoption of the generalised Jacobian matrix $\mathbf{K}$ instead of $\mathbf{A}$ and $\vec{\mathbf{F}}$ instead of $\mathbf{F}$.

The entire flux into a cell is then the summation of the flux through all surrounding faces, such as -

$$\Delta \mathbf{U}_x = -\frac{\Delta t}{\Delta V} \sum_{\alpha=1}^{no.faces} \mathbf{F}^*_\alpha Area_\alpha \tag{4.20}$$

4.3 Validation

To test the convergence, stability and accuracy of the numerical scheme two typical cases were utilised. The first case study was the flow over circular arc hump in a channel and the second was of a NACA 0012 foil at a small angle of attack. The former was chosen because it is a simple flow with easily understood characteristics and solution. The NACA 0012 foil was chosen because is can be easily validated yet contains all the flow properties to be tested; incompressible rotational flow.

The geometrical meshes were all created using the in-house mesh generator *FLEXIMESH*, detailed in Ship Science Report 101, [48] and by Rycroft [20].

4.3.1 Flow over a circular arc hump

Geometric definition

The first test case is a three dimensional representation of a two dimensional problem. The domain is a simple channel of unit height with a circular arc obstruction on its lower surface, centrally placed along the length. The arc's height is 10% of its chord, with the free stream channel height equalling the chord. The upstream and downstream boundaries of the domain are placed 3 chord lengths distant from the beginning and end of the arc respectively. Figure 4.3 shows a wire frame model of the domain.

To test accuracy, stability and convergence capabilities a series of eight meshes were used. Table 4.1 lists the number of nodes on each of the meshes. Detailed results were obtained for the 38×10 mesh and the 268×67 mesh, the meshes for which are shown in Figure 4.4. The geometrical distribution of nodes was maintained for all meshes.

Mesh number	Nodes in X direction	Nodes on hump in X direction	Nodes in Y direction	Nodes in Z direction	Total number of nodes
1	18	6	3	5	270
2	38	12	3	10	1140
3	58	18	3	15	2610
4	78	24	3	20	4680
5	98	30	3	25	7350
6	118	36	3	30	10620
7	178	54	3	45	24030
8	268	80	3	67	53863

Table 4.1: Node distributions for meshes of flow over a 10% hump

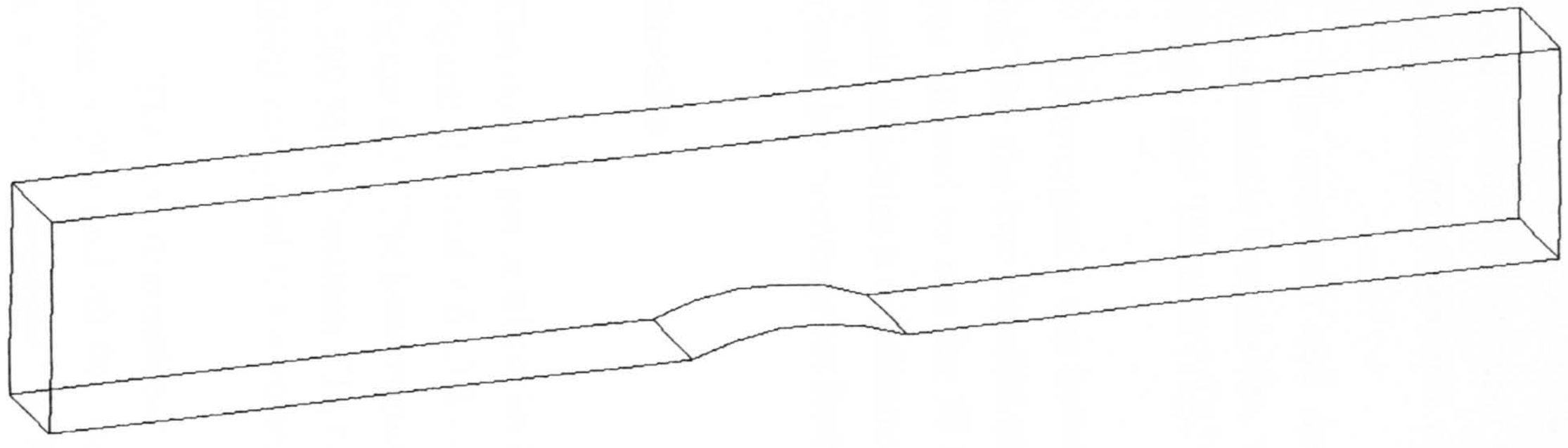


Figure 4.3: Wire frame of hump mesh

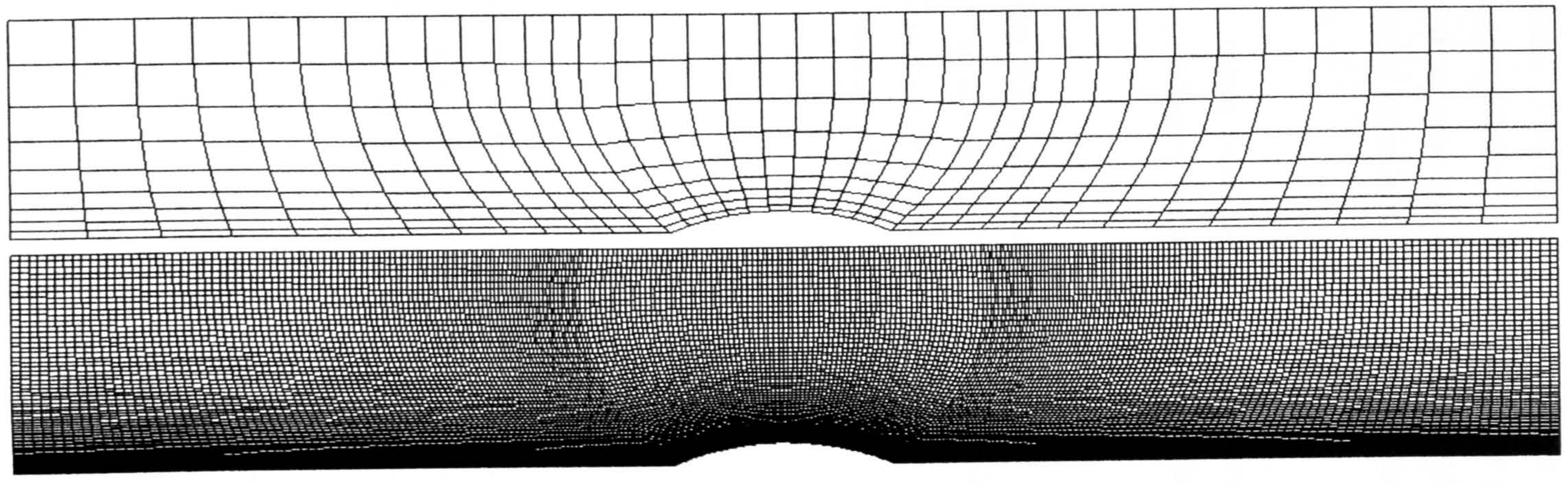


Figure 4.4: Hump meshes utilised to test effects of accuracy, stability and convergence; 38×10 and 268×67 respectively

Flow domain definition

The flow was defined to have a velocity of 5 ms^{-1} in the x direction, with zero velocity components in the y and z directions. The fluid was declared to have a density of 1000 kgm^{-3}. The artificial compressibility constant, $\mathcal{K}$, was given a value of 1.5, and the timestep safety factor, ω, was set to 0.65. The artificial speed of sound limit $\mathbf{c_{MIN}}$ was left at a nominal value of 0.1 because an absolute stagnation point was not expected to occur.

The upstream and downstream $y - z$ plane boundaries were declared as characteristic boundaries, while all other boundaries were defined as walls (see Wright and Turnock [71]).

Convergence was assessed by monitoring the maximum x momentum residual. For the two detailed studies of the 38×10 and 268×67 meshes the solution was allowed to run for 15 000 and 50 000 iterations respectively. For all other mesh densities a maximum X momentum residual of 1×10^{-8} was deemed to be a suitable convergence limit.

Results

The convergence histories for the 38×10 and 268×67 meshes are presented in Figures 4.5 and 4.6. All convergence traces, to a level of 1×10^{-8}, are shown in Figure 4.7. The time required to complete one iteration for each of the meshes on a 500 MHz Pentium III processor, with a Linux operating system fully optimised GNU compiled C++ code, is listed in Table 4.2.

The non-dimensional pressure over the hump surface for the 38×10 mesh after 8 000 and 32 000 iterations, as well as when the maximum residual was 1×10^{-8} is presented in Figure 4.8. The pressure plot for the 268×67 mesh after 21 000 iterations, 50 000 iterations and when the maximum X momentum residual was 1×10^{-8} is presented in Figure 4.9. The pressure plot for the 18×5 and 268×67 meshes are presented in Figure 4.10, as are the pressure maxima and minima for all other flow solutions. The pressure plot for the 268×67 mesh

Mesh number	Nodes in X direction	Nodes in Z direction	CPU time per iteration	CPU time per node per iteration
hump 1	18	5	0.15	5.55×10^{-4}
hump 2	38	10	0.64	5.61×10^{-4}
hump 3	58	15	1.48	5.67×10^{-4}
hump 4	78	20	2.78	5.94×10^{-4}
hump 5	98	25	4.4	5.98×10^{-4}
hump 6	118	30	6.25	5.88×10^{-4}
hump 7	178	45	14.51	6.03×10^{-4}
hump 8	268	67	31.4	5.83×10^{-4}

Table 4.2: CPU time usage for meshes over 10% arc hump

is again presented in Figure 4.11, in this case plotted against results from the potential flow panel code, PALISUPAN, presented by Turnock [73]. This code has been validated previously and is deemed to be a suitable benchmark for this particular case study.

Solutions of the velocity field for the 38×10 and 268×67 meshes are presented in Figures 4.12 and 4.13 respectively. Forces generated by the flow over the hump for all meshes are listed in Table 4.3.

Mesh number	Force in X direction	Force in Z direction
hump 1	0.008159	-0.074621
hump 2	0.007349	-0.115046
hump 3	0.007374	-0.150628
hump 4	0.007232	-0.17789
hump 5	0.007038	-0.198296
hump 6	0.006831	-0.213893
hump 7	0.006242	-0.244247
hump 8	0.005571	-0.26821
PANEL code	0.0000	-0.2695

Table 4.3: Forces imposed upon 10% arc hump for all meshes

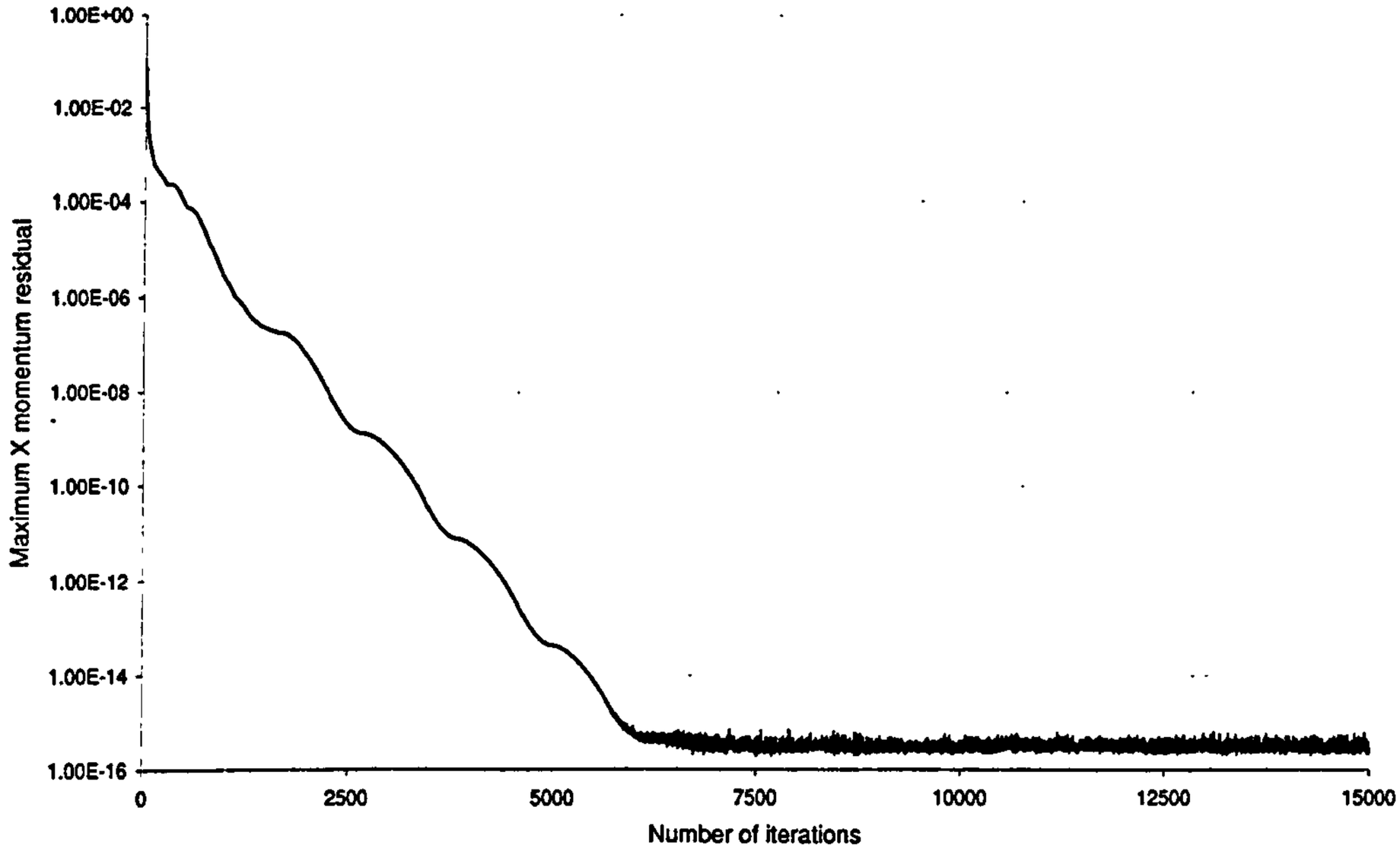


Figure 4.5: Convergence history for 38×10 mesh on 10% arc hump

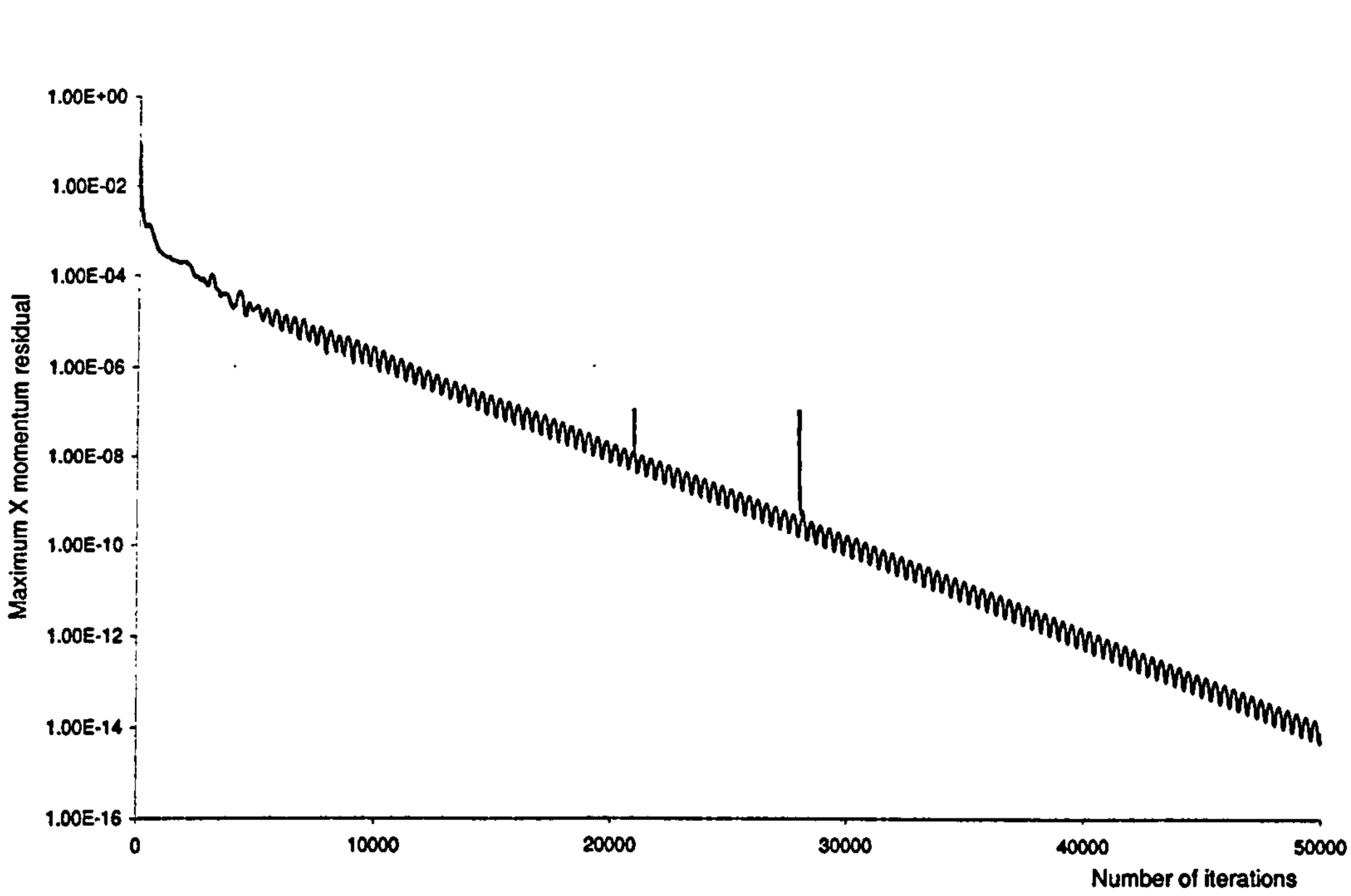


Figure 4.6: Convergence history for 268×67 mesh on 10% arc hump

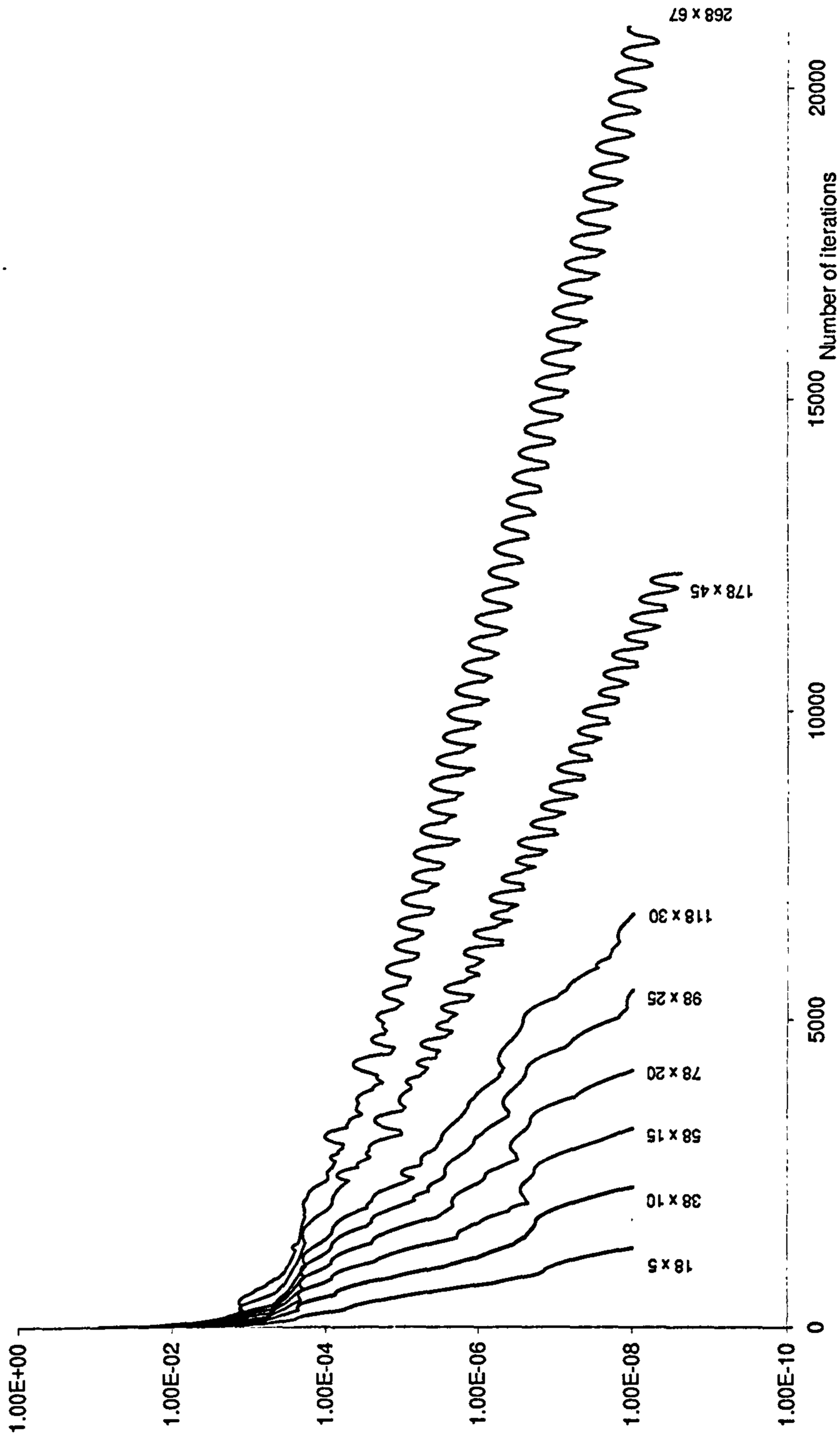


Figure 4.7: Convergence history for all meshes on 10% arc hump

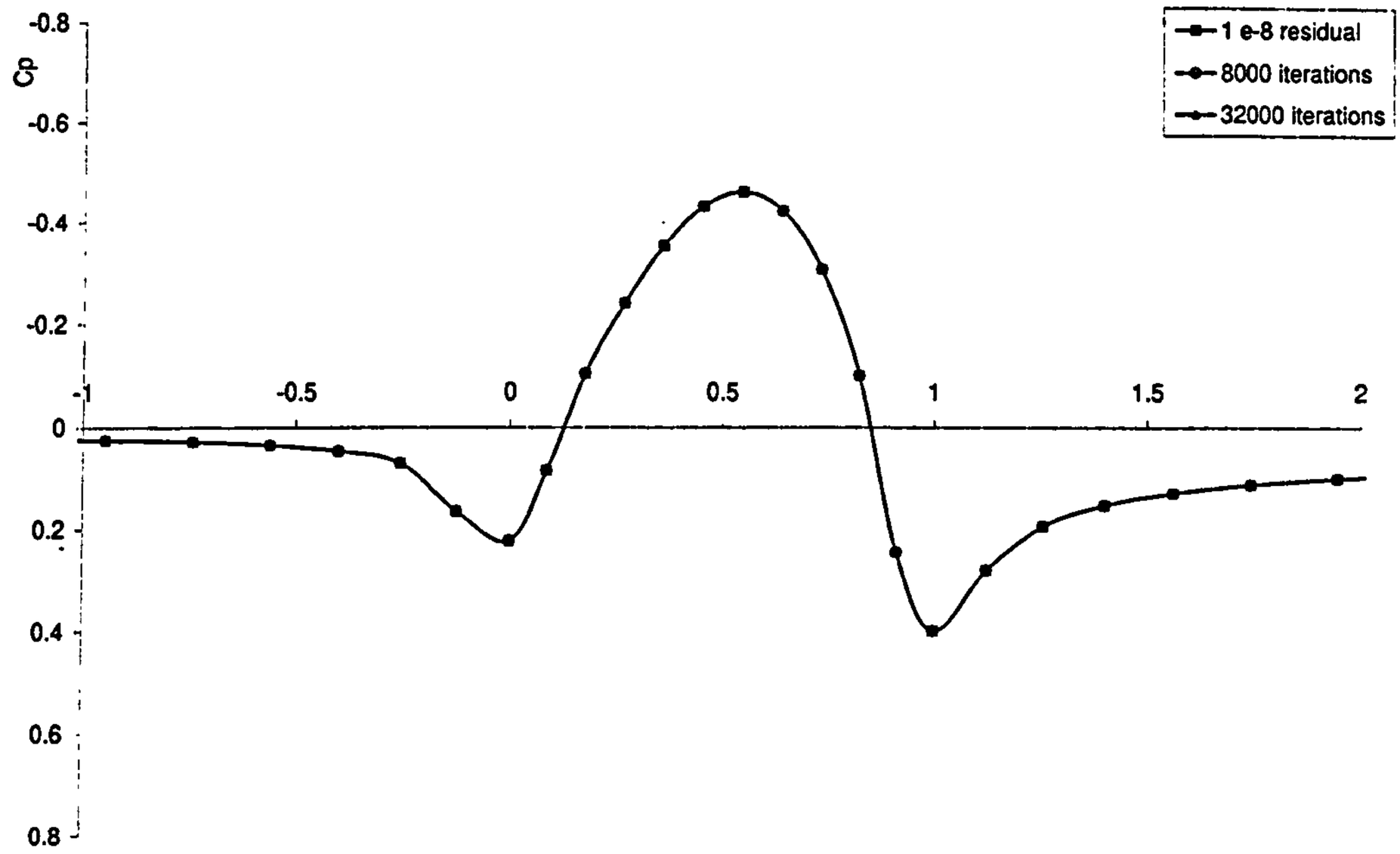


Figure 4.8: Pressure coefficient upon 10% arc hump surface for 38×10 mesh

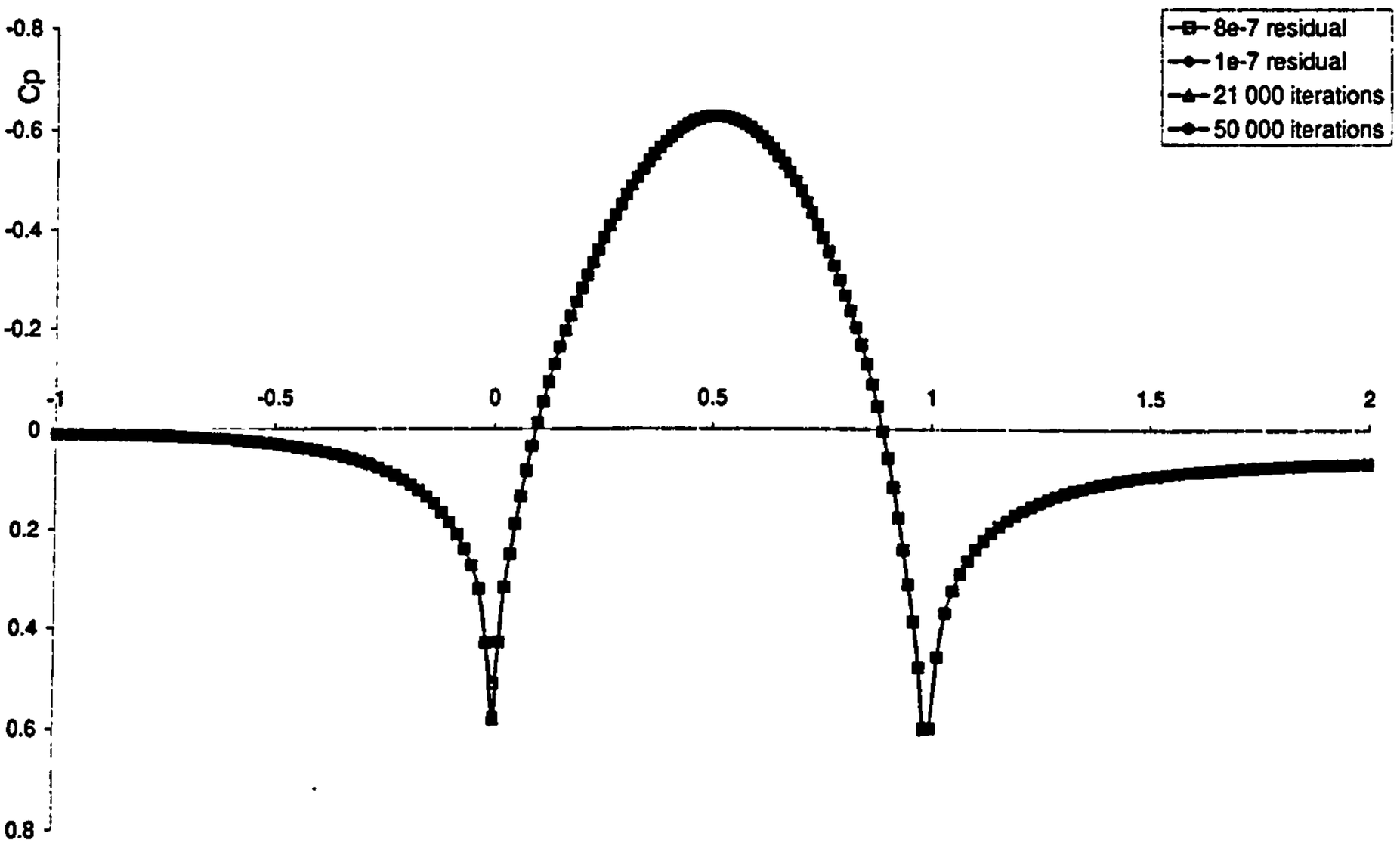


Figure 4.9: Pressure coefficient upon 10% arc hump surface for 268×67 mesh

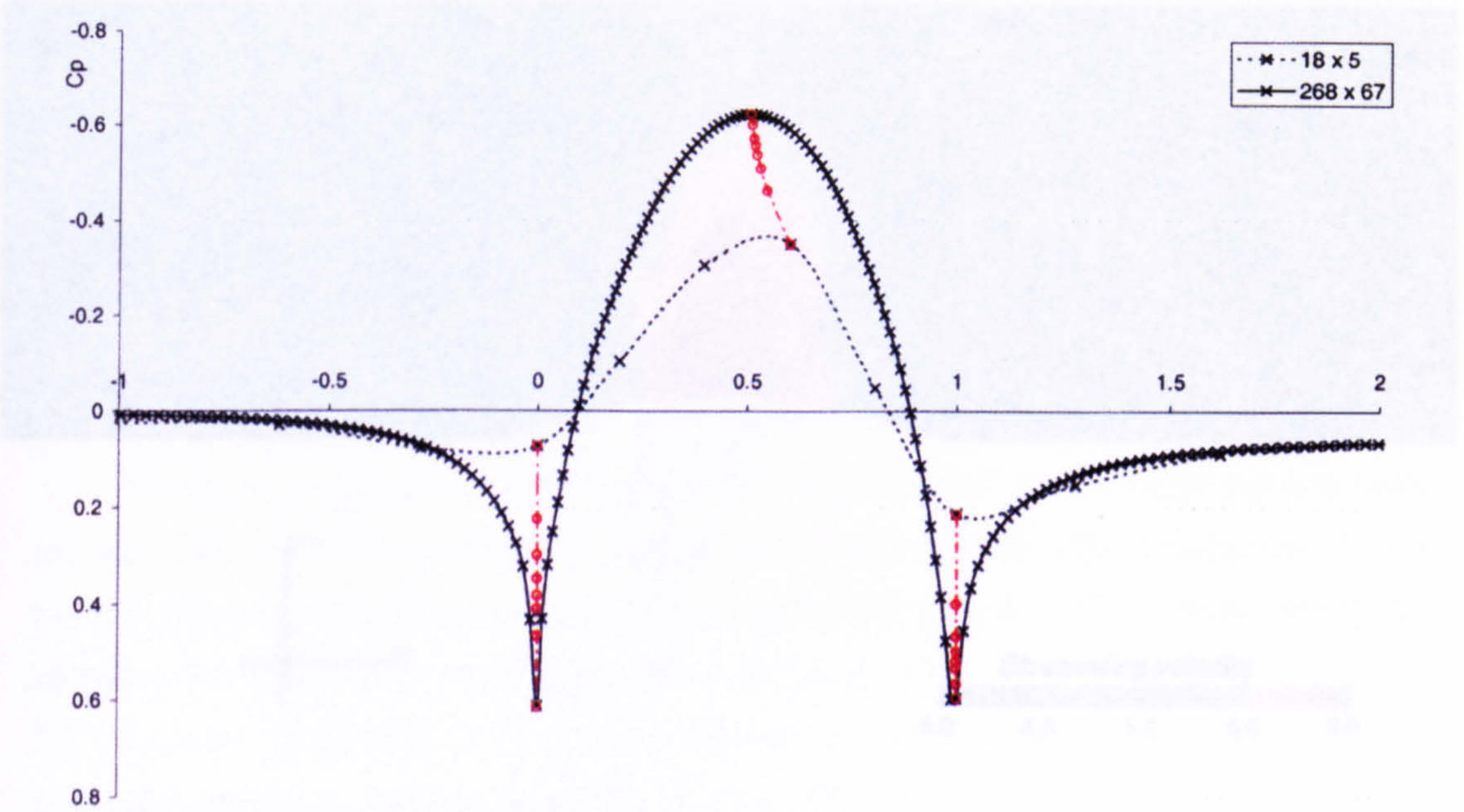


Figure 4.10: Pressure coefficient upon 10% arc hump surface for finest and coarsest meshes, showing pressure maxima and minima for all meshes

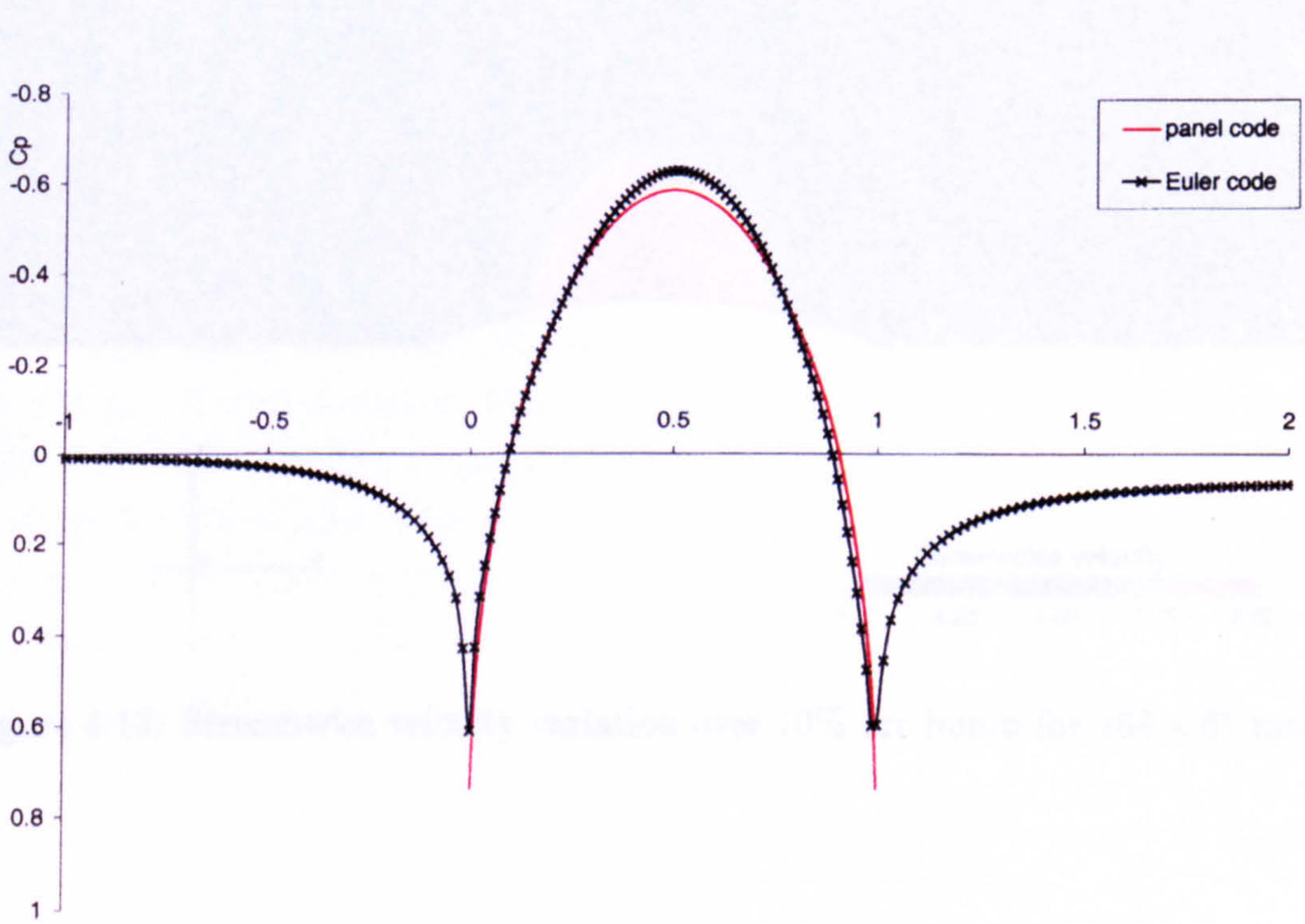


Figure 4.11: Comparison of finest mesh 10% arc hump flow with potential flow code

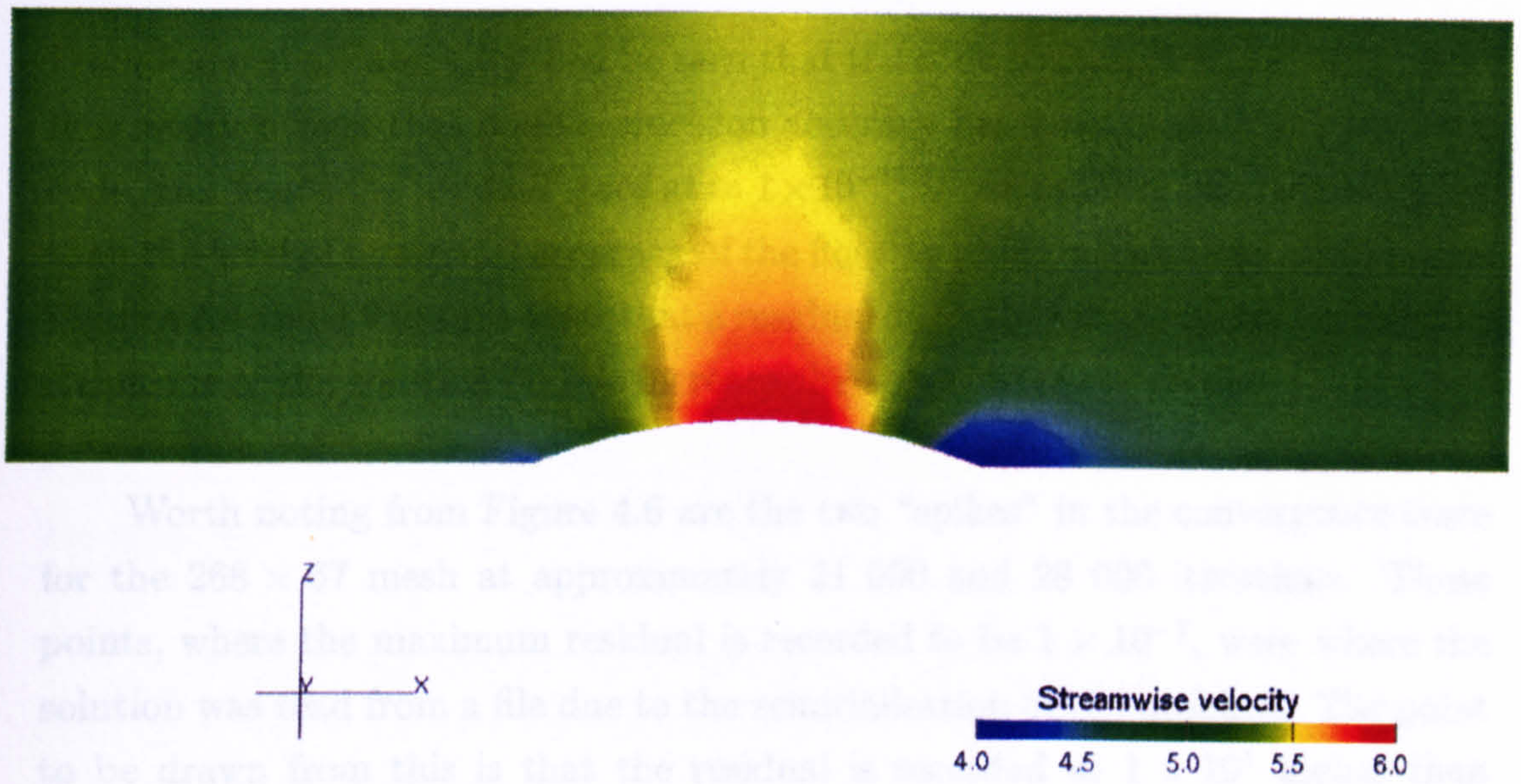


Figure 4.12: Streamwise velocity variation over 10% arc hump for 38×10 mesh

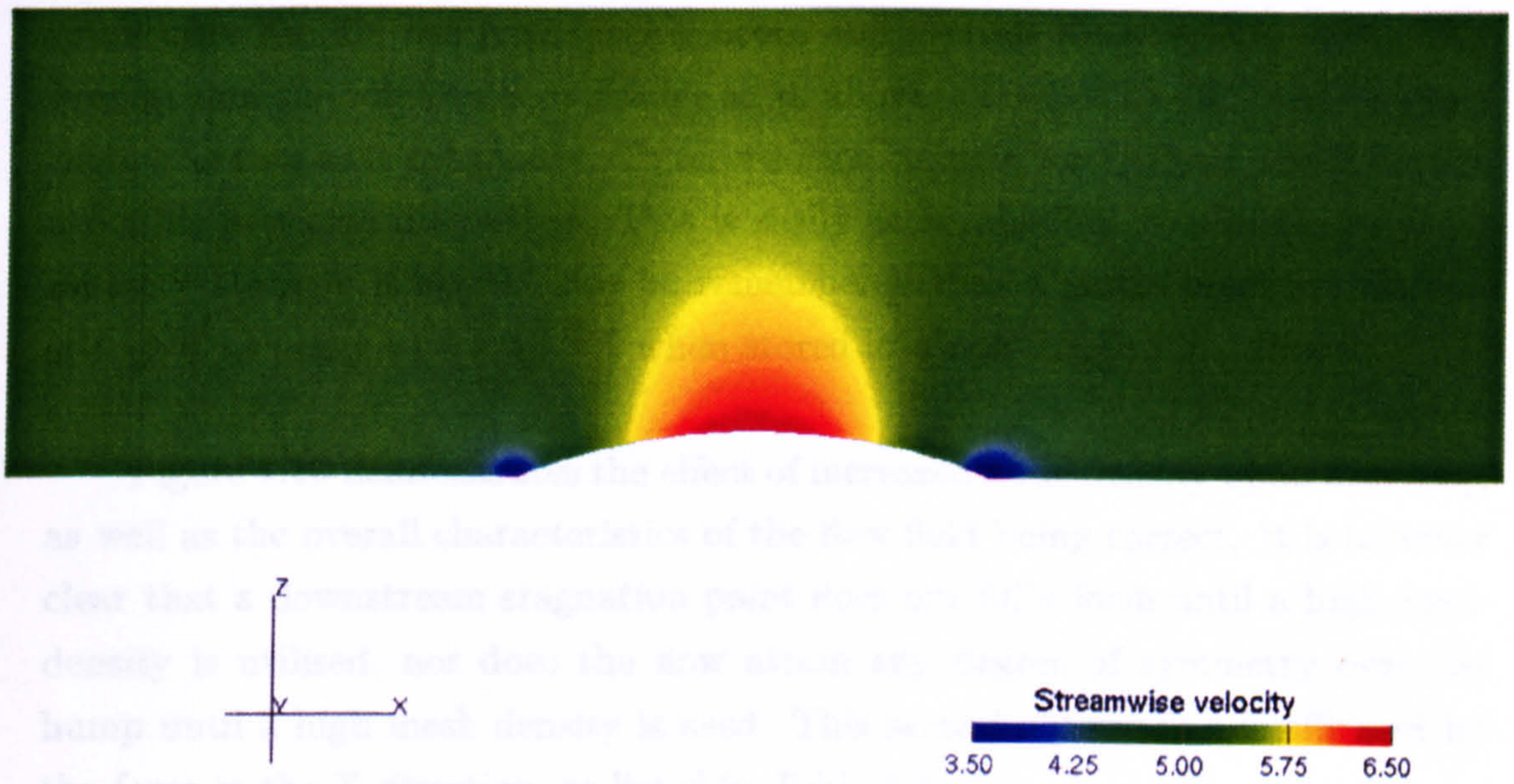


Figure 4.13: Streamwise velocity variation over 10% arc hump for 268×67 mesh

Discussion

From Figures 4.5 and 4.6 it can be seen that the code converges to a steady state. It is worth noting that double precision accuracy has been used throughout the code, and hence the residual trace at $\approx 1 \times 10^{-15}$ is one order of magnitude higher than the level of numerical accuracy of the floating point calculations undertaken. Figures 4.8 and 4.9 in turn show that a residual of 1×10^{-8} is adequate, and further iterations of the solution do not increase the accuracy of the result.

Worth noting from Figure 4.6 are the two "spikes" in the convergence trace for the 268 × 67 mesh at approximately 21 000 and 28 000 iterations. These points, where the maximum residual is recorded to be 1×10^{-7}, were where the solution was read from a file due to the reinitialisation of the problem. The point to be drawn from this is that the residual is recorded at 1×10^{1} higher than the computational accuracy of the file (single precision), as was the converged solution of the 38 × 10 mesh from the computational accuracy of the computer code. Also evident from the convergence trace of Figure 4.6 is that the residual trace *after* the file was read quickly drops and overlies the previous trace. This implies that the solution does change at, or above, a level of 1×10^{-8} and becomes more accurate as it progresses. To record such changes for analysis would require a double precision file output. This is easily achievable but would require extra memory storage; it should also be remembered that a single precision file will still give accuracy to 1×10^{-6}% when stored in a non-dimensional format.

Figure 4.10 demonstrates the effect of increased node density upon accuracy, as well as the overall characteristics of the flow field being correct. It is however clear that a downstream stagnation point does not fully form until a high mesh density is utilised, nor does the flow attain any degree of symmetry over the hump until a high mesh density is used. This second observation is affirmed by the force in the X direction, as listed in Table 4.3.

4.3.2 NACA 0012 foil

Geometric definition

The second test case was again a three dimensional representation of a two dimensional problem, but with the added flow feature of rotationality. A NACA 0012 foil has been placed in a uniform freestream, with the domain conforming to a C mesh. The extents of the domain are 10 chord lengths upstream, downstream, above and below the foil. A wire frame model of the domain is shown in Figure 4.14.

To test accuracy, stability and convergence capabilities a series of four meshes were used. Table 4.4 lists the number of nodes on each of the meshes. Detailed results were obtained for the 123×28 and 593×33 mesh, the meshes for which are shown in Figure 4.15.

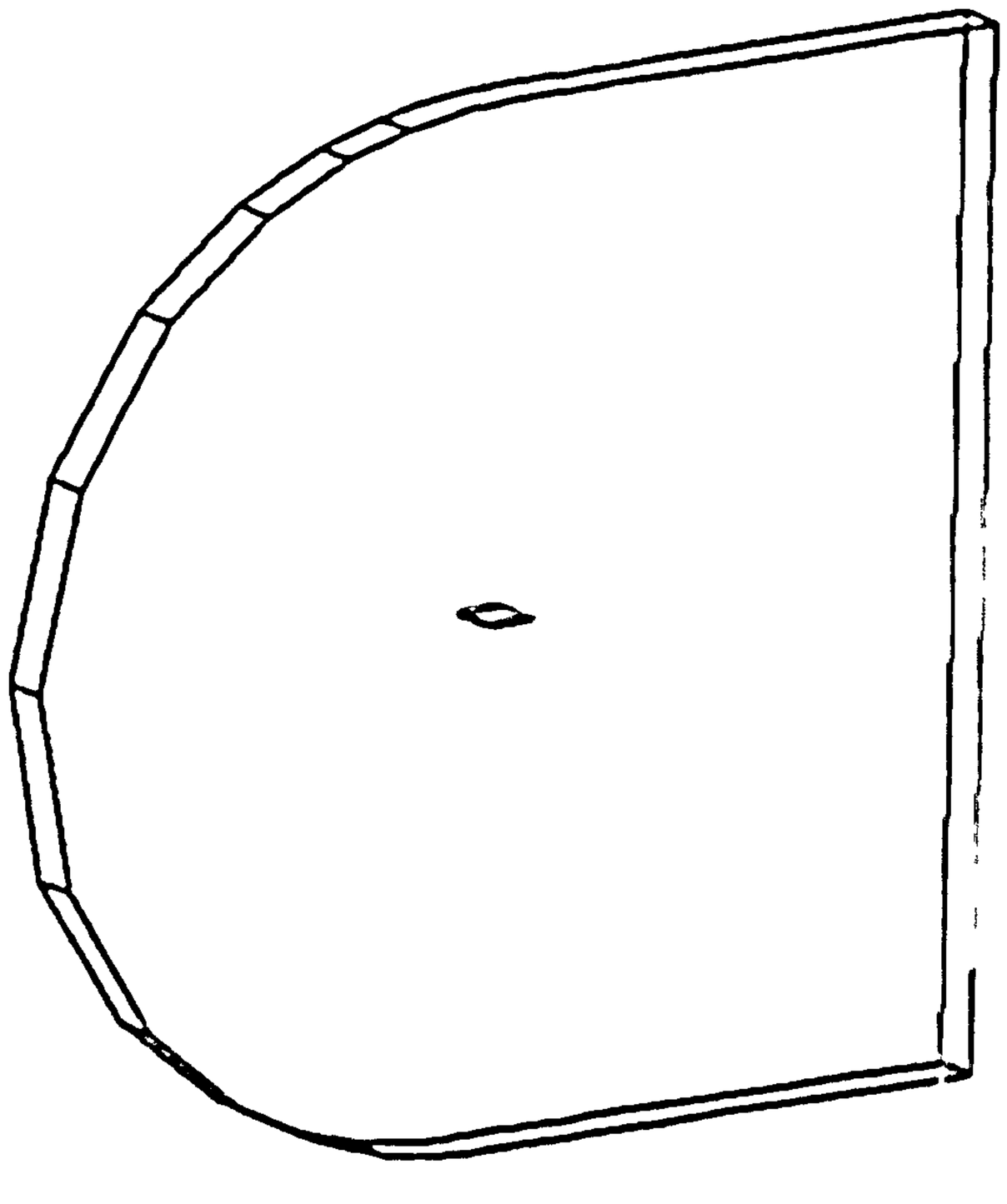


Figure 4.14: Wire frame of foil domain

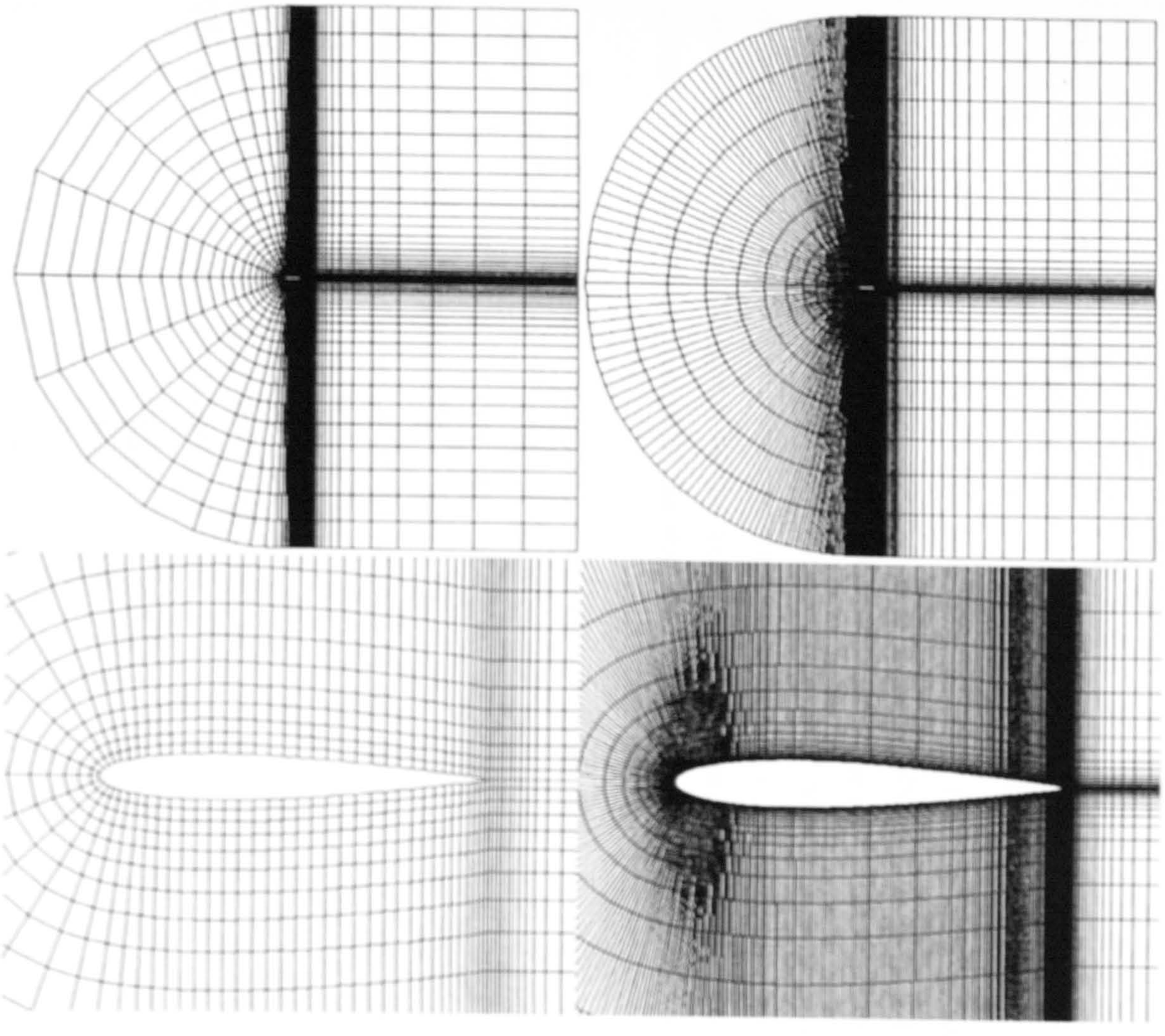


Figure 4.15: 123×28 and 593×33 meshes around NACA 0012 foil

Mesh number	Nodes on foil	Nodes in wake	Nodes normal to foil	Nodes in Z direction	Total number of nodes
foil 1	74	24	28	3	10257
foil 2	245	49	25	3	25575
foil 3	245	49	33	3	33807
foil 4	495	49	33	3	58557

Table 4.4: Node distributions for meshes of flow over a NACA 0012 foil

Flow domain definition

To model a 4° angle of attack while still maintaining a mesh quality standard the inflow velocity was rotated, leaving the foil orientated with the XZ plane axial. The freestream velocity was defined as $5ms^{-1}$, hence the x velocity was $4.988ms^{-1}$ and the y velocity was $0.349ms^{-1}$. The fluid was declared to have a density of 1000 kgm^{-3} while the timestep safety factor, ω, was set to 0.5. The lower value of ω, with respect to the hump flow, has been used because of the greater mesh distortion present at the stagnation point around the tip of the foil.

Because a full stagnation point was developing in this case the artificial compressibility constant, $\mathcal{K}$ and the limit on the minimum artificial speed of sound, c^2_{MIN}, had to be altered from the previous study. Publications upon the use of artificial compressibility vary in their values for c^2_{MIN}, with the only consensus being that c^2_{MIN} cannot be allowed to get very small relative to the general flow. Turkel [66] states that "the reasons for this are still unclear" and "a complete theory .. is quite difficult". As a result a degree of variation was used in the values of $\mathcal{K}$ and c^2_{MIN} until a satisfactorily converged solution was found. The values that were used in each mesh are shown in Table 4.5.

Mesh	$\mathcal{K}$	c^2_{MIN}
foil 1	3.6	1.0
foil 2	3.6	1.1
foil 3	3.6	1.1
foil 4	3.6	1.1

Table 4.5: Artificial compressibility factors for a domain with a stagnation point

The foil surface and the boundaries in the xy plane were defined as walls, while all other boundaries were classified as characteristic boundaries.

Convergence was assessed by monitoring the maximum X momentum residual. For the two detailed studies of the 123×28 and 593×33 meshes the solution was allowed to run for 25 000 and 60 000 iterations respectively. For all other mesh densities a maximum X momentum residual of 1×10^{-8} was deemed to be a suitable convergence limit.

Results

Convergence histories of the maximum X momentum residual for the coarsest mesh can be seen in Figure 4.16, while the convergence traces for all the meshes can be seen in 4.17.

The pressure coefficient over the foil surface for all the meshes can be seen in Figure 4.18 For validation purposes the pressure coefficient over the foil for the densest mesh and experimental data from NPL Aero Report 1308 [74] are compared in Figure 4.19. Contour lines of pressure for the 593×33 mesh can be viewed in Figure 4.20. The forces generated by the foils at 4° are tabulated in Table 4.6, as is experimental data from "Theory of Wing Sections" [75].

Mesh number	C_D	C_L
foil 1	0.021	0.0927
foil 2	0.0164	0.2295
foil 3	0.015	0.2798
foil 4	0.015	0.317
Experimental	0.007	0.45

Table 4.6: Forces imposed upon NACA 0012 foil at 4° angle of attack for all meshes

Discussion

Figure 4.16 clearly shows that like the 10% arc hump case, the solution converges to one of computer accuracy, in a linear manner. The traces for the three denser meshes show a slower rate of convergence due to the increased limiter. This tends to define the artificial speed of sound to be more uniform throughout the domain, and hence slows the overall rate of convergence to the rate of the most unsta-

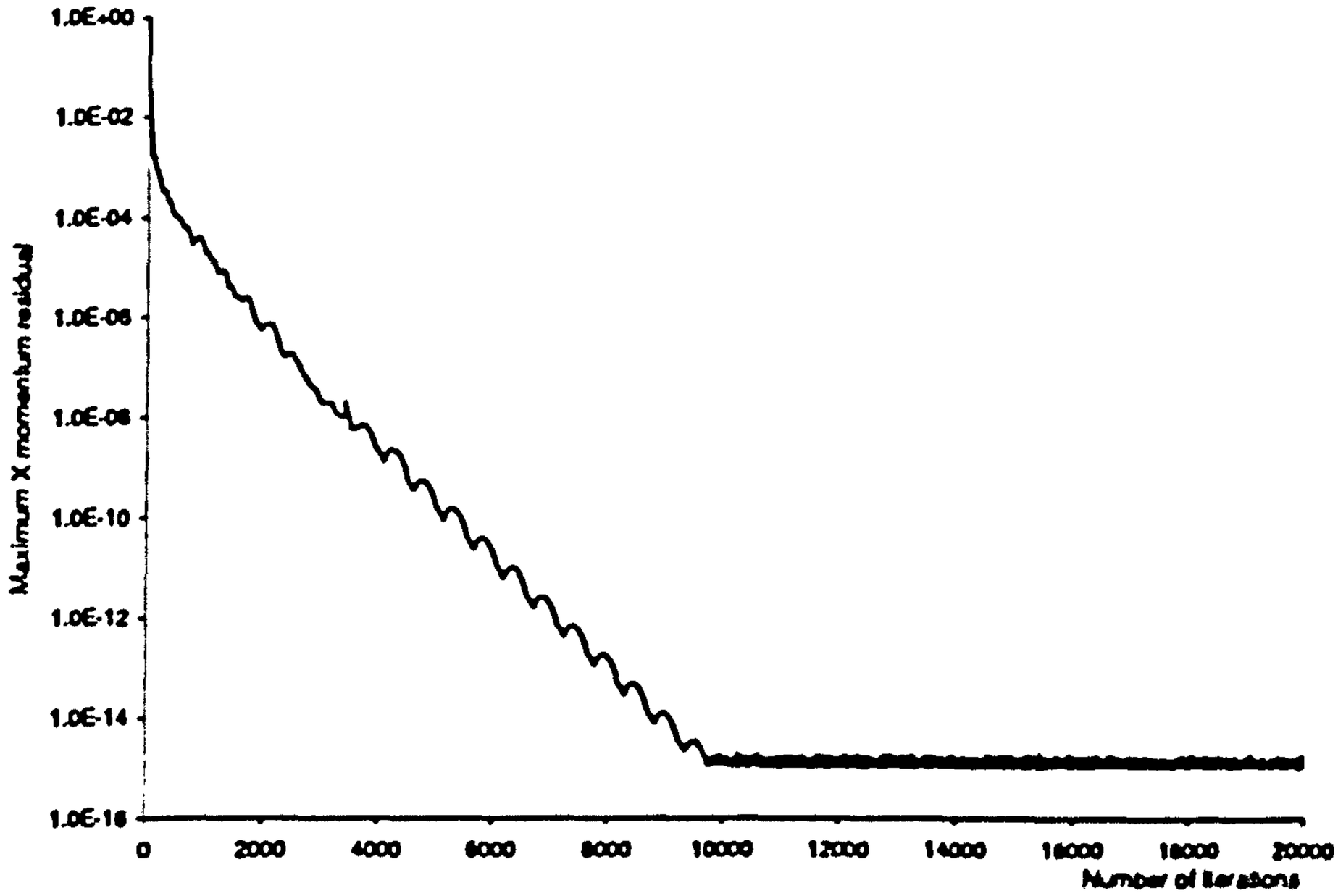


Figure 4.16: Convergence history for coarse mesh around NACA0012 foil at 4 degrees angle of attack

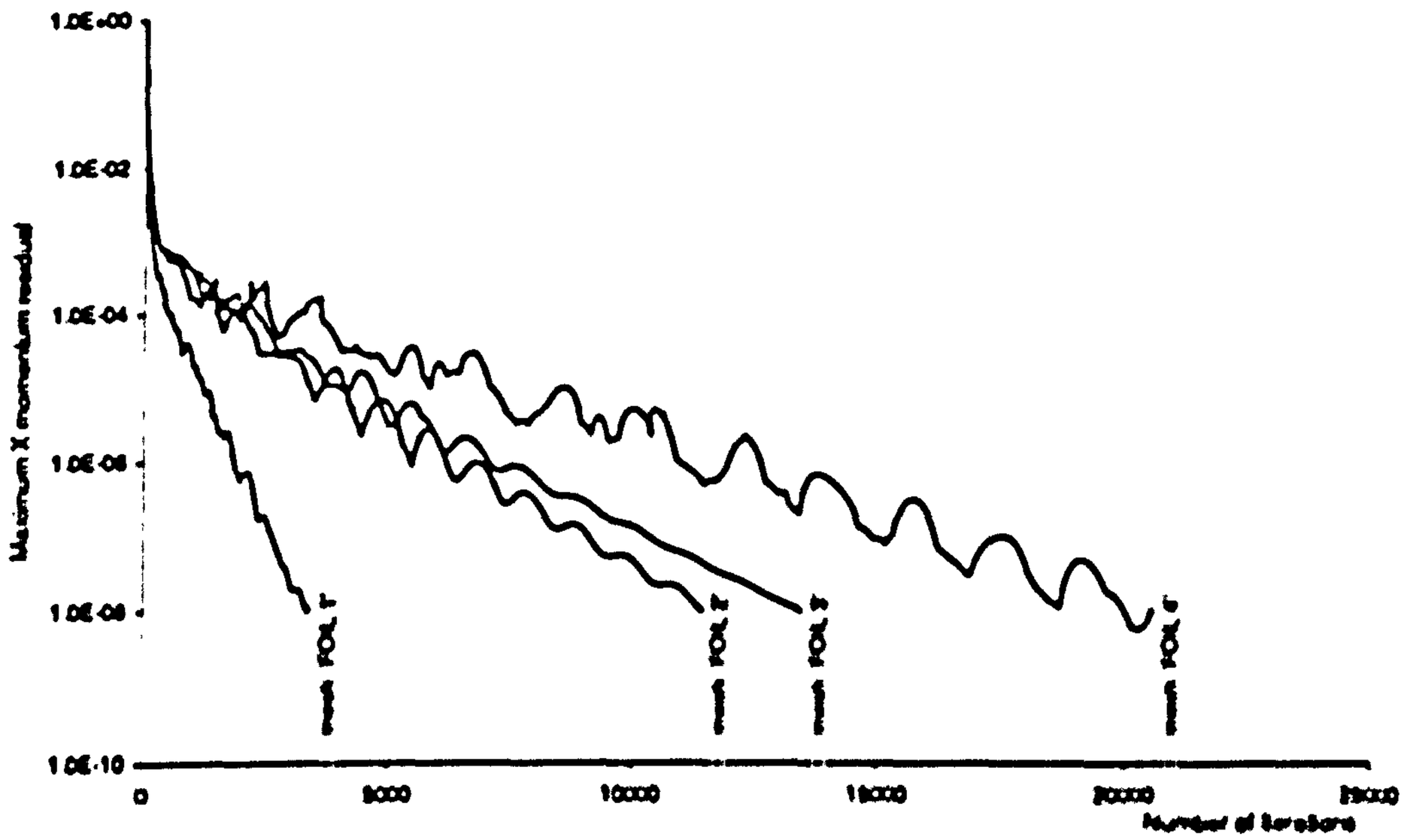


Figure 4.17: Convergence history for all meshes around NACA0012 foil at 4 degrees angle of attack

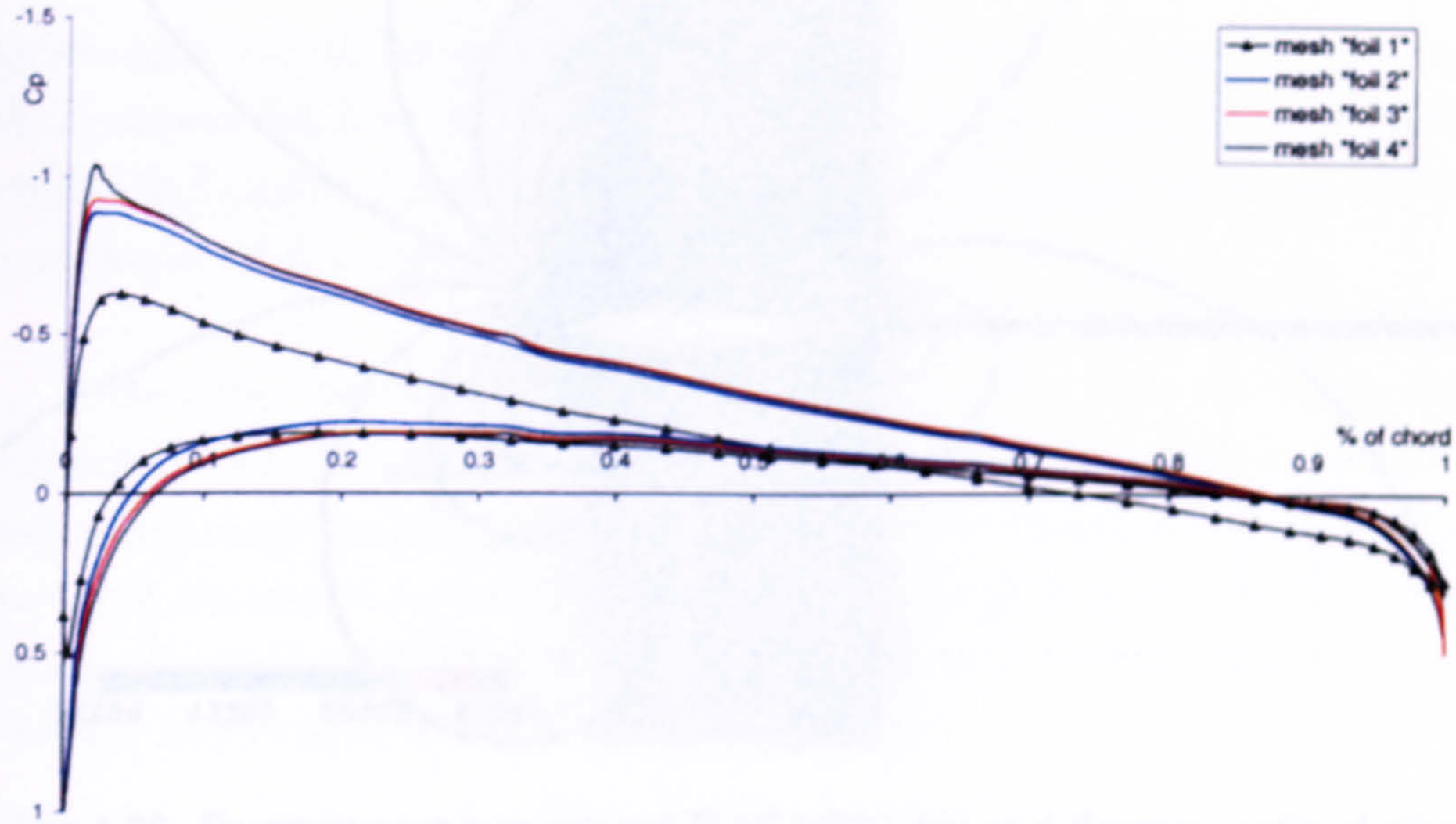


Figure 4.18: Pressure coefficient for NACA0012 foil at 4 degrees angle of attack, for all meshes

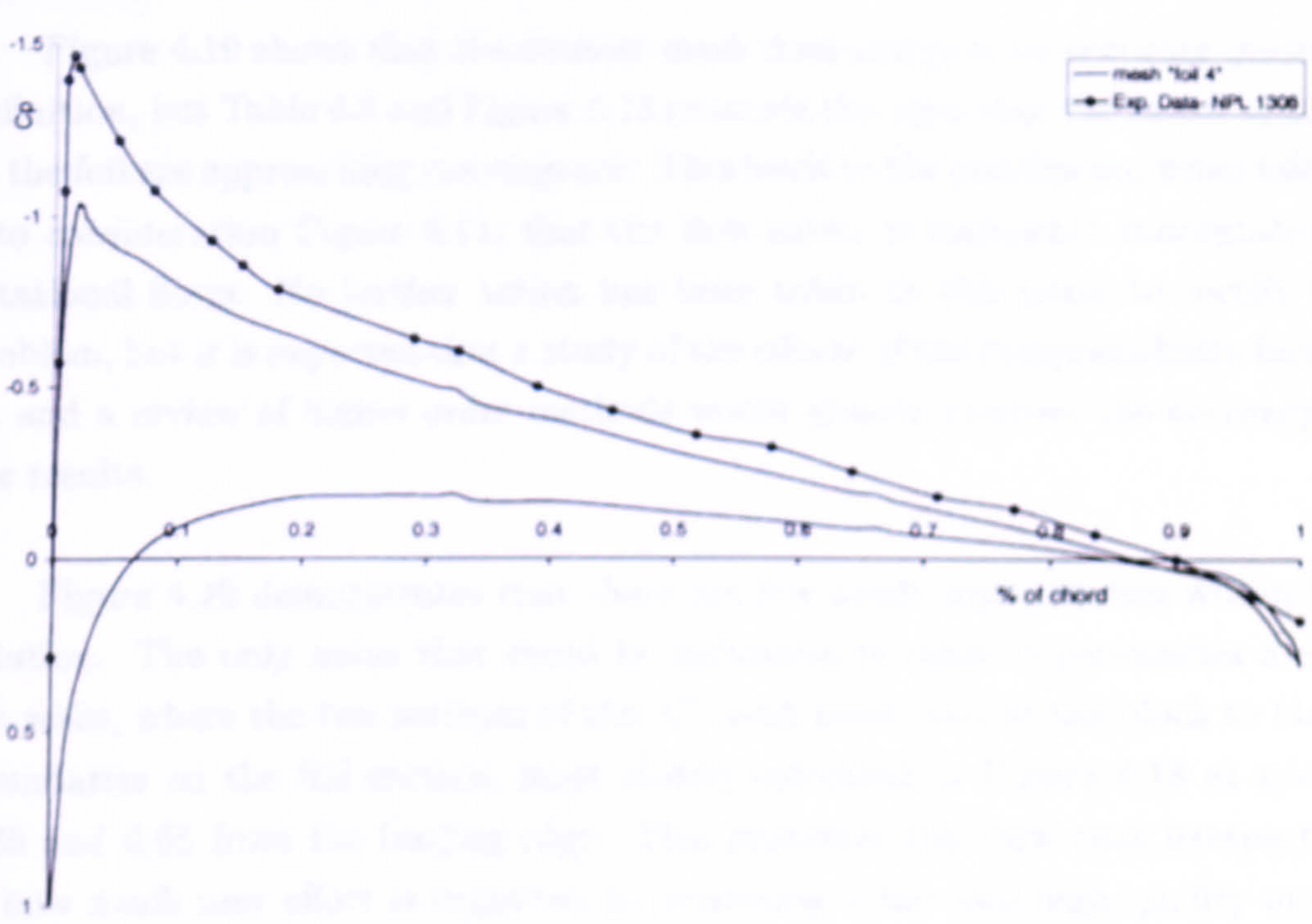


Figure 4.19: Pressure coefficient for NACA0012 foil at 4 degrees angle of attack, compared to experimental data

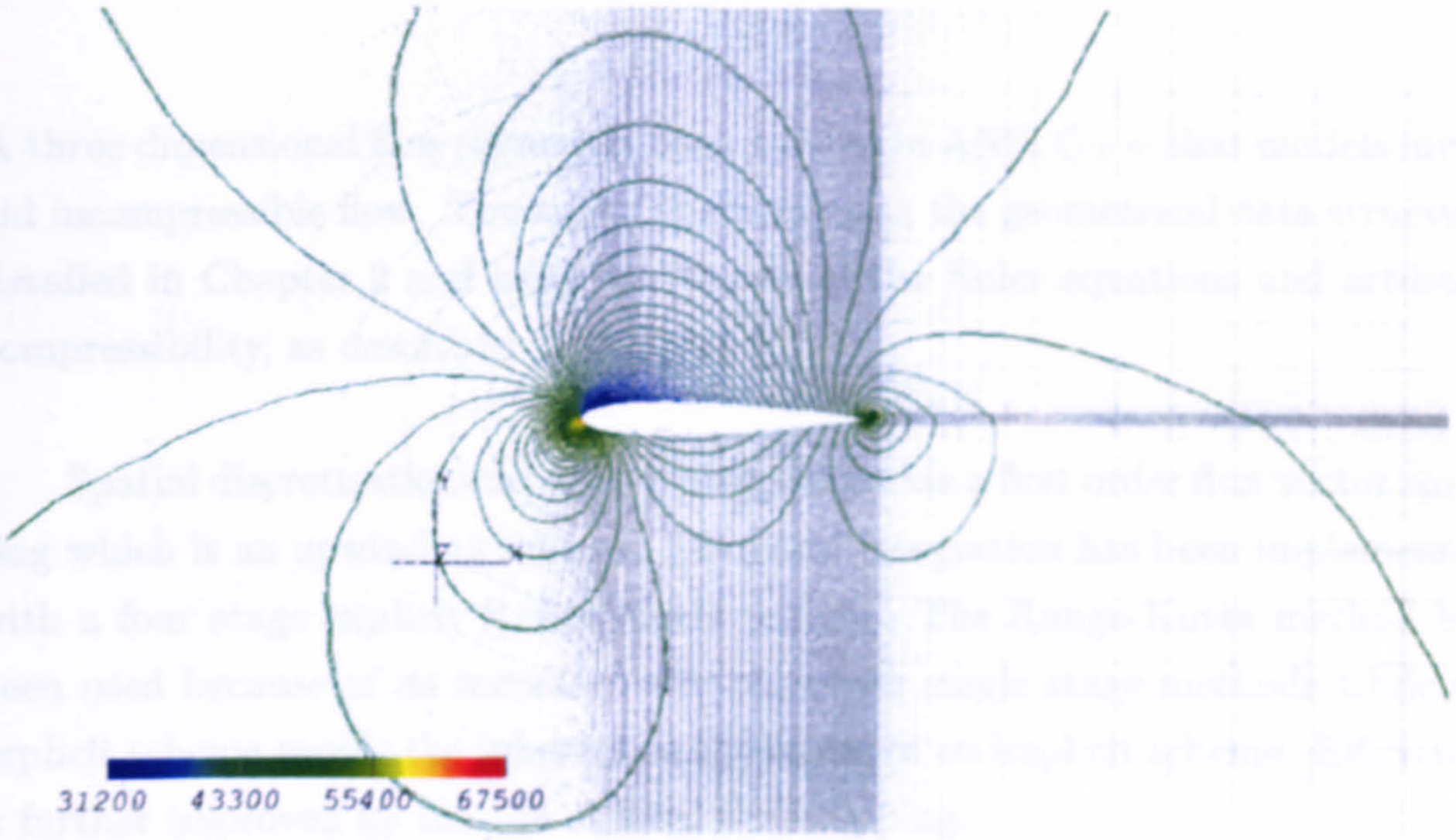


Figure 4.20: Pressure contours around NACA0012 foil at 4 degrees angle of attack for densest mesh

ble region. The increased limiter was used to increase stability of the solution, particularly in areas of stagnation.

Figure 4.19 shows that the densest mesh does not give an accurate pressure definition, but Table 4.6 and Figure 4.18 promote the view that the forces applied to the foil are approaching convergence. This leads to the conclusion, when taking into consideration Figure 4.11, that the flow solver is somewhat inaccurate for rotational flows. No further action has been taken at this point to rectify the problem, but it is expected that a study of the effects of the compressibility factor, $\mathcal{K}$, and a review of higher order methods would greatly improve the accuracy of the results.

Figure 4.20 demonstrates that there are few mesh dependencies within the solution. The only areas that could be indicative of mesh dependencies are in the wake, where the two sections of the 'C' mesh meet, and at the block to block boundaries on the foil surface, most clearly indicated in Figure 4.18 at $x/c = 0.35$ and 0.65 from the leading edge. This promotes the view that irrespective of how much user effort is imparted to producing a fair and high quality mesh, there will always be areas of poor mesh regularity without numerical (and usually automated) mesh distribution.

4.4 Summary

A three dimensional flow solver has been written in ANSI C++ that models inviscid incompressible flow. The solver operates upon the geometrical data structure detailed in Chapter 2 and using the theory of the Euler equations and artificial compressibility, as described in Chapter 3.

Spatial discretisation has been accomplished via a first order flux vector splitting which is an upwinding scheme. Temporal integration has been implemented with a four stage explicit Runge Kutta scheme. The Runge-Kutta method has been used because of its increased efficiency over single stage methods while an explicit scheme avoids the inherent complexities of an implicit scheme. Efficiency is further improved by the use of local timestepping.

This scheme has been verified and validated with two test cases; a circular arc hump in a internal channel and a NACA0012 foil at 4 degrees angle of attack in an external free stream. All solutions show good convergence rates, with a selection being driven to computer accuracy to prove the method is fully converging. Pressure forces converge to experimental and previously validated computational results for increased discretisation of the domain.

More detailed analysis of the error distribution throughout each domain is presented in Chapter 5, and a method of increased efficiency via an improved convergence rate is detailed in Appendix D.

5 Grid Quality Assessment

5.1 Introduction

Qualitative and quantitative measurement of the confidence level with which the results of computational fluid dynamics (CFD) codes can be used has received increasing levels of attention as numerical simulations become more commonly used and depended upon within the design environment [76][77]. For a numerical simulation to be used effectively and efficiently there must be a high degree of trust in the quality of a given solution. Sources of error and uncertainty are numerous, but one of the most commonly identified is that of the geometrical mesh. Paradoxically however, geometrical grid properties have been accorded little mathematical correlation to solution error. The manner of generating this grid is a significant consideration in CFD; the type of grid chosen for a given problem strongly influences whether a solution is possible, the amount of computational effort required, and the spatial accuracy of the solution [20].

To clarify the problem the definition of quality must first of all be stated. "Quality" is "a distinctive attribute or faculty" or "the relative nature or kind or character of a thing", while "quality control" is "a system for maintaining standards ... against the specification" [78]. In this case the object is the discrete mesh, and the acceptable levels are the accuracy of the final flow solution and the resources required (be they time, computing facilities or technical ability) to gain that solution. Thus a linking of the grids physical properties and the flows properties must be created. From this a direct relationship between the grid's physical characteristics and the flows accuracy can be derived.

The spatial accuracy of a numerical solution is commonly termed the discretisation error. Strictly speaking this term only covers the local truncation error from the use of discrete piece-wise constant entities in the place of a continuous domain. In two and three dimensional problems there is another possibility for error, closely linked to the discretisation error of the numerical model, and often incorporated within the generic grouping of grid errors. The misalignment

of cells and faces with respect to the fluid flow, and the removal of orthogonality within the mesh can be shown to cause a loss of accuracy, and in many cases a loss of stability, within the solution. For this work such geometrical deviations from orthogonal and equally spaced control volumes has been termed the *distortion* error. The effect of such geometrical properties as skew, aspect ratio and deviation from planarity of faces are known to affect the solution [44], but little quantifiable data has been presented. Most presented data of error distribution through a numerical domain quantify the total error, and some provide a direction to this error [23], but no direct linking of a specific geometrical property to the error has been found.

The work in this Chapter is seen as the first step towards a fully automated grid generation procedure. The local numerical error of a solution has been assessed with respect to the local geometrical dimensions in terms of both discretisation and distortion error. Underlying discretisation error can be assessed and removed by a grid dependency study, as suggested by Eça and Hoekstra [79], leaving the distorting effect of the various geometrical properties. Polynomial curve fitting or least squares error fitting can then be applied to link generic geometric trends to the error estimation. This method has been applied to the flow solver described in Chapters 3 and 4.

While overall trends will most likely adhere for other codes this work primarily presents a method of assessing grid error, particularly for the *a priori* unknown flow solution analysis.

5.2 Error Analysis of Numerical Solution

In a one dimensional problem there are three distinct levels of error between the physical reality and the numerical solution; the omission of physical parameters; the error incurred in the representation of the physical world by a mathematical equation; and the error of representing a continuous domain by a discrete numerical mesh [80], as shown in Figure 1.1. The order of accuracy of any scheme, both temporally and spatially, can be defined from truncation error analysis, and the stability via Von Neumann analysis or a similar method. This provides a discretisation error in terms of an order of Δx and Δt. Figure 5.1 graph-

ically illustrates this break down of the error into modelling assumptions and mis-representations, discretisation error and the distortion error. As the grid becomes coarser the discretisation error increases and so also can the distortion error. It must be noted that as with coarser meshes the discretisation error *will* increase, but the distortion error may not, depending as it does upon properties that are non-dimensional, and hence not dependent upon cellular size. Yamaleev [81] comments that "the concentration of grid points in regions which most influence the accuracy of the solution may at the same time introduce additional errors due to the grid non-uniformity". The current work deals only with a first order Euler code, but this should work as an advantage, highlighting the effects of grid distortion that can cause problems in higher order viscous solvers.

Figure 5.1: Division of error for mesh size variation

The dependence of a solution upon the discrete mesh has been further observed and quantified, via forms of the Richardson extrapolation, as well as general discussion of the definitions of error and verification, by Stern et al [82] and Eça and Hoekstra [70].As noted in Section 5.1, as well as this error due to the use of discrete entities, the lack of orthogonality and planarity within the mesh inserts another source of error. The majority of research concerning the topic of grid quality has tended to be of a qualitative nature, rather than a quantitative nature, and in total the topic appears to have been given little theoretical

definition.

One of the earliest papers found that deals directly with geometrical grid quality is that by Haimes, Connell and Vermeersch [21]. The emphasis of the paper is the accurate and easy visualisation of overall grid quality, but also highlights the requirement that "the user have the ability to construct a grid quality measure that corresponds to the accuracy and stability of the flow solver's spatial and temporal discretisation". An unstructured tetrahedral grid is analysed, with cell aspect ratio and skew being used to measure the grids quality, and grid adaptation via flow entropy averaging is considered. There are no direct links drawn between specific grid quality levels and solution accuracy.

Ilinca et al [83], Oliger & Zhu [84] and Rassineux [85] all incorporate some form of numerical calculation of geometrical error to define the mesh quality in the presence of adaptive meshing. Ilinca et al introduce the explicit adherence to the Global Conservation Laws as being of vital importance, and states that the "role of the error estimator is to provide a measure of the grid quality ... and to provide information for mesh adaptation". The concept of the optimal grid for the numerical solution being to equalise the local error for all of the computational cells is also reiterated. Oliger & Zhu [84] developed the concept of linking stability and convergence with error estimation and meshes that are unsuitable for the given flow situation. Rassineux [85] defines mesh quality by control volume shape and size via -

$$Q(tetrahedra) = \frac{\alpha\rho}{h} \tag{5.1}$$

where h is the longest edge length; ρ is the ratio of the radius of the sphere inscribed in the element; and α is a coefficient to set the highest value (for an equilateral element) at 1.

Individual qualities of the mesh are not defined. A size limiter is placed using the average mesh density. A number of examples are detailed, with optimised meshes. The user specifies the lowest allowable quality value (Q) as well as a maximum to minimum edge ratio, and the algorithm defines a suitable mesh.

5.2.1 Error definition

To clarify the approach taken in the present work, the present definitions and understandings of uncertainty, error and verification will be stated. [82]

1. The error, ϵ, is the difference between a simulation or experimental value and the truth.

2. An uncertainty, U, is an estimate of an error such that the interval $\pm U$ contains the true value of ϵ 95% of the time

3. Verification is defined as a process for assessing numerical uncertainty U_{SN} and, when conditions permit, numerical error ϵ_{SN} itself.

If ϕ_o is the *true* solution, and ϕ_i the solution of the i^{th} mesh, then the error for mesh i is -

$$\epsilon_i = \phi_o - \phi_i \tag{5.2}$$

$$= K_G h_i^p \tag{5.3}$$

where p is the apparent order of accuracy, K_G a constant and h a control volume measure. h_i identifies a mesh with representative cell size i where h_0 is taken to be a mesh with zero cell size, and hence provides the ϕ_0 solution.

5.2.2 Error distribution for current scheme

As stated in Chapter 4 the numerical scheme used in this work is an explicit upwinding scheme using artificial compressibility on arbitrary shaped finite volumes. The control volumes are m-faced, node centred polygons, and the faces are n-edged surfaces, defined by triangular segments. Figures 2.12 and 2.13 shows a representation of the faces and control volumes.

By separating the computational algorithm into its individual procedures a discretisation error for the numerical model and a distortional error for the

geometrical entities involved can be determined. Table 5.1 lists the procedures in the algorithm, and the numerical discretisation and geometrical distortion errors. Figure 5.2 details the local frame of reference used for this analysis. From this study the relative influence of the various parameters declaring the geometrical definition can be analysed.

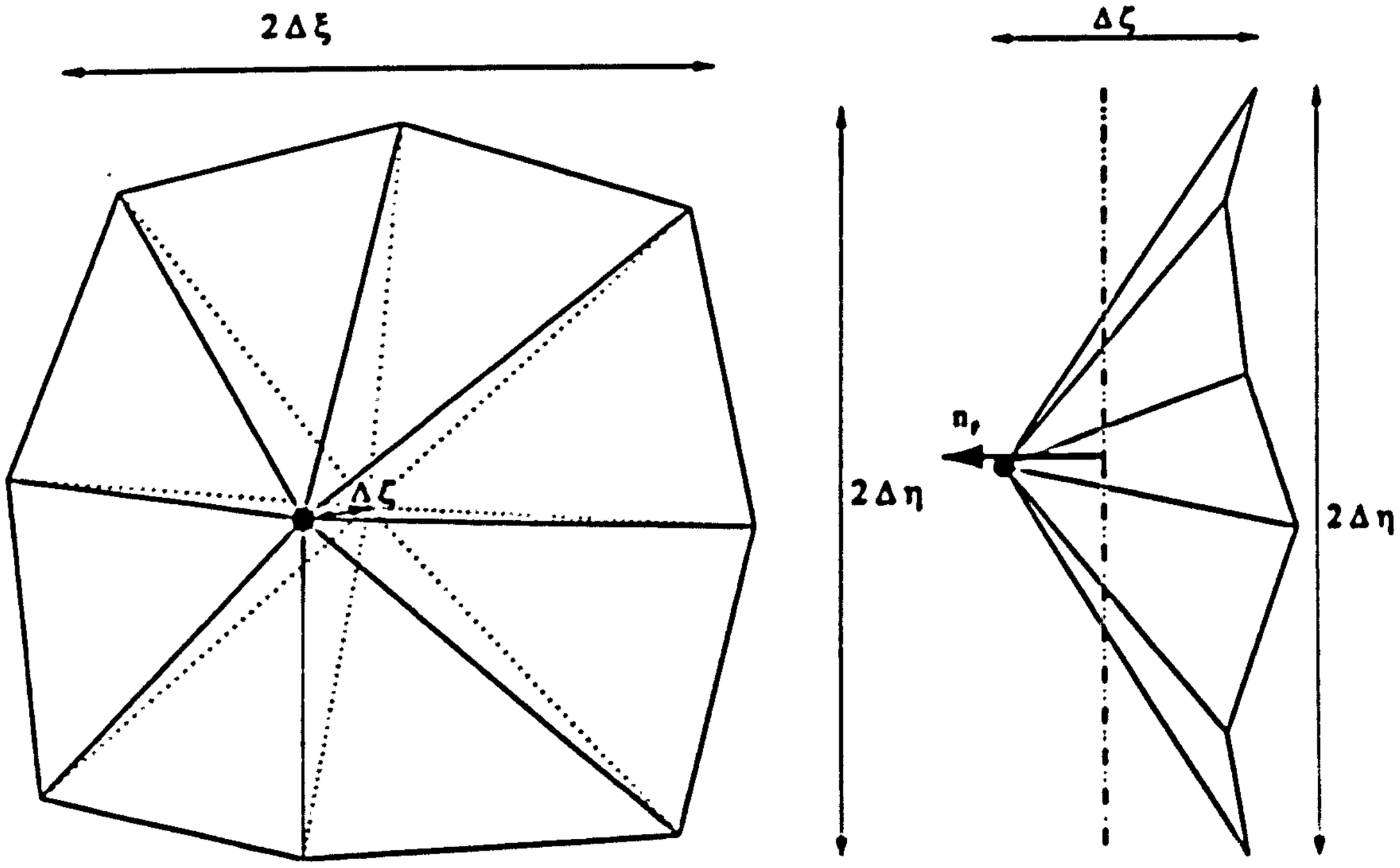


Figure 5.2: Face definition in local frame of reference

Algorithm operation	Mathematical operation	Discretisation error	Distortion error
Domain discretisation	$V_{CV} = \sum V_{tetra}$	-	0
	$A_F = \sum \lvert \vec{n}_F \cdot A_{tri} \rvert$	-	$\epsilon_A \propto \Delta\xi\Delta\zeta + \Delta\eta\Delta\zeta$
	$\vec{n}_F = \sum \sum \frac{A_{tri}}{A_{tri}} \vec{n}_{tri}$	-	$\epsilon_{\vec{n}_F} \propto \frac{\Delta\zeta}{\sqrt{\Delta\xi^2+\Delta\eta^2}}$
Face flux	$\vec{F}_F = \frac{1}{2}\left(\vec{f}_R + \vec{f}_L\right) - \frac{1}{2}\left(\lvert\vec{f}_R\rvert - \lvert\vec{f}_L\rvert\right)$	$\epsilon \propto O(\Delta x)$	$\epsilon_f \propto \epsilon_{\vec{n}_F}, \epsilon_{\lvert\rvert} \propto \epsilon^2_{\vec{n}_F}, \rightarrow \epsilon_{\vec{F}_F} \propto \epsilon^2_{\vec{n}_F}$
CV residual	$R_{CV} = \sum_{j=1}^{m} \vec{F}_j A_j$		$\epsilon_{R_{CV}} = \sum_{j=1}^{m} \epsilon_{\vec{F}_{F_j}} \epsilon_{A_j} \propto \epsilon^2_{\vec{n}_F} \epsilon_A$
Local time step	$\Delta t = \omega\, CFL \frac{1}{S_x\lambda_x + S_y\lambda_y + S_z\lambda_z}$	$\epsilon \propto O(\Delta t^4)$	$\epsilon_{\Delta t} \propto \epsilon_A \epsilon_{\vec{n}_F}$
Sum residual to state	$U^{a+1}_{CV} = U^a_{CV} + \Delta t \sum_{k=1}^{4} \alpha_k R_{CV}$		$\epsilon_{\Delta U^a_{CV}} = \epsilon \Delta t \sum_{k=1}^{4} \alpha_k \epsilon_{R_{CV}} \propto \epsilon^3_{\vec{n}_F} \epsilon^2_A$

Table 5.1: Break down of discretisation and distortion errors

5.3 Properties of Influence

The key problem in assimilating and tabulating the geometrical grid quality data into an easily digested format is its shear quantity. A small, two dimensional numerical grid can have in the order of a few thousand control volumes, and a similar number of interfaces, while research level grids used in the solution of three dimensional problems presently contain from hundreds of thousands of control volumes up to several million. A vast quantity of data can be generated, without any clear conclusions being drawn.

Data visualisation has become a prominent tool in this field, highlighting 'hot' spots of poor quality. Such methods are however still very much using an absolute approach, with no concern of the relative nature of one section of a grid to another. It is now a generally accepted rule that the optimal grid for the computation of a numerical solution is the one for which the local error is the same for all the computational cells [83]. Thus it is not only areas of poor grid quality that are worthy of attention; areas of excessively high quality will be ineffective, and result in flow solver inefficiency. As a result, both areas of low and high geometrical grid quality should be altered for the most efficient solution.

Therefore the required form of measurement must be able to specify a set level of quality and mark extreme values. The obvious choice here is some form of statistical measure, incorporating averaging and deviation from the norm. Such a method will obviously be able to cope with extremely large quantities of data, spotting trends and marking areas for rectification. With the assumption of a standard form of distribution, the bandwidth of the grid quality can be easily set.

The other problem in rating a face or a cell is that each entity has a number of properties, as defined in the following sections, all of which affect the quality to varying extents yet the entity has to be rated by one numerical value.

To understand correctly the effect a given mesh will have upon a certain computational code, the nature of the underlying numerical algorithm must also be known. In partial differential equations there are obviously two or more forms of discretisation; the norm in engineering problems, both structural and fluid dynamics, is spatial and time differentiation. The solution to a system of quasi-linear

partial differential equations of the first order, if hyperbolic, will be associated with propagating waves and properties of the physical system will be dominated by wave-like phenomena.

5.3.1 Facial properties

A face can be specified to be an enclosed surface, in three dimensional space, defined by an arbitrary number of edges, which each are in turn the shortest vectors between two unique nodes.

Due to the arbitrary number of edges, and their arbitrary direction, they are not necessarily of the same length, nor do they necessarily lie within the same plane, as illustrated by Figure 5.2.

The only stipulations required by the GCL are that the edges fully enclose the surface, and that the edges do not intersect except at their limit (that is they do not twist over each other).

Therefore there can be deviations in the planarity and the orthogonality of the edges, as well as a variation of surface area and aspect ratio. A fifth variable is the misalignment of the face normal with the vectored direction between control volume data points. All five of these properties have a bearing upon the numerical solution gained from the numerical mesh.

Face area is the integration of the faces width along its length. Decreasing face area has the obvious effect of increasing accuracy; as noted in Chapter 3.3, there will always be a step wise linearity to the solution, but decreasing face area will nominally result in smaller control volumes and smaller 'steps' between adjacent volumes. Smaller face areas also result in smaller interface fluxes - of primary interest to Finite Volume Method (FVM) calculations. These smaller fluxes cause smaller residuals between time steps, and therefore smoother change to the domain flow.

Face aspect ratio is the ratio of one face dimension to another in an orthogonal framework. The aspect ratio of the face is important to consider, particularly when concentrating control volumes around a body of interest or areas of high

gradients. The focusing of discretisation in one body fitted dimension only produces very elongated cells - so while the face areas have been reduced, the aspect ratio of approximately half the faces will have increased sharply. This is a major problem in node centred meshes for those control volumes adjacent to the boundaries. High aspect ratio faces (and therefore volumes) are of use in viscous flow calculations in areas where large shear forces are present, but in non-viscous applications they can cause localised 'chequer-board' effects and increased errors, particularly in lower order schemes. This is normally due to the timestep being mis-calculated, causing instabilities due to CFL limit contravention.

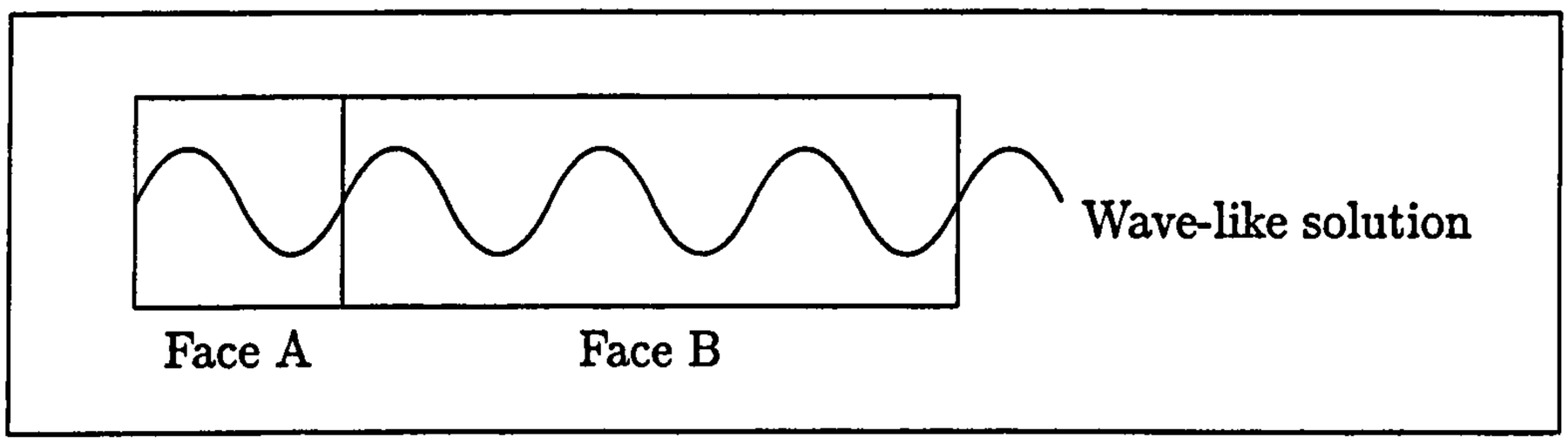


Figure 5.3: Wave-like solution passing across interfaces

Figure 5.3 diagrammatically shows a wave-like solution passing through two faces, one with an aspect ratio of approximately 1.0 (face A) and one with a much higher aspect ratio (face B).

While only one wave form (approximately) can be present in the cell projected onto face A, several can be present in the cell projected onto face B. Because the timestep is commonly calculated from a cell's volume, with an averaging of the cells facial sizing, the extreme dimensioning of face B would probably not be recognised. As a result the stability criterion for the numerical algorithm would be broken and local instabilities would be incurred, resulting to the observer as a chequer-board effect of alternatively 'hot' and 'cold' patches in the flow. Another effect of a long face (and cell) would obviously be a much higher gradient across the face, which can lead to a stretching of the underlying finite differencing scheme beyond its limits, thus lowering accuracy.

Face deviation from planarity is relevant only in non-triangular faces. A plane can be uniquely defined by two vectors, which, if originating from a unique point in space, also uniquely define a triangle (the third side joining the free ends of the vectors). A surface with an arbitrary number of sides can be

split into a set number of triangles, n, using a regular stencil of nodes defining the face . As a result n planes can be defined, not necessarily identical. An averaged value of the orientation of these planes can be utilised to define the plane of the surface, and the variation of the triangular sub-element normals' used to define the deviation. Deviation from planarity usually causes a loss of accuracy, and in extreme cases will result in serious problems such as sources and sinks within the physical domain of the model. This is due to the fact that a non planar face will not have the same area and normal direction as one that is planar; area calculations are required to be fully developed surface definitions for domain volume calculations, yet projected planar for flux calculations. It is difficult to accurately calculate both during the flow solution process. What is a greater problem is the definition of the face normal. A planar face will have a constant normal direction across its entirety, but a non-planar surface will have a varying normal direction. The problem is in the choosing of a representative normal direction for the face as an entity. This will cause geometrical errors and hence a loss of accuracy.

Face skew is the deviation of the surface from a symmetrical shape. The easiest example is that of a quadrilateral face; due to the arbitrary nature of the edge vectors, adjoining edges are not necessarily orthogonal to each other (whatever the plane). A more generalised approach to this geometrical problem, that is applicable to an arbitrarily edged face, uses the edges' normal and the face centroid. If the normal (or negative of the normal, whichever points towards the face centroid) is taken from the edge centre, the skew is the scalar distance by which the projected normal misses the face centroid. In the case of an orthogonal quadrilateral the distance would be zero, as would be the case for an equilateral triangle, and indeed any symmetrical face. The variation in this distance can be used as a measure of asymmetry of the face. The skew of a face destabilises a numerical scheme by producing an asymmetry to the face. This results in a variation in the distance travelled by an entity crossing the face, when travelling in a straight line. That is, the distance a wave-like form would spend tangentially crossing the face would vary. The variation in distance can be specified in wavelengths; if the number of wavelengths incurred is greater than the CFL number then the scheme will become unstable. Therefore a highly skewed face will produce more extreme distances, which are unlikely to be registered in the time-step calculation, and hence more likely to cause instabilities.

Face misalignment of normal is the misalignment of the face normal with the vector between the nodes either side of the face. This property is taken as being the dot product of the interface normal with the vector between the control volume centroid's on either side of the face. Thus an ideal value would be unity and as the value drops from this towards zero so also does the alignment of the face with the edge. Misalignment of the face normal will cause inaccuracies in the flow solution due to the face area not equalling the projected face area that the flow from data point to data point 'observes' (the dot product of the velocity and the face normal).

5.3.2 Cellular properties

A cell, or a control volume (CV), is a three dimensional body enclosed by an arbitrary number of faces, as detailed in Figure 2.13.

The Global Conservation Law (Equation 2.1), in particular the SCL, must be adhered to. Each and every face defining a side of the volume will have the properties defined in the last section. Another key point worth noting is that the faces are directional. Because the face has vector defined edges it has a uniquely defined plane of existence, which has a unique normal vector. This face normal defines a 'positive' and 'negative' side to the face.

Cell volume is the integration of the cross sectional area along the length of the three dimensional entity. The largest single influence upon solution accuracy, from a geometric point of view, is the volume of the control volumes. The coarser the mesh, the lower the accuracy. Indeed, entire flow features can be omitted if the mesh is not fine enough in the required area. It should also be stressed at this point that a fine mesh throughout the domain is generally a very inefficient method of gaining the desired accuracy. The key issue is the relative size of the flow gradients and the cellular volume size. Stability can also be affected via the extrapolation of properties between CV's beyond that suitable for the method utilised.

Ratio of adjacent cell volumes is the arithmetical ratio of the enclosed volumes of adjacent control entities. The ratio of adjacent volumes can alter the final flow solution due to the common use of interpolation and extrapola-

tion routines. When adjacent volumes are widely different checkerboard errors may form from either over- or under-prediction of properties. This is also closely linked to the type of numerical solver applied. Central differencing is unstable in most cases, so numerical diffusion is used, as it is also for higher order upwinding schemes (there termed a limiter); second and fourth order terms are common, hence implicitly involving adjacent volumes. As with the actual volume specification, the key aspect is the ratio of flow property gradients. Ideally these values should be as close to unity as possible through out the domain.

Cell twist is akin to the skew of a face. For a hexahedral cell it is the variation of the cells' edge's orthogonality from a localised framework, while more generally it can be thought of as the scalar distance by which the projected normal of a face (from its centroid) misses the cells volumetric centroid. Control volume twist tends to de-stabilise a system by allowing misrepresentations of cellular geometry to occur in the time step calculation. Another danger of highly twisted control volumes is miscalculation of interface fluxes due to unrecognised misalignment to the flow; the face normal (a weighted average of its planar components) may be orthogonally aligned to the flow, but its extremities will not be, causing numerical errors.

Solid angles are the segments of a sphere subtended by the joining of two or more faces of the volume. Thus the joining of three mutually orthogonal faces will create a solid angle of $\frac{1}{6}\pi r^3$, and if a unitary radius is assumed then the value is $\frac{\pi}{6}$. The final geometrical property of a discrete volume to significantly affect the flow solution, solid angles, tend to influence accuracy and numerical error. This is due to flux calculation error normally. Very acute and/or obtuse angles result in high aspect ratio segments of control volume between the face and the control volume storage point. This in turn produces an imbalance to the numerical differencing scheme. In addition the very high aspect ratio nature of the control volume segment, as well as the highly acute/obtuse nature of the angle itself, tend to approach the limits of floating point precision, hence introducing a greater numerical error.

5.3.3 Flow solution properties

The primitive properties of a fluid will vary throughout the domain of interest, in varying quantities and in a unique manner. There is some cross coupling (for example as velocity increases, pressure usually decreases), but in the main the properties are independent of each other, or non-linearly coupled.

The **gradient** of any property in a given direction is merely the rate of change of the property with respect to distance, and can be uniquely defined for any point within the domain. In areas of very high flow property change the main effect will be a loss of accuracy due to small scale variations in the flow. As a result of these high gradients another problem - particularly in higher order solutions - is 'overshooting', where the high gradient causes ripple-like oscillations. This clearly leads to localised instabilities.

Lower than average gradients on the otherhand do not cause any detrimental effects to either the accuracy or the stability of the solution, but are however a major waste of resources (CPU time and memory requirements in particular). These resources could be much more advantageously utilised elsewhere in the solution, such as the volumes of higher than average gradients.

This general principle of minimising the range of gradients applies to *all* the primitive variables.

The **divergence** is the extent to which the vector field is expanding or compressing around a point. For relatively low speed flows (i.e. incompressible flows), the continuity equation reduces to the requirement for zero divergence. Physically the divergence of a flow is the time rate of change of the volume of a moving fluid element, per unit volume [18]. In incompressible flows the presence of flow divergence is a sign of a major instability in the solution, but is also a serious problem in compressible flows. If divergence occurs then the mass fluxes through the mesh cell-boundaries will not cancel and hence the numerical scheme will not keep the total mass constant [9]. This type of problem can often appear at singularities within the mesh, and at the conjunction of interfaces of vastly different size.

The *circulation* is defined as the line integral of the velocity around any

closed curve. Thus the circulation is given by -

$$-\Gamma = \oint_C \vec{V} \cdot \vec{dr} \tag{5.4}$$

By Stokes theorem it is clear that if the circulation of a flow is zero, then it has no rotationality. **Rotationality** (or vorticity) is the extent to which the flow is rotating about a given point [18, 86]. The ideal situation is to have all flow movement either normal or tangential to the interfaces, thus reducing the numerical error due to rounding errors. Therefore a highly rotational flow would be more suited to using a cylindrical or spherical mesh (depending on the exact nature of the flow) rather than a cartesian based mesh. That is, a 'O' type mesh would have much greater accuracy than an 'H' mesh in volumes of large vorticity. Thus the variation of rotationality, when taken in conjunction with the formation of the mesh present, will have a marked effect upon the conservation of properties during the run-time of the solver as well as the accuracy of the final solution.

There are other geometrical and flow solution properties that can be identified and which do have an influence upon the solution accuracy, but for the sake of simplicity and clarity they shall not be discussed here. A review of the cellular properties shows a relationship between these properties and the facial properties; for example a control volume with large area faces will probably have a large volume. Similarly highly skewed faces lead to twisted cells. Literature upon the subject is of the opinion that deviation from planarity and orthogonality are the two key topics of interest, as well as mesh regularity [19, Chp 32-34].Considering this, the fact that the flow solver operates upon a face based algorithm, and the error analysis of Table 5.1 it has been decided to consider only the following properties-

- Face aspect ratio
- Face skew
- Face deviation from planarity
- Face misalignment of normal
- Flow solution gradients (pressure and velocity)

The flow solution gradients will be assessed with respect to the discretisation error and the four geometrical properties with respect to the distortion error.

5.4 Numerical study of flow solver

5.4.1 Error inducement

Discretisation errors, and a number of distortion errors are present in any grid by definition, but to analyse them in depth a range of values have to be obtained for the same problem. A range of discretisation errors for both case studies presented in Section 4.3 have already been created by the use of a range of mesh densities. This was deemed a suitable range and hence no extension of the range was modelled; instead the present range was studied in depth.

Distortion errors however have not been varied in the modelling to this point and hence require inducement and study. Three forms of distortion error have been devised that shall be applied to the 'base' meshes used in Section 4.3 and for the discretisation error analysis. These three types of movement are shown in Figure 5.4 and can be defined as:

Planar movement of the central node on a face The central node on a face is moved in the direction of the face normal, thus increasing deviation from planarity, altering face area and altering cell volume. Other face properties will be altered to a minimal extent.

Misalignment of face normal with node to node vector All the nodes on the face are moved in a planar manner pivoting on the central node of the face. Thus the face normal will change, and deviate further from the node to node vector. All other face properties will have minimal variation, although faces using the same nodes will move the said node in different directions, hence causing a greater range of facial properties than initially desired.

Asymmetrical misalignment of face normal with node to node vector All the nodes except the centre node on the face are moved, in the same direction, in the plane of the face. This produces a 'V' shaped face. As a result the face deviates from planarity as well as having the face skew altered. Facial area and cellular volume will also be affected.

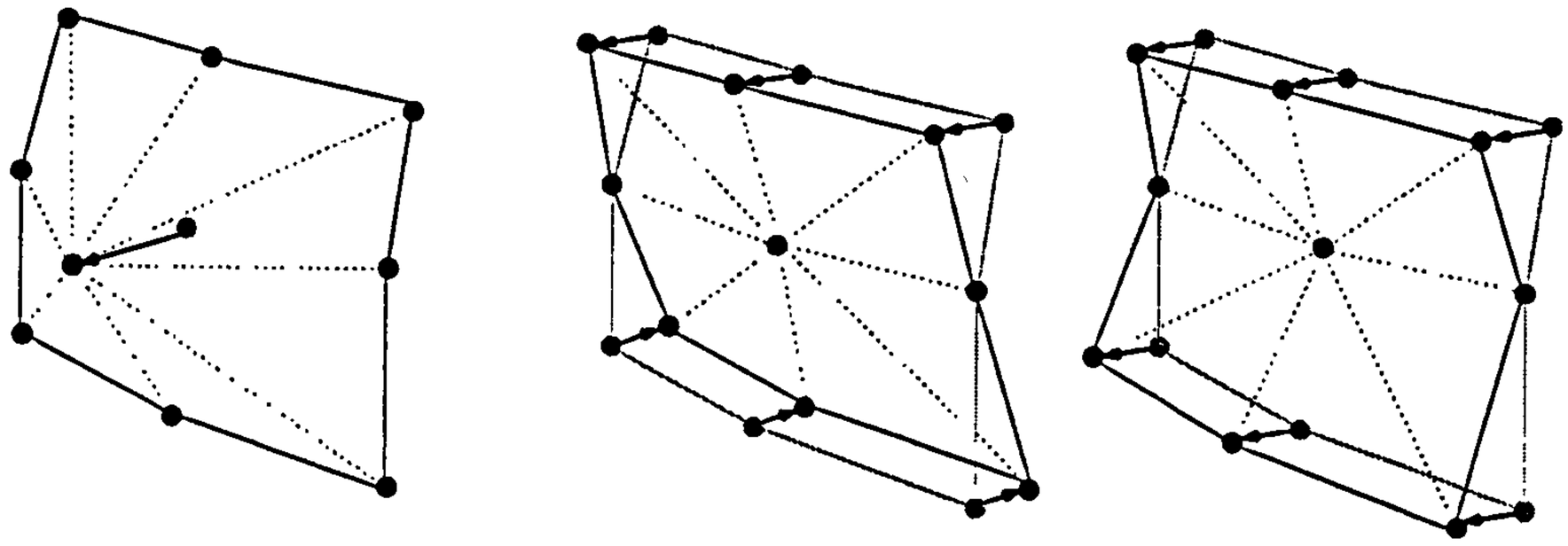


Figure 5.4: Types of facial distortion created to induce distortion errors in solution

5.4.2 Error analysis

The local error for each solution defined was calculated as the error in the pressure value for each node of the mesh when compared to the pressure obtained for the finest mesh at the same point. The finest mesh was the 268×67 mesh for the 10% arc hump channel flow and the 593×33 mesh for the NACA 0012 foil at 4° angle of attack. This was non dimensionalised with respect to the pressure of the finest mesh. That is, the local error was the percentage difference of the pressure solution at the given point, as detailed in Equation 5.5 -

$$\epsilon_{local} = \frac{|P_{\text{fine mesh}} - P_{\text{current mesh}}|}{|P_{\text{fine mesh}}|} \tag{5.5}$$

The average local error of the solution was defined to be the average of the local pressure error throughout the domain.

$$\overline{\epsilon_{local}} = \frac{\sum_{i=1}^{N} \epsilon_{\text{local } i}}{N} \tag{5.6}$$

A global measure of error was deemed to be the force upon a wall boundary of interest (the hump itself and the aerofoil for the two respective case studies). The forces obtained were compared to the validation data, and the error was deemed to be the difference. It is worth re-iterating that these forces are non-dimensional forces, in the streamwise and vertical directions for the hump and lift for the foil.

The flow gradients of a given point were determined to be the average absolute difference of the flow properties at the point and all surrounding control

volumes, as detailed in Equation 5.7. Thus average flow gradients could also be defined for a given mesh by averaging over all control volumes.

$$\text{FLOW GRADIENT}_i = \frac{\sum_{j}^{\text{N adjoining CV}} |P_i - P_j|}{\text{N adjoining CV}} \tag{5.7}$$

Facial properties were calculated for all faces of a mesh and, where required (aspect ratio etc), offset or inverted to give a range from zero for no distortion to increasing positive values as the quality drops. The average and standard deviation for the mesh were determined from these values. Standard deviations were calculated to provide an overall guide to the nature of the data distribution, while averages were used to highlight trends between sets of results.

$$\text{AVERAGE} = \bar{x} = \frac{\sum^{N} x}{N} \tag{5.8}$$

$$\text{STANDARD DEVIATION} = \sigma = \sqrt{\frac{\sum^{N}(x^2) - \frac{1}{N}(\sum^{N} x)^2}{N-1}} \tag{5.9}$$

In order to determine what relationship (if any) there is between a given parameter and the flow solution accuracy two methods of analysis were used. The average local error for a given solution was compared to the relevant solution flow gradients and mesh geometrical properties on a global basis graphically. This highlighted overall trends, and determined what order of proportionality (i.e. linear, squared etc.) was present.

The second method was to perform a least squares fit between the local error and the local geometrical distortion. This second method of analysis creates a template of the relative dependence of the solution upon the chosen parameters. The additional reason for the use of the least squares fit is that because any given solution will have a combination of sources of error a global analysis of the properties in isolation will not provide the true relationship. The method of least squares fitting is detailed in Appendix C.

In an attempt to separate the error due to geometrical parameters from the discretisation error, the 'base' solution local error for each mesh density was removed from the solution local error of the geometrically distorted mesh leaving

the local error due to distortion. Where average local errors for such meshes are listed it is the average of this difference in local errors.

$$\epsilon_{MESH} = \epsilon_{DISCRETISATION} + \epsilon_{DISTORTION} \tag{5.10}$$

5.5 Validation of quality assessment

5.5.1 Flow over 10% hump

For discretisation error analysis all eight meshes were analysed, and for distortion analysis the 18×5, 58×15 and 118×30 meshes were analysed. Each of these three meshes were altered in the three manners prescribed in Section 5.4 with two levels of movement - 30% and 45% of the distance between the cell centroids - for the coarse and medium mesh and one level of movement (30%) for the densest mesh. This gave in total fifteen altered meshes.

The variation in the total non dimensional force upon the hump for the eight meshes utilised in Section 4.3 is plotted against the relative mesh density (H_1/H_i) in Figure 5.5. The result from the panel code is marked, to provide an indication of the absolute error in force present. The averaged local error is presented in Figure 5.6, plotted against the mesh density. With the inclusion of the 15 distorted meshes the variation of force in the streamwise and vertical directions upon the hump, again to a base of mesh density, is presented in Figures 5.7 and 5.8. The panel code solution value is indicated in the same fashion as Figure 5.5. As an indication of the error distribution within the domain, Figures 5.9 and 5.10 show the error in the pressure definition for the 58×15 mesh and for the same mesh with the face normals misaligned with the node to node vector by 45%.

Plots of the variation of flow property gradients against the average local solution error are presented in Figure 5.11. These plots contain only results for the meshes analysed in Section 4.3, and are deemed to have insignificant distortion error. That is, any error present is discretisation error alone.

Plots of the variation of geometrical properties with average local solution error are presented in Figures 5.12-5.15. As stated in Section 5.4 the error presented in these results is the average local distortion error. The discretisation error, that presented in Figure 5.9, has been subtracted from each point and the resultant error distribution averaged to produce the values presented. Both the average and the standard deviation of the geometrical properties are presented, denoted by different symbols. The three mesh densities are signified by different colours of symbols. Where possible, trend lines for each of the different mesh densities are also presented.

From these results it was determined that there were predominantly linear relationships present, thus the relationship used within the least squares fit was

$$\begin{aligned} \text{GRID QUALITY} &= \beta_1 \text{skew} + \beta_2 \text{AR} \\ &+ \beta_3 \text{deviation from planarity} + \beta_4 \text{misalignment of } \vec{n} \end{aligned} \tag{5.11}$$

where $\beta_1 - \beta_4$ are constants of proportionality to be determined by the least squares fitting. Therefore a grid quality value of zero would indicate zero distortion and high quality. A large number would indicate a mesh of poor geometrical quality.

In total 5 least squares fitting procedures were accomplished; one to each of the families of geometrical errors. That is, the 18×5 mesh with its solution was fitted with the 30% movement distorted meshes and their respective solutions, as well as being fitted with the 45% movement distorted meshes, and so on for the 58×15 and 118×30 meshes. The influences of the four geometrical properties, by least squares fitting, for all the families of distorted meshes are presented in Figure 5.16.

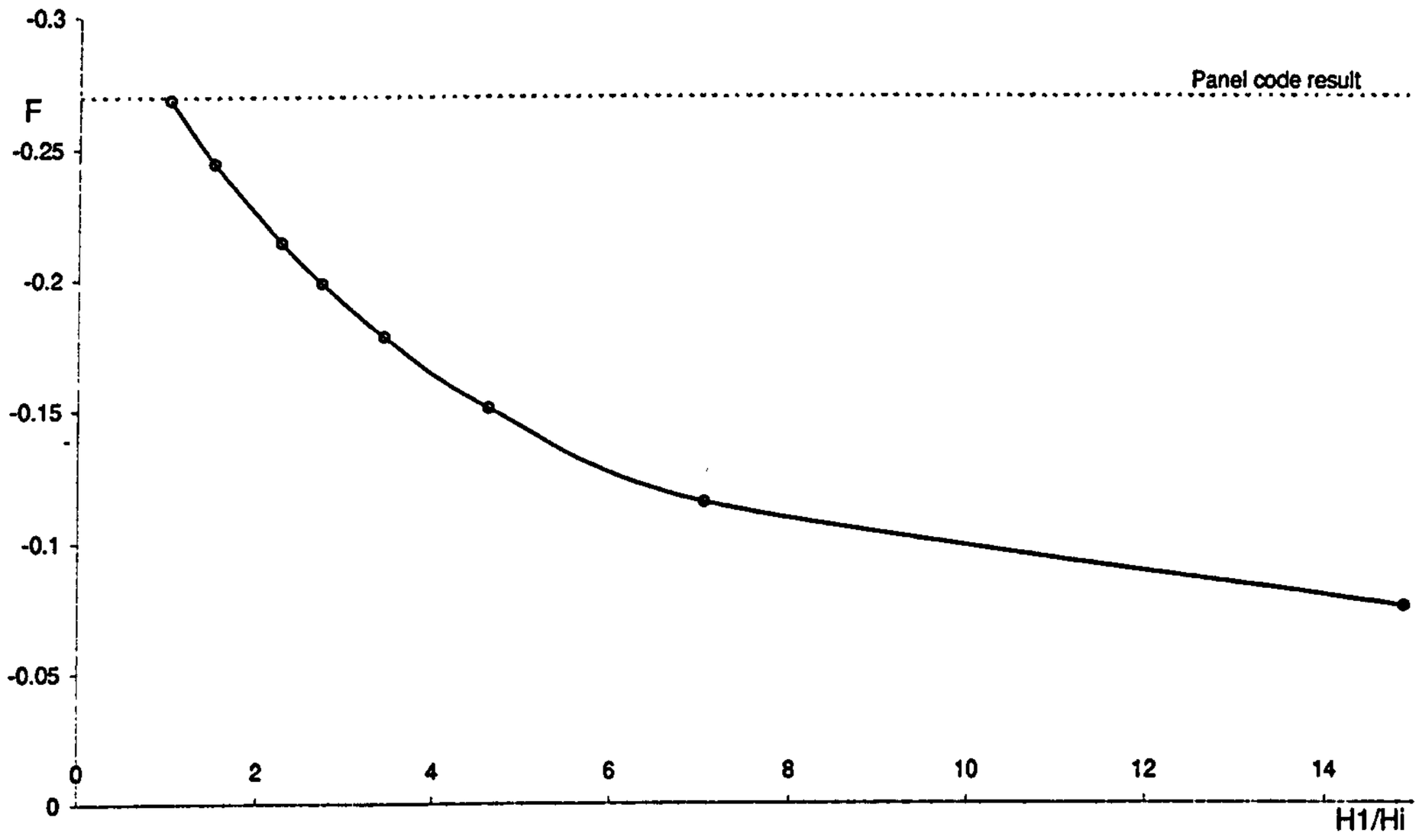


Figure 5.5: Error in force for 10% arc hump with mesh density variation

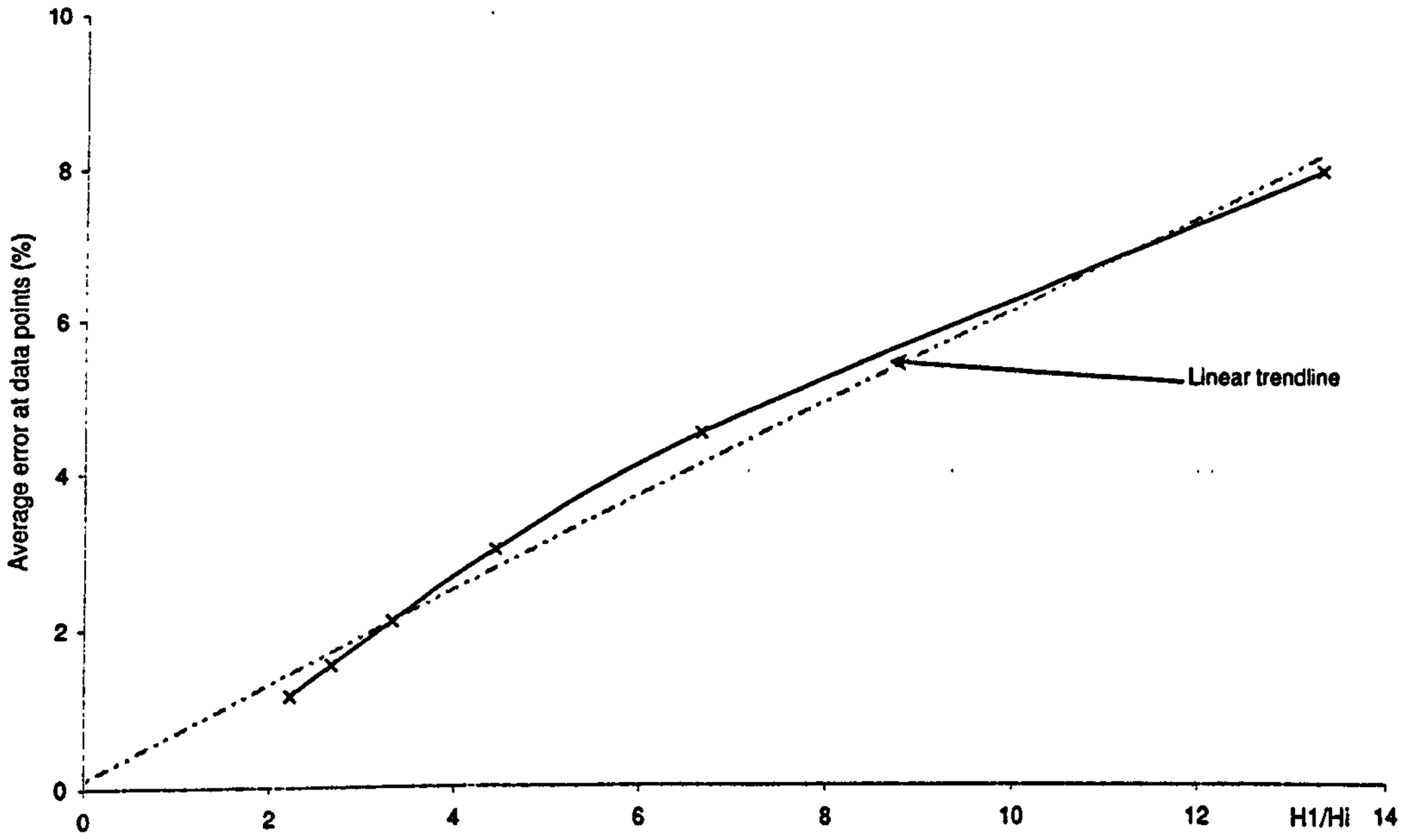


Figure 5.6: Variation in average local error with mesh discretisation for 10% arc hump

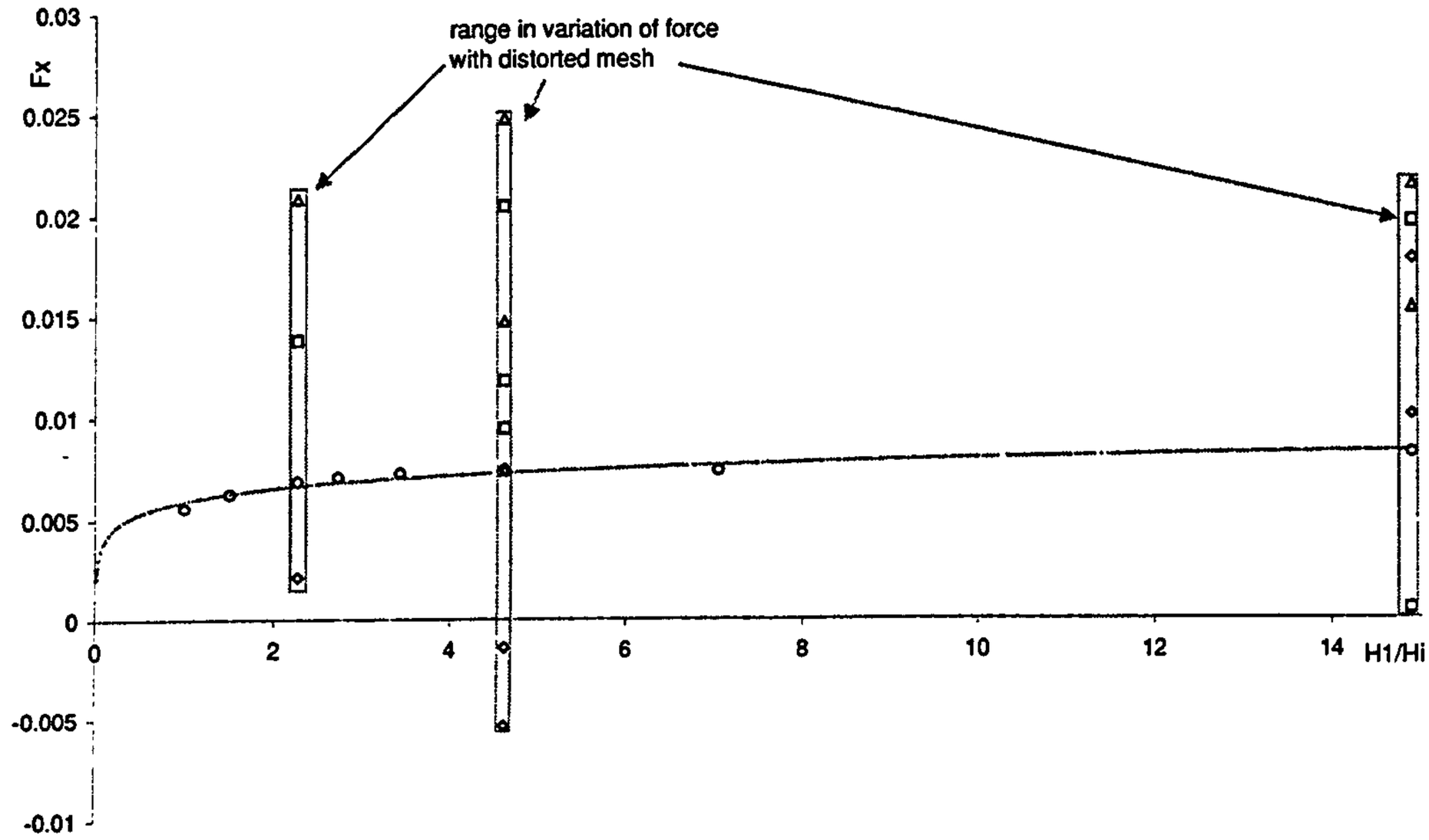


Figure 5.7: Error in X direction force for 10% arc hump with mesh density and quality variation

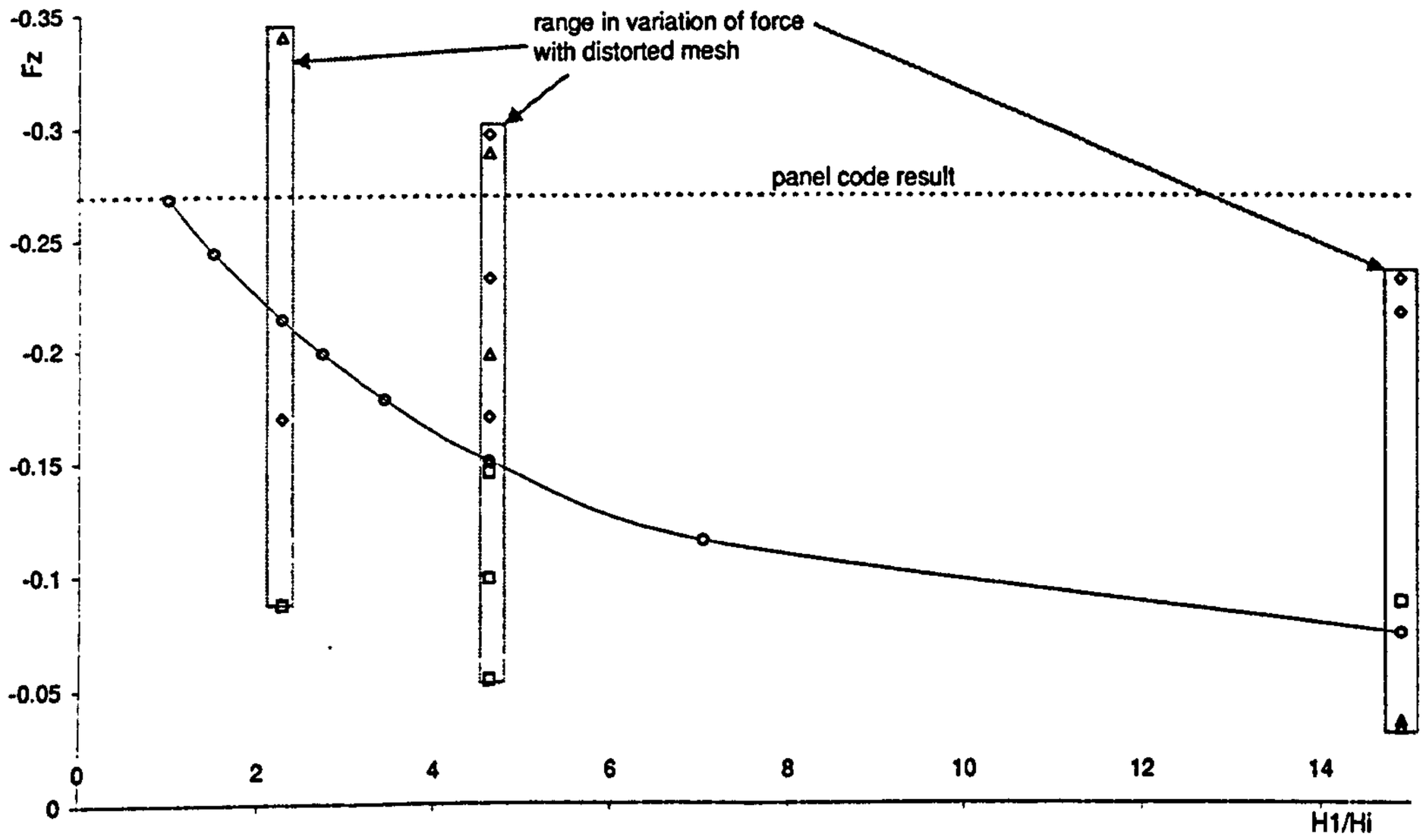


Figure 5.8: Error in Z direction force for 10% arc hump with mesh density and quality variation

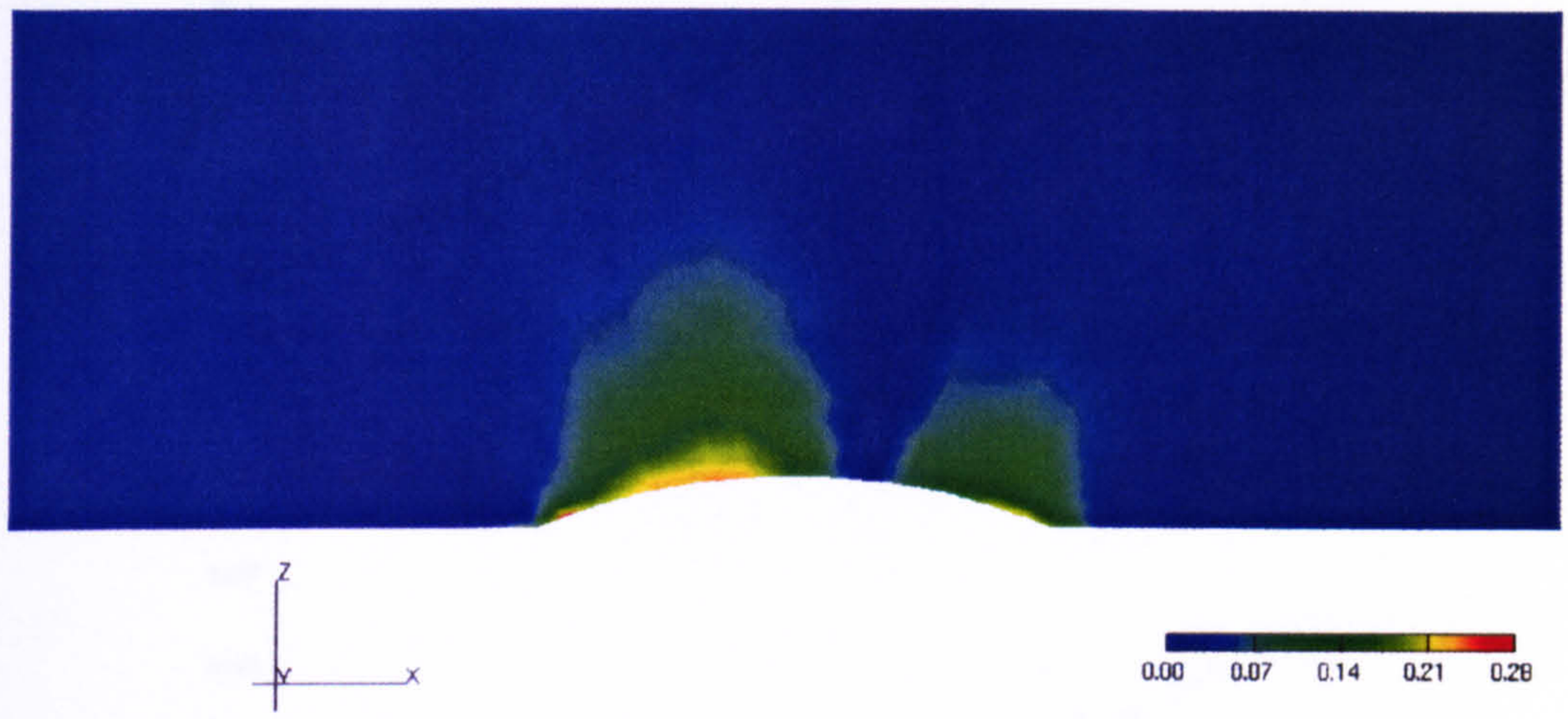


Figure 5.9: Distribution of local error for 58×15 mesh over 10% arc hump

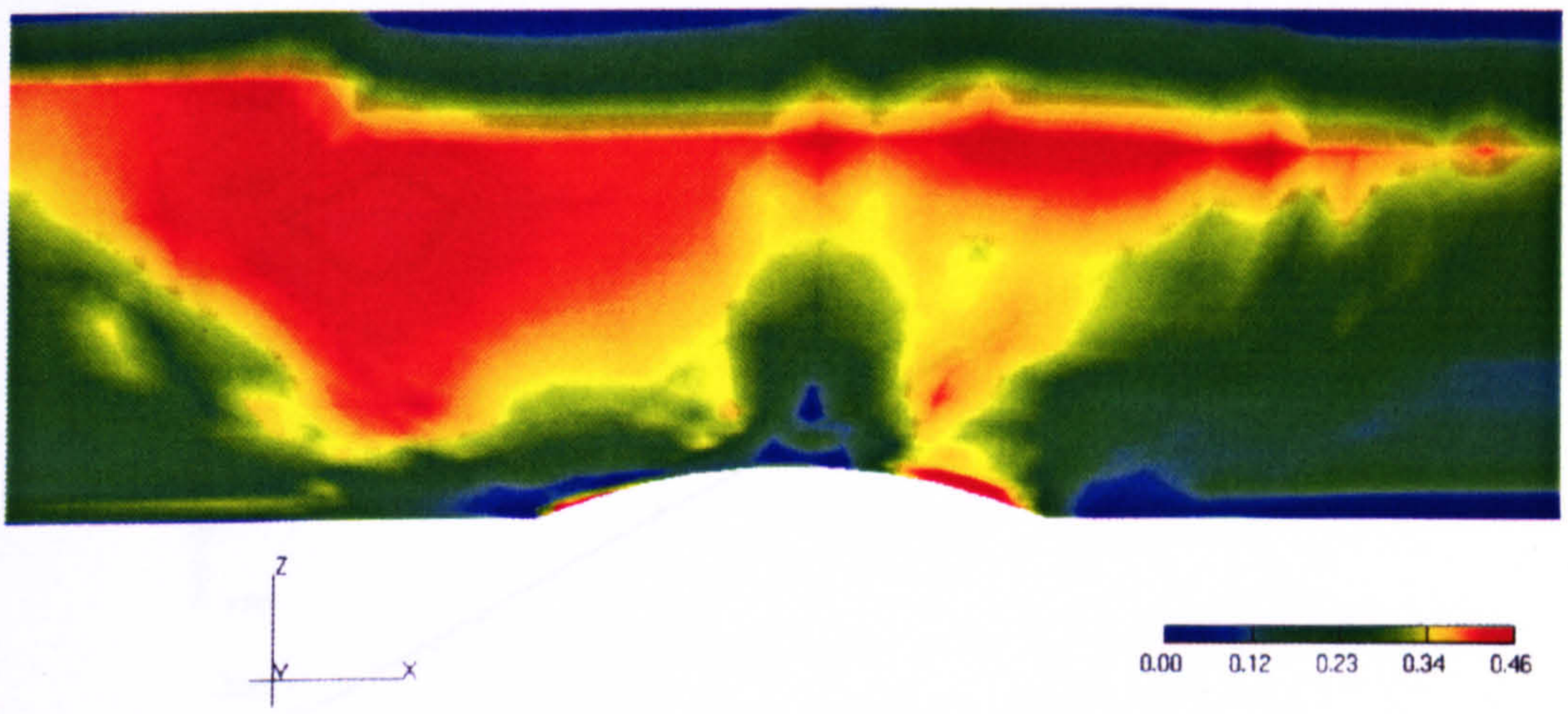
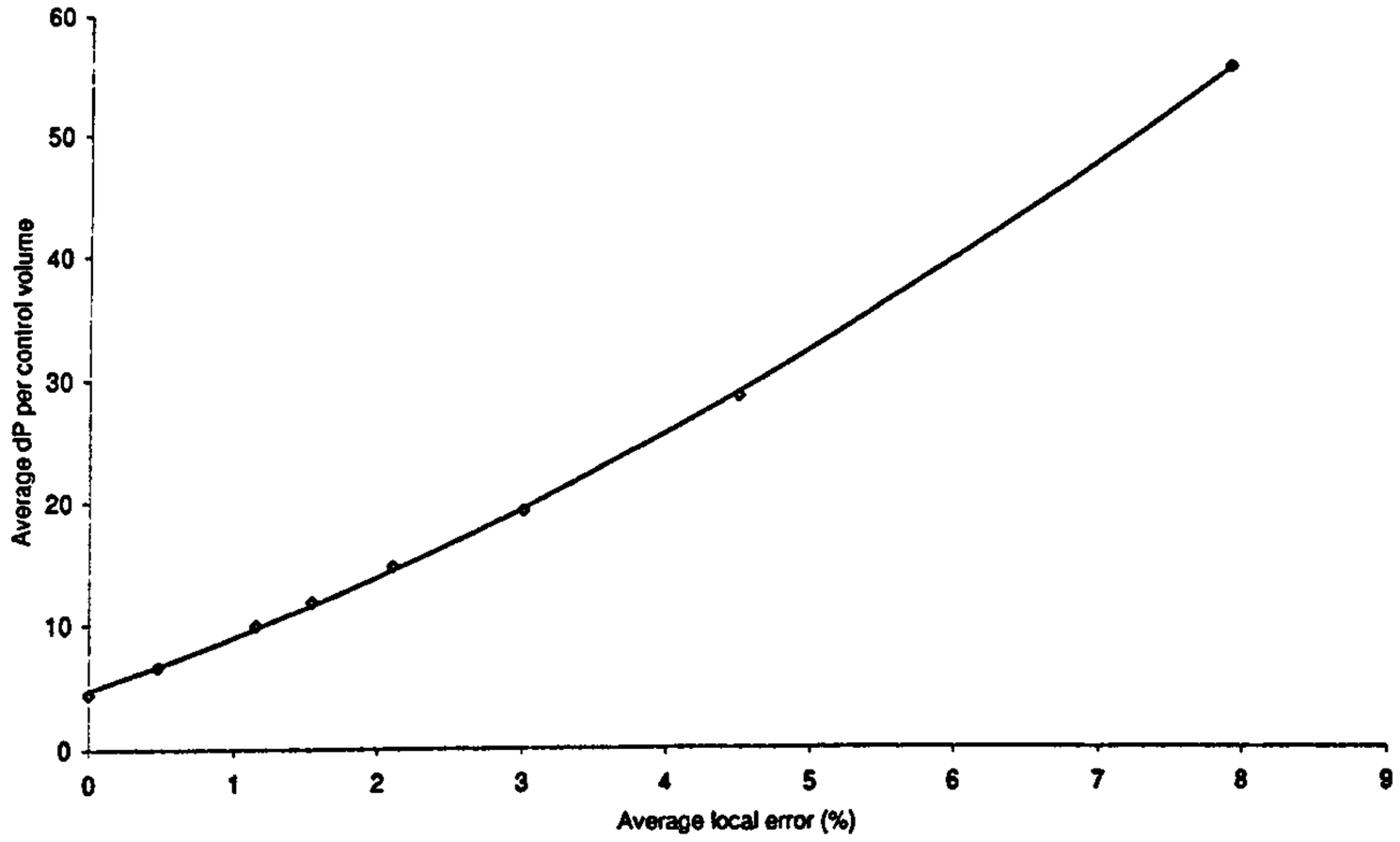
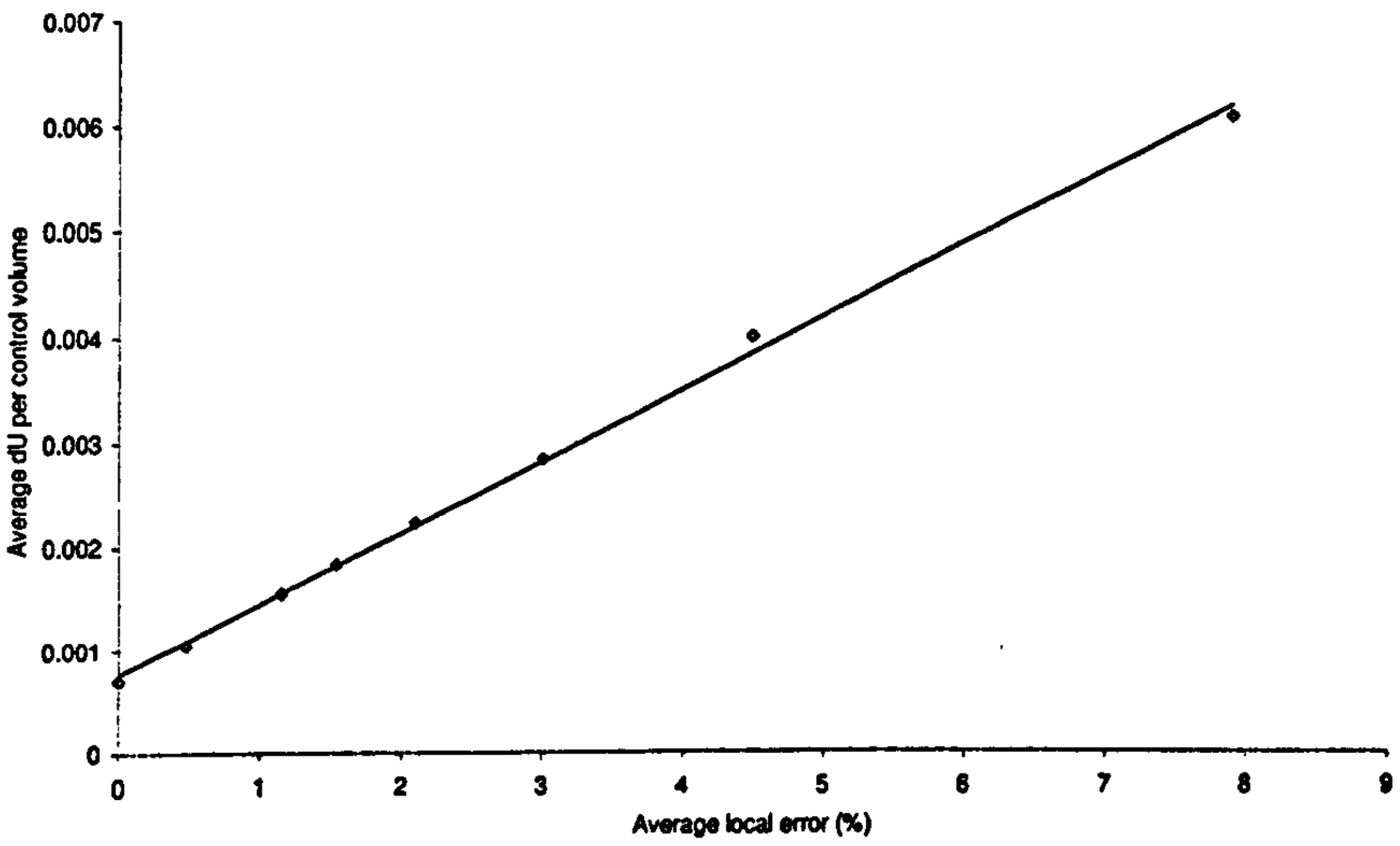


Figure 5.10: Distribution in local error for 58×15 mesh with distortion





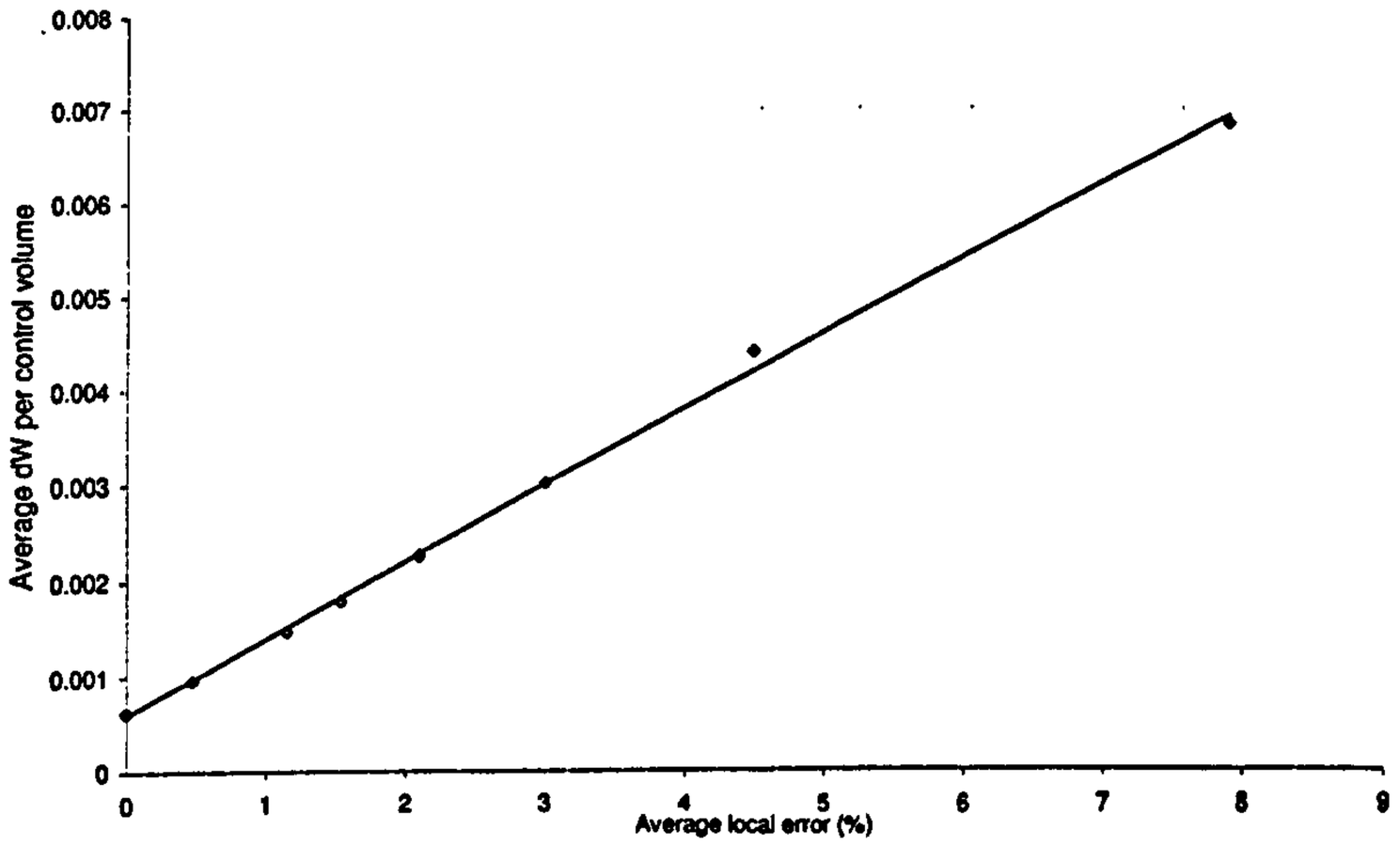


Figure 5.11: Variation in differential of flow properties (P, u, v respectively) over 10% arc hump

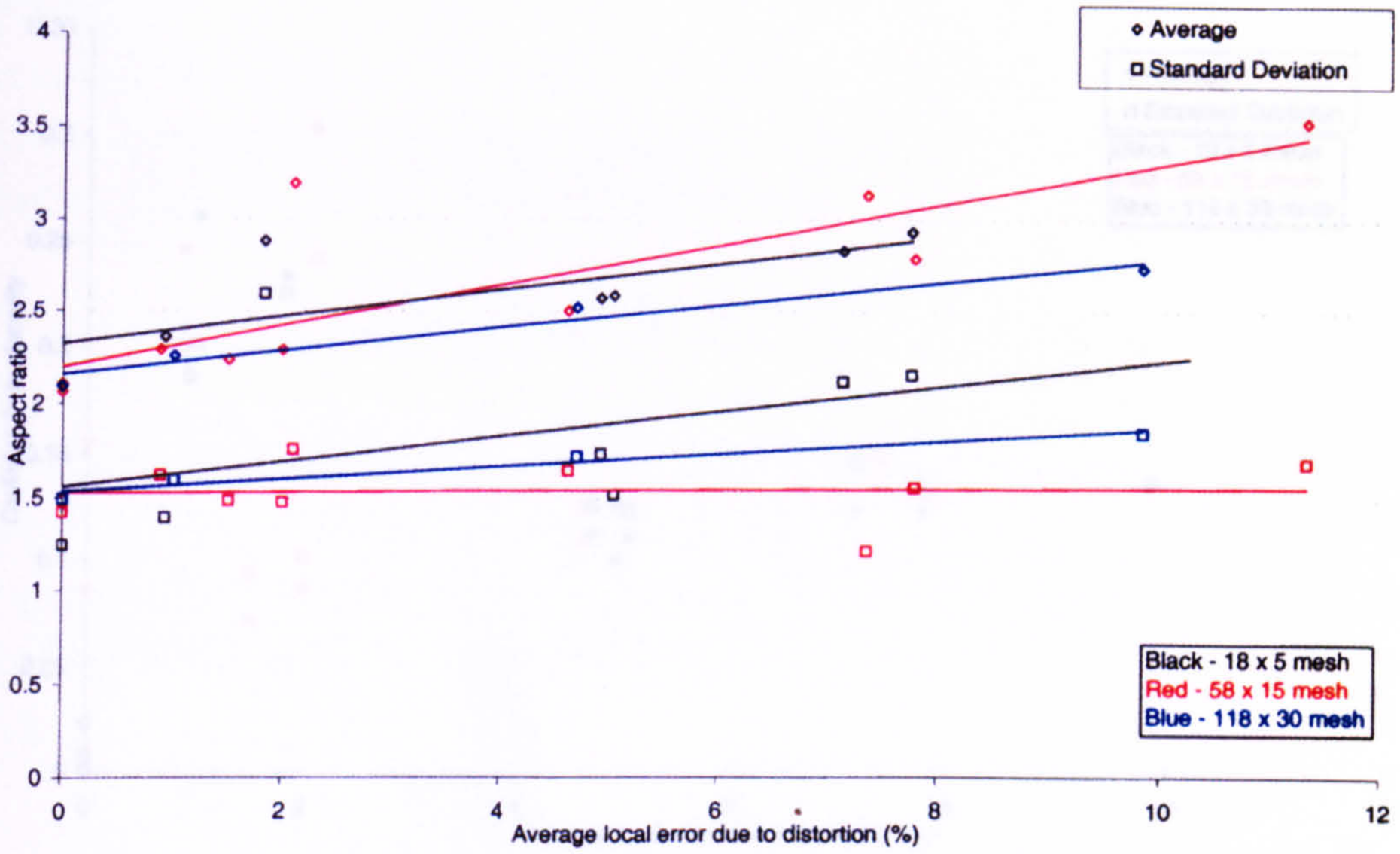


Figure 5.12: Variation in solution error with facial aspect ratio for all distorted meshes

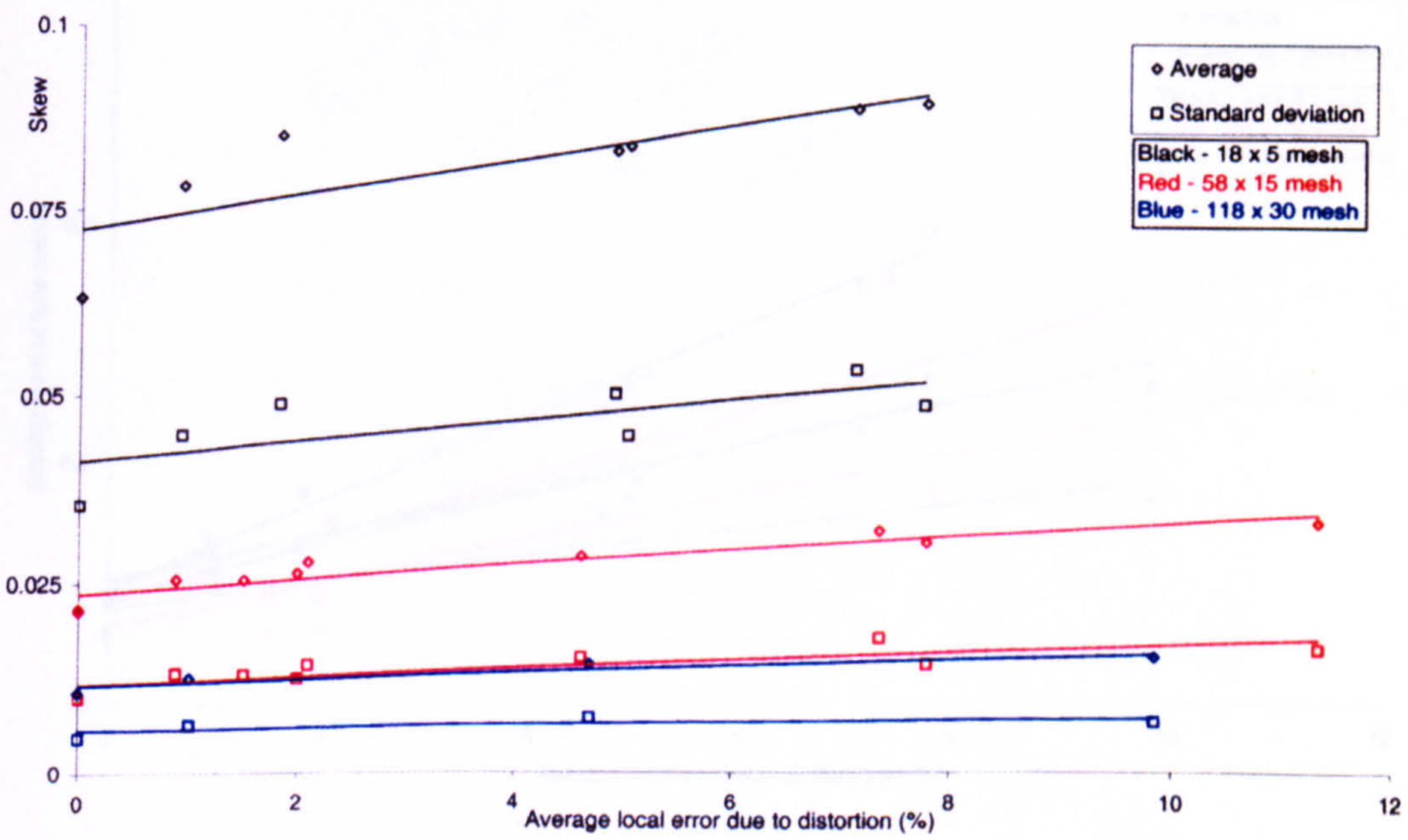


Figure 5.13: Variation in solution error with facial skew for all distorted meshes

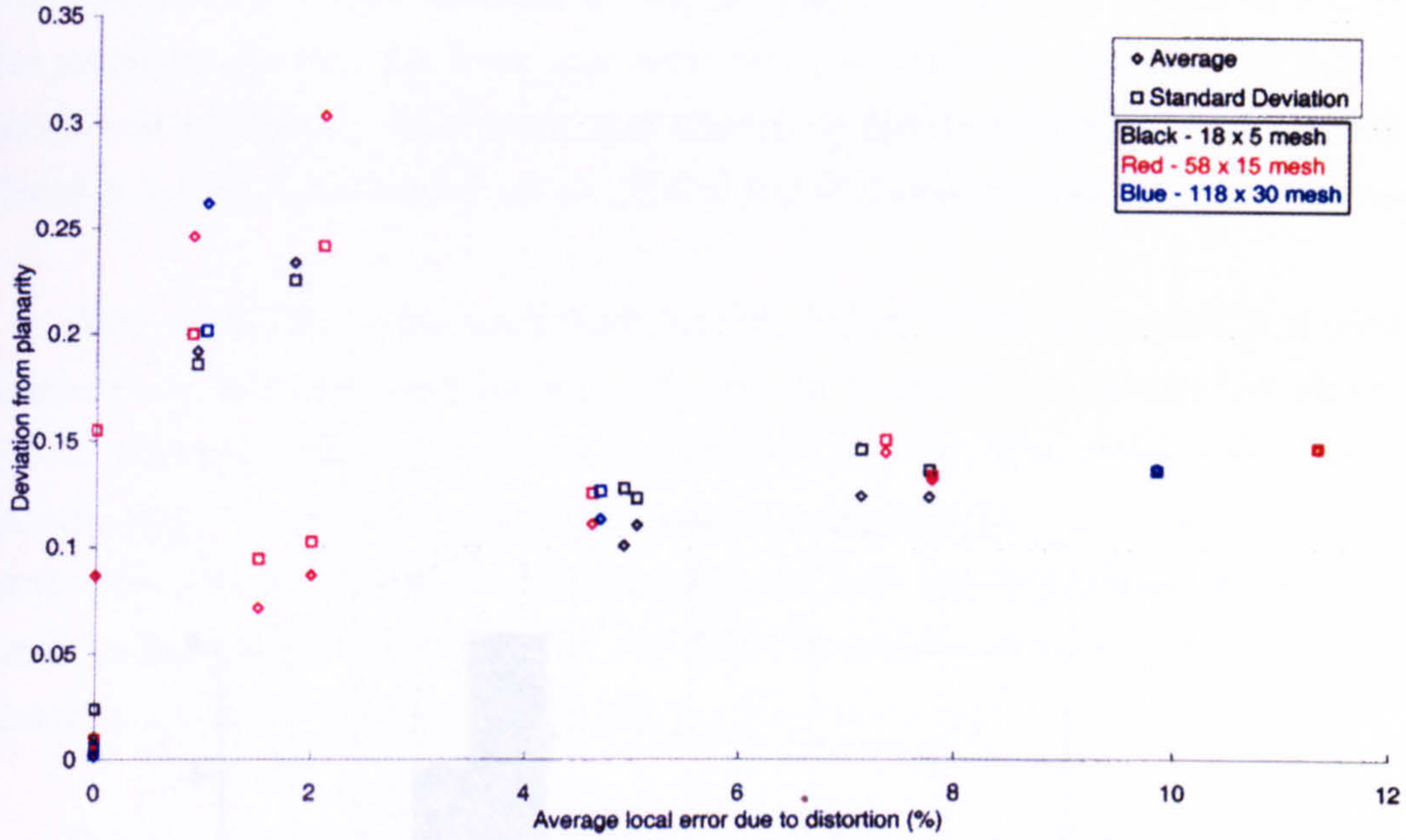


Figure 5.14: Variation in solution error with facial deviation from planarity for all distorted meshes

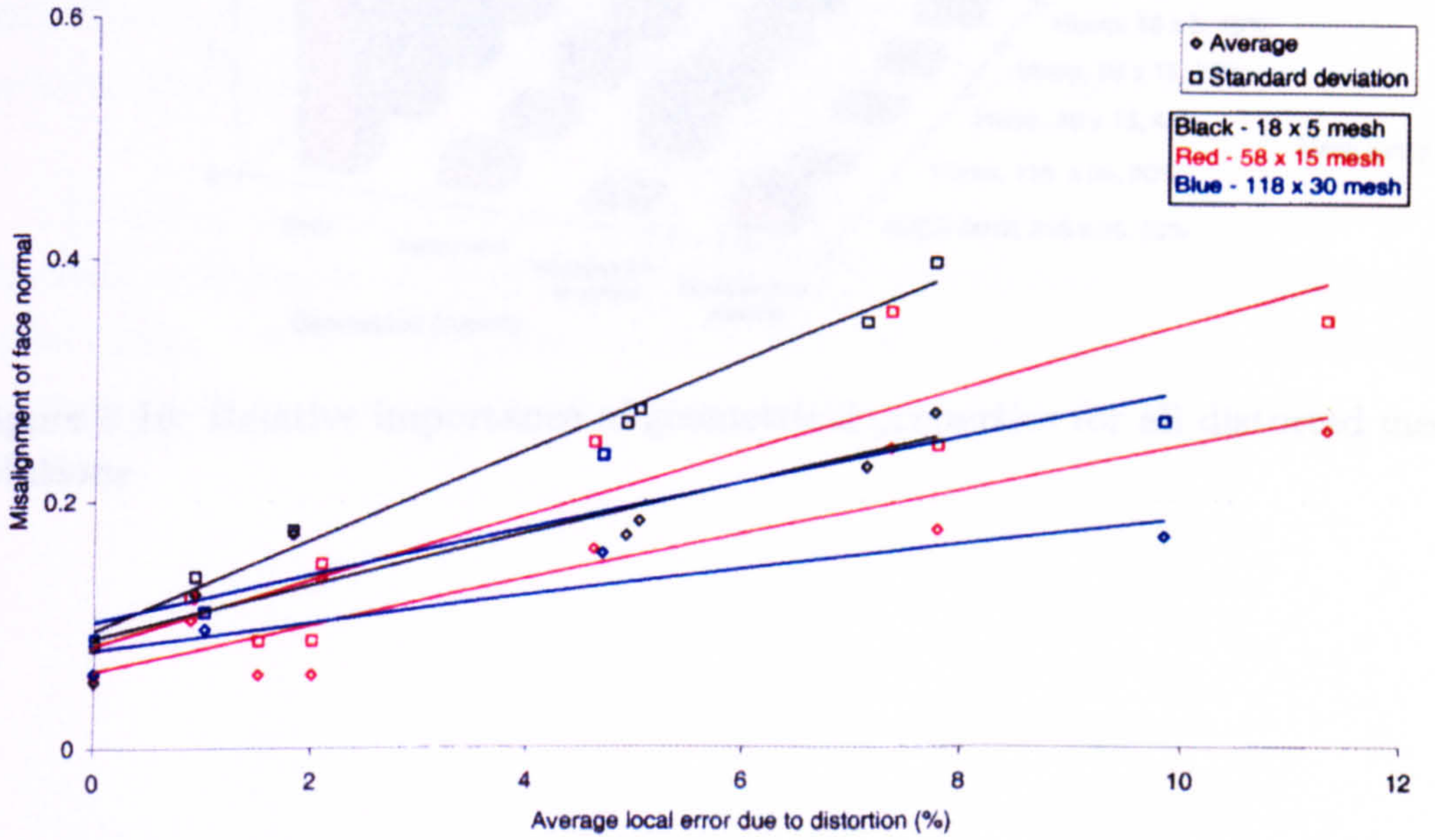


Figure 5.15: Variation in solution error with misalignment of facial normal for all distorted meshes

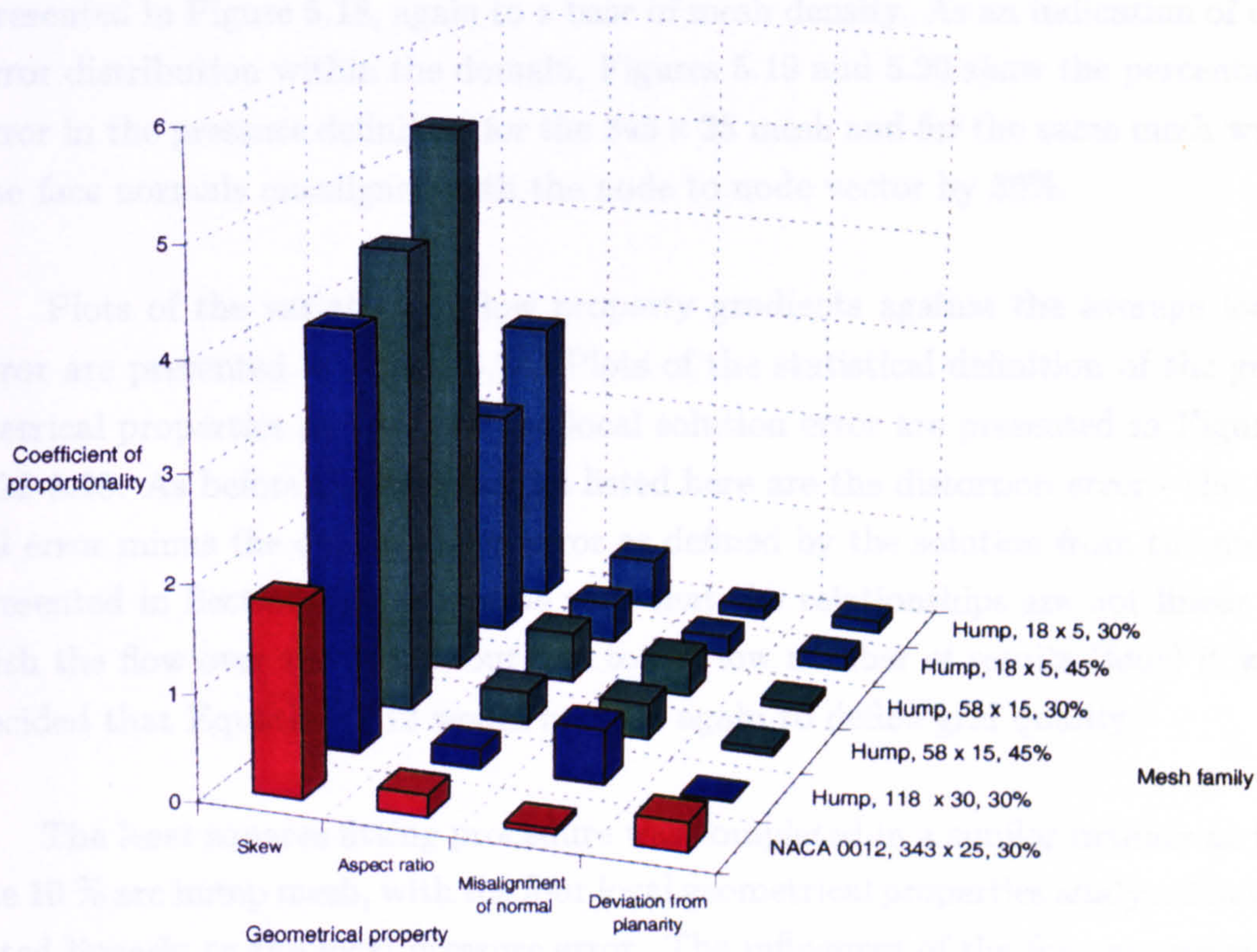


Figure 5.16: Relative importance of geometrical properties for all distorted mesh solutions

5.5.2 Flow around 2D foil

For discretisation error analysis all four meshes at an angle of attack of 4 degrees presented in Section 4.3 were analysed, and for distortion analysis the 343×25 mesh was analysed. This mesh was altered in the three manners prescribed in Section 5.4 with movement set at 30% of the distance between the cell centroids.

The variation in lift coefficient for the foil with the four meshes is plotted against the relative mesh density (H_1/H_i) in Figure 5.17, as are the resultant lift coefficients obtained from the 3 distorted meshes. The average local error is presented in Figure 5.18, again to a base of mesh density. As an indication of the error distribution within the domain, Figures 5.19 and 5.20 show the percentage error in the pressure definition for the 343×25 mesh and for the same mesh with the face normals misaligned with the node to node vector by 30%.

Plots of the variation of flow property gradients against the average local error are presented in Figure 5.21. Plots of the statistical definition of the geometrical properties against average local solution error are presented in Figures 5.22-5.25. As before the error values listed here are the distortion error - the total error minus the discretisation error as defined by the solution from the mesh presented in Section 4.3. It can be seen that the relationships are not linear as with the flow over the hump, but due to the low number of results (four) it was decided that Equation 5.12 would be used again to define grid quality.

The least squares fitting procedure was completed in a similar manner as for the 10 % arc hump mesh, with the four local geometrical properties analysed being fitted linearly to the local pressure error. The influences of the four geometrical properties, by least squares fitting, are presented in Figure 5.16.

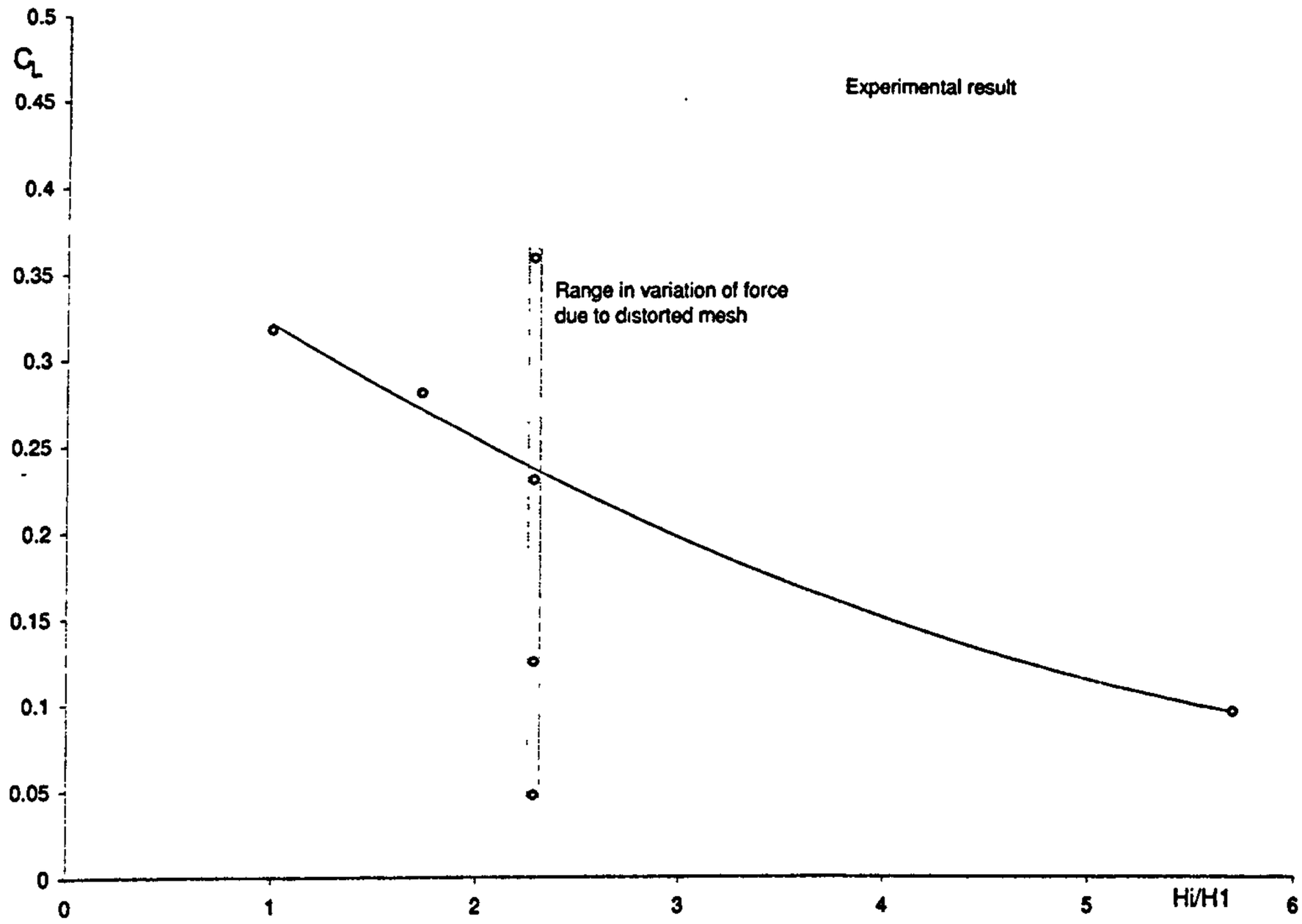


Figure 5.17: Error in force for NACA 0012 foil at 4° with mesh density variation

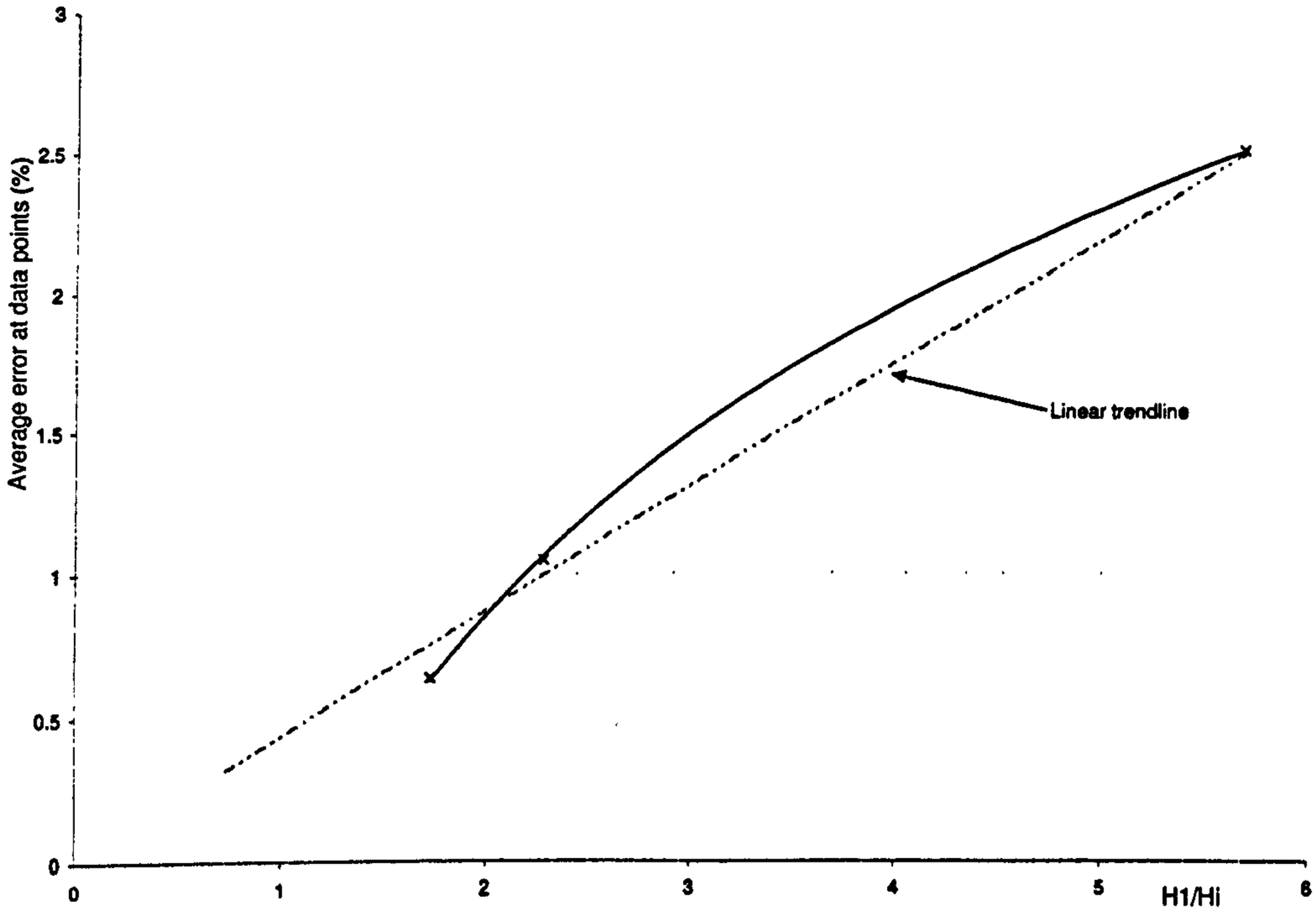


Figure 5.18: Variation in average local error with mesh discretisation for NACA 0012 foil

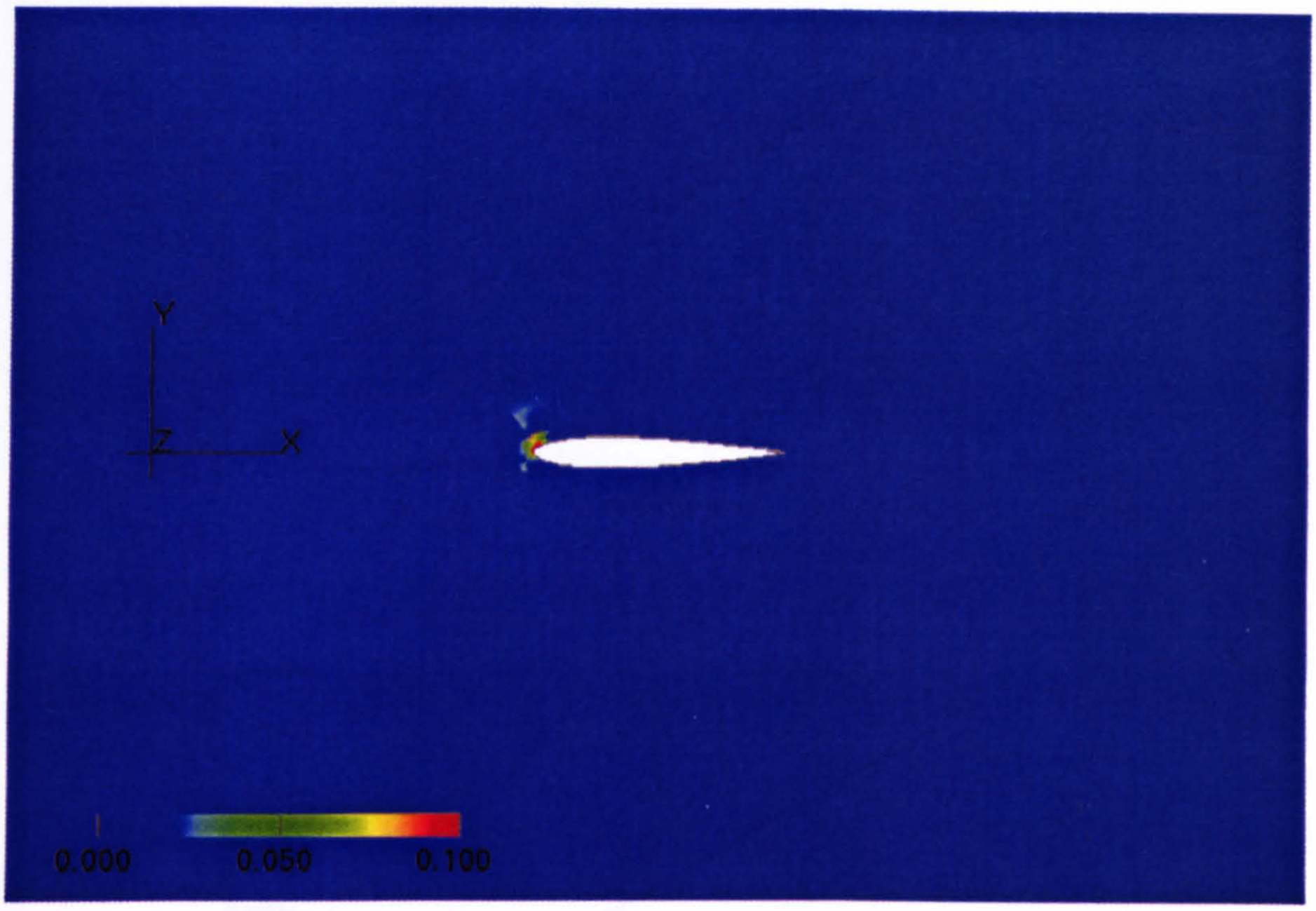


Figure 5.19: Distribution of local error for 58×15 mesh over NACA 0012 foil

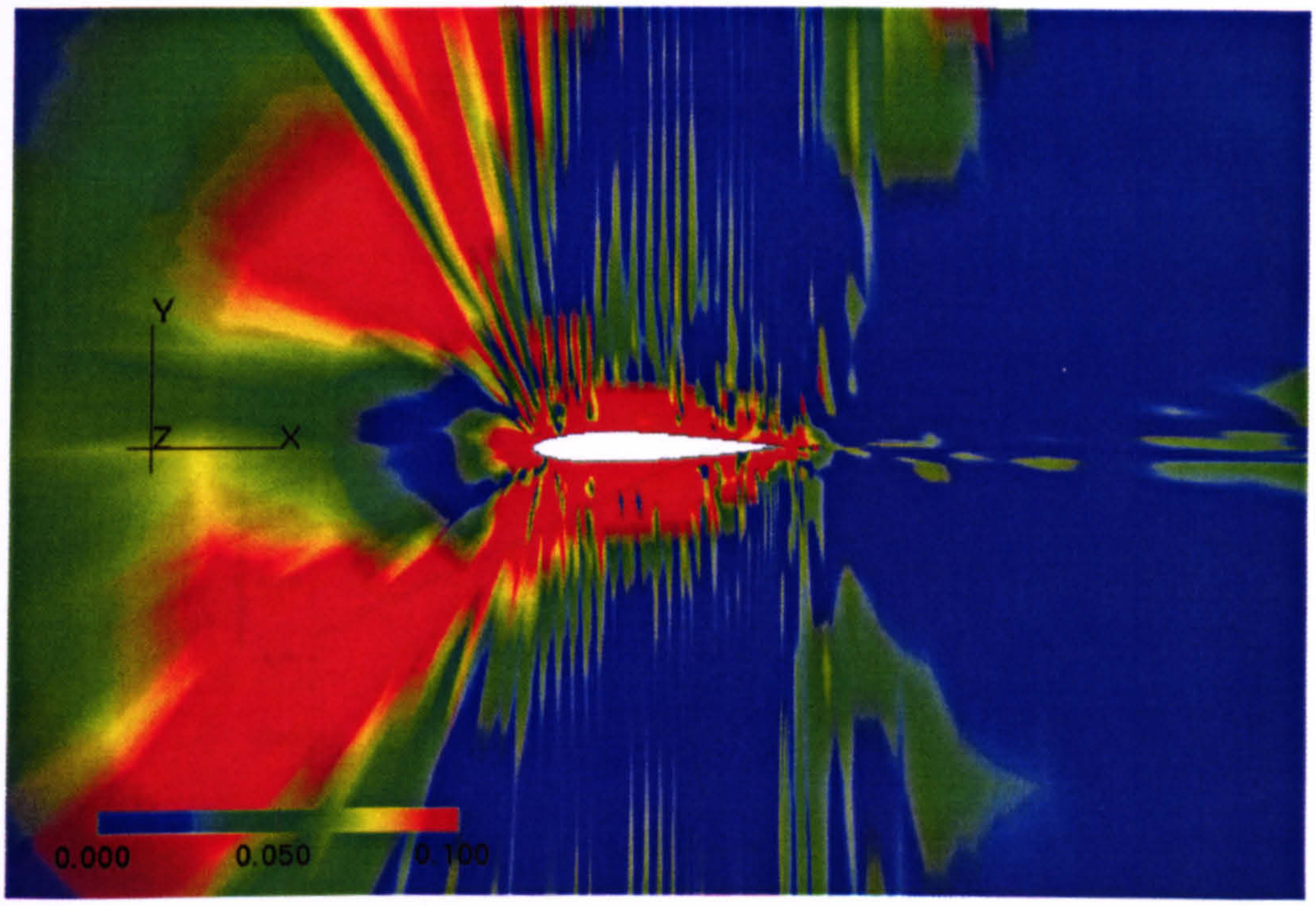
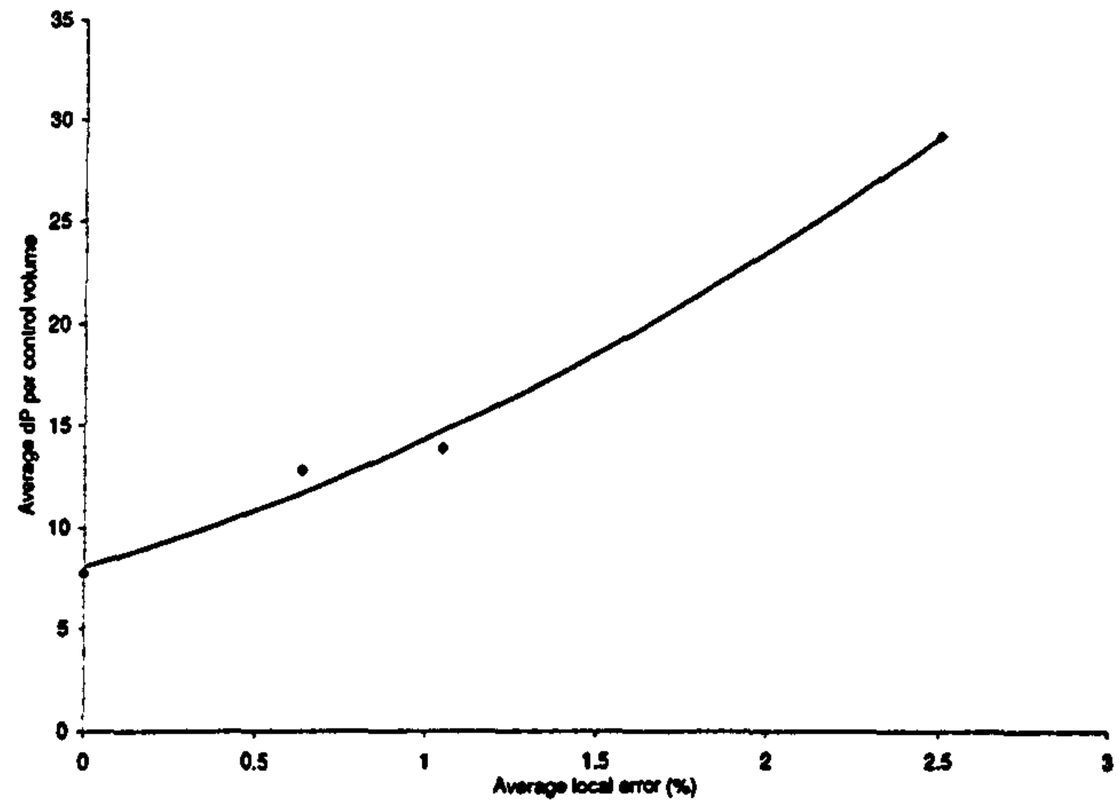
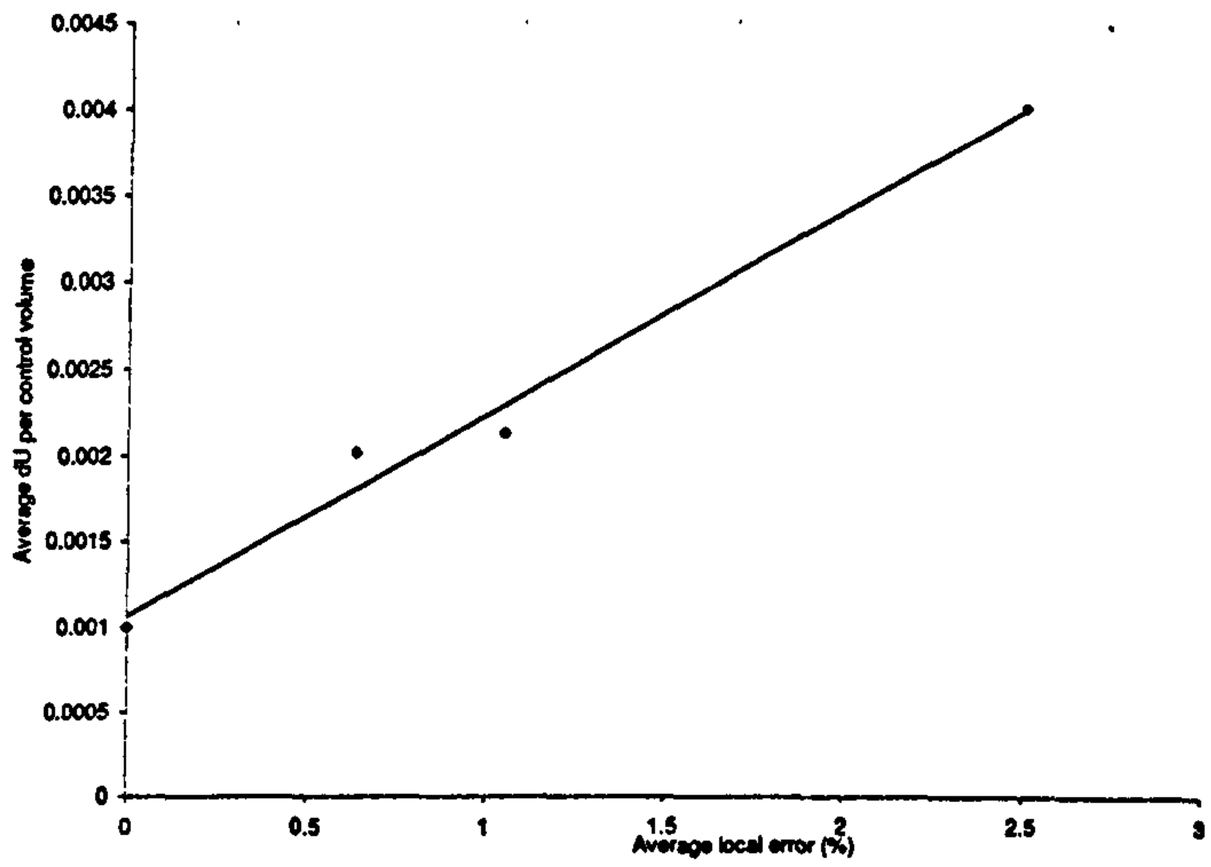


Figure 5.20: Distribution in local error for 58×15 mesh with distortion





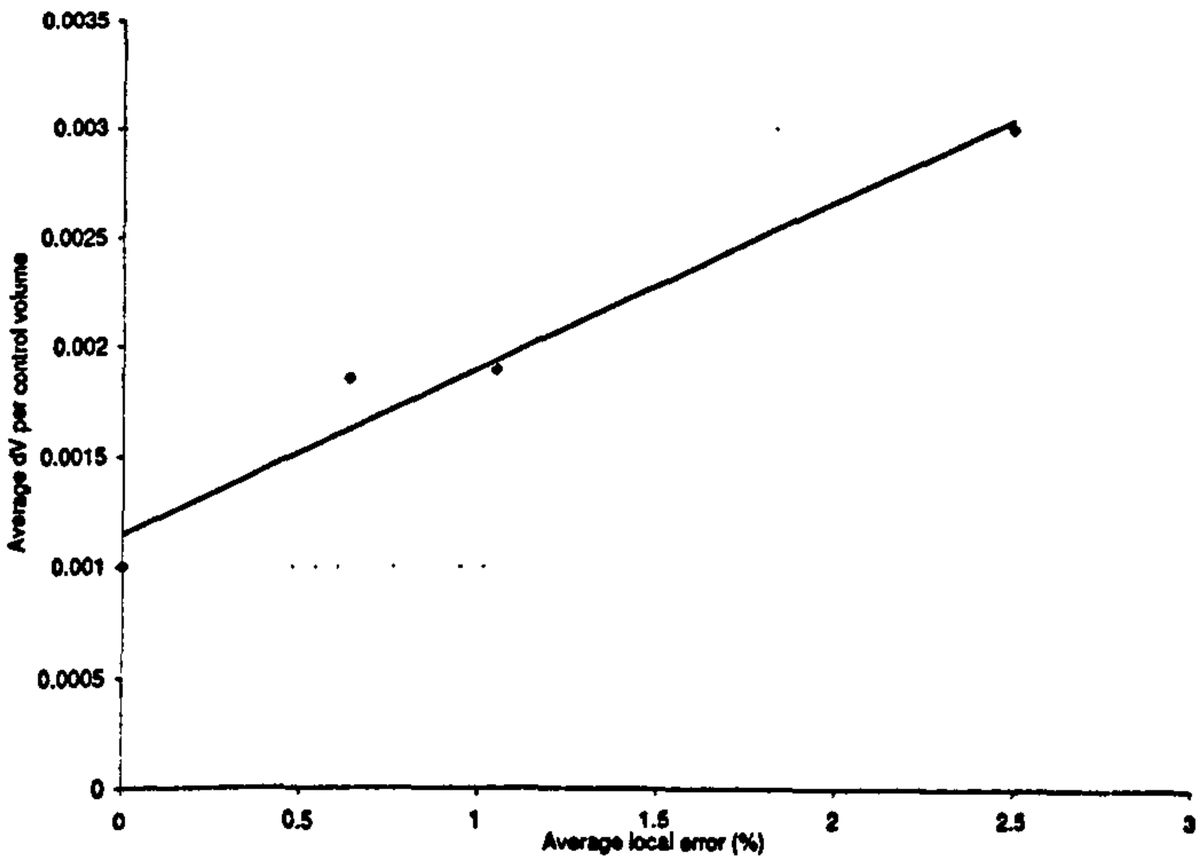


Figure 5.21: Variation in differential of flow properties (P, u, v respectively) over NACA 0012 foil

UNIVERSITY OF SOUTHAMPTON LIBRARY

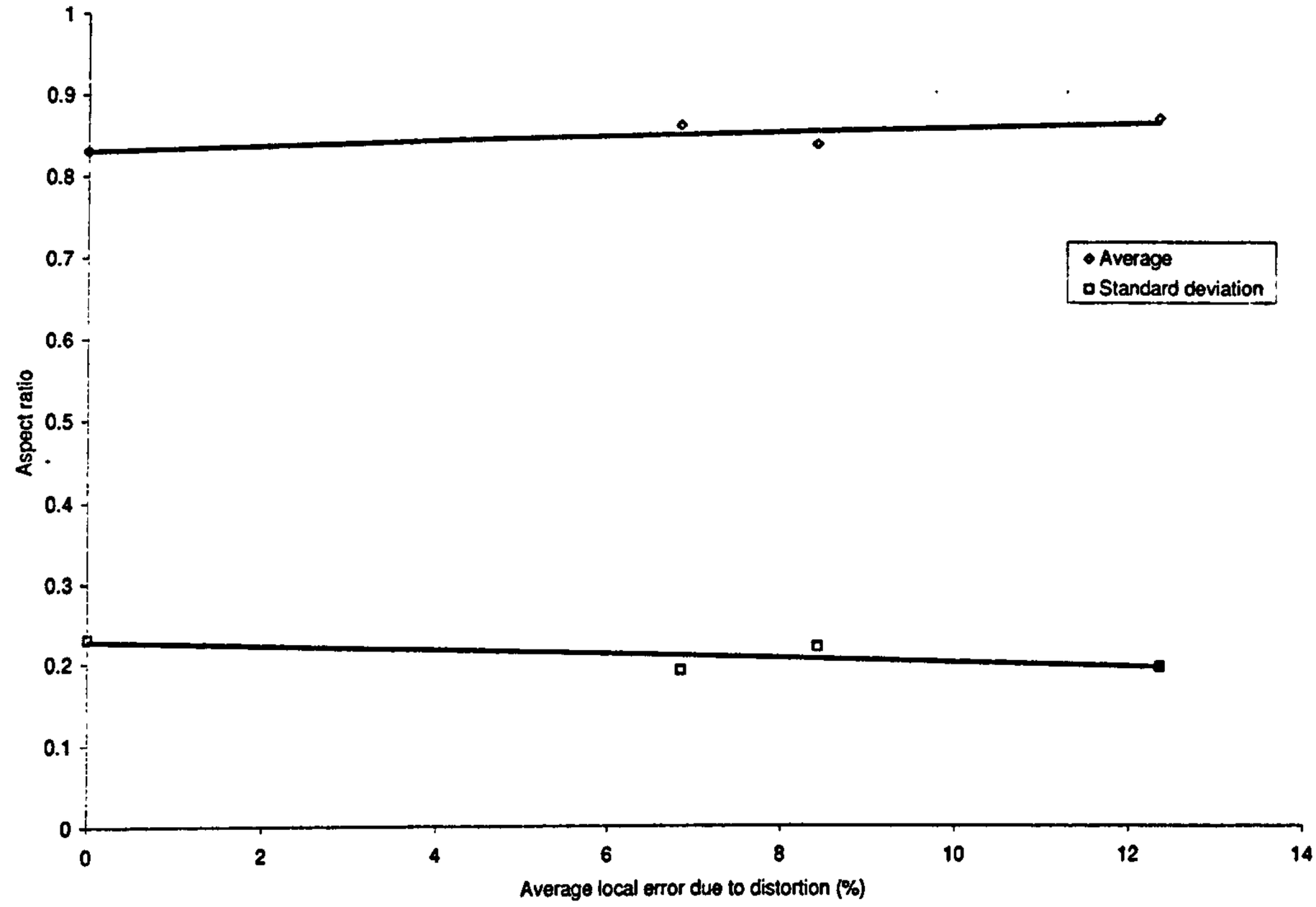


Figure 5.22: Variation in solution error with facial aspect ratio for distorted meshes

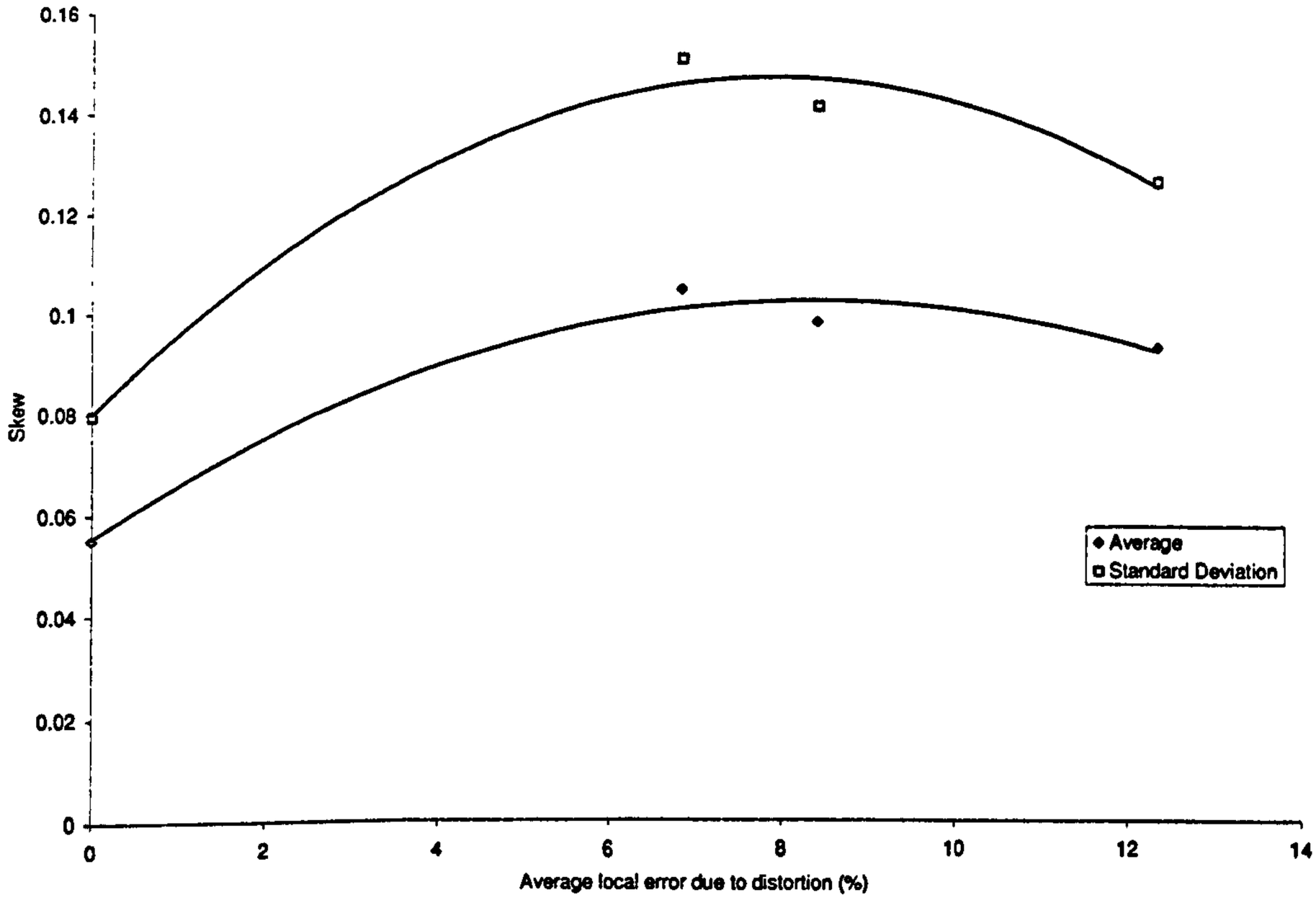


Figure 5.23: Variation in solution error with facial skew for distorted meshes

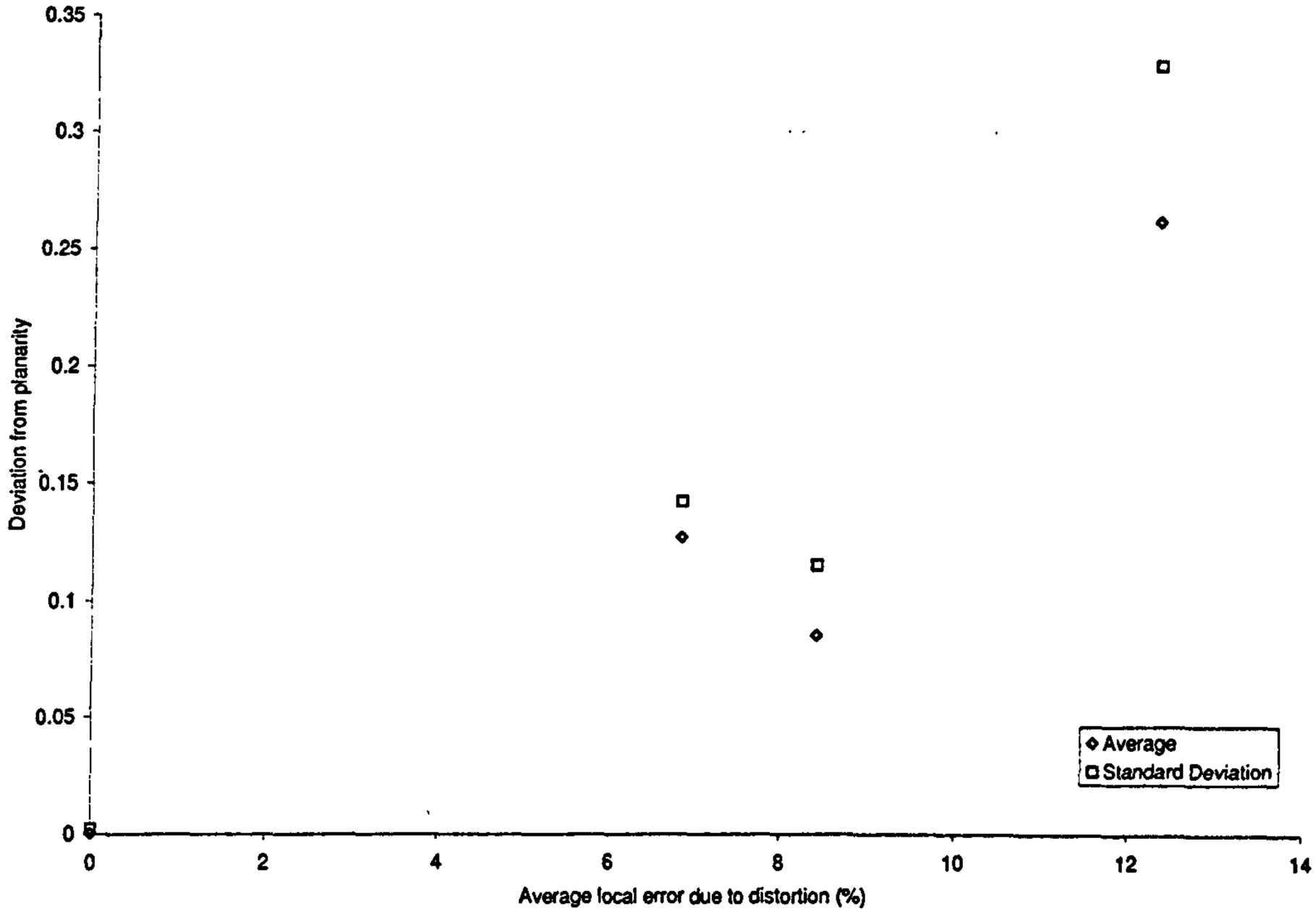


Figure 5.24: Variation in solution error with facial deviation from planarity for distorted meshes

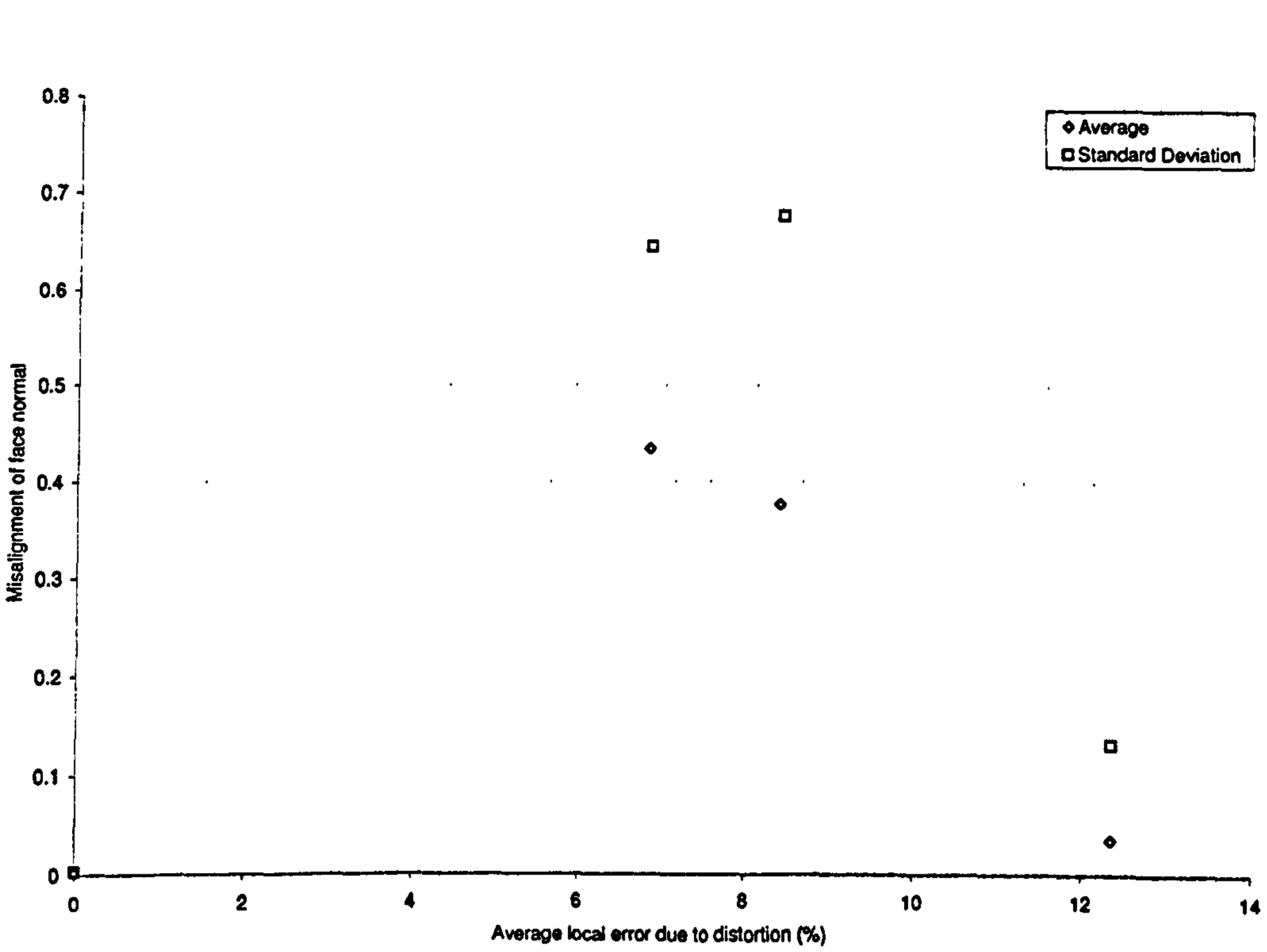


Figure 5.25: Variation in solution error with misalignment of facial normal for distorted meshes

5.6 Discussion of numerical results

Figures 5.5 and 5.17 shows the convergence characteristics of the flow code for increased mesh density, and indicates the necessity of a mesh dependence study to gain trustworthy results. Figure 5.6 shows that the local error in the mesh, when compared to the linear trendline also presented, varies linearly with mesh discretisation, as would be expected of a first order spatial discretisation code. The values 'tail off' towards lower H_1/H_i values because the error results are taken with respect to a finite solution ($H_1/H_i = 1$). The variation in local error for the NACA 0012 foil shown in Figure 5.18 does not have as linear a relationship as the hump flow which can be attributed to two factors. Firstly only 4 different readings were taken as opposed to 8 for the hump flow, and secondly the relative mesh density distribution varies between the meshes (e.g. the second and third mesh have the same number of nodes around the foils surface, but a different number normal to the surface).

Figures 5.7, 5.8 and 5.17 show the variation of results with a mesh of the same density but varying distortion. While the levels of distortion induced in these meshes was high, it is evident that mesh distortion has a large influence upon the solutions accuracy. Hence a purely mesh density based convergence study is insufficient to produce results of high quality and repeatability.

Figure 5.9 demonstrates that the vast majority of the solution error is upon the hump wall surface. This is expected when Figures 4.12 and 4.13 are considered; the vast majority of the flow disturbance is within the same area. Figure 5.11 shows that the solution is most sensitive to changes in the pressure reading, with a squared relationship between pressure and error compared to a linear relationship between the error and velocity gradients. This difference can be expected when considering Bernoulli's equation relates velocity to pressure by a squared relationship.

Figure 5.19 and 5.20 show the difference in error distribution for a mesh of the same density and distribution but of differing grid quality. It is evident that the errors and hence the solution of a poor quality mesh are very dependent upon the mesh orientation.

Figures 5.12 to 5.15 and 5.22 to 5.25 do not provide conclusive proof of the

relative importance of the various geometrical parameters, mainly because the methods of mesh distortion failed to alter the parameters independently. Each form of movement altered more than one property, and in a non-linear manner. Points that can be drawn from the property averages presented are :

- Aspect ratio has a linear relationship to error, that is not affected by mesh discretisation;
- Facial skew has a strong proportional relationship to error that is also strongly dependent upon the mesh discretisation;
- Facial deviation from planarity does seem to have a proportional effect upon error, but is non-linear and strongly dependent upon cross coupling with other geometrical properties;
- Misalignment of the facial normal has a proportional relationship to error and appears to be independent of mesh discretisation, but there are clearly non linearities in the relationship.

It can be seen also that the variation of the standard deviations for each of the solutions follows the same trends as the averages. From this it can be concluded that the overall statistical distribution of values appears to maintain a similar profile for all the meshes.

The least squares fit, presented in Figure 5.16, strongly suggests that facial skew is by far the most important parameter to consider when reviewing solution error for both irrotational and rotational flows. Facial aspect ratio and misalignment of the face normal both vary with the mesh density, but the level of variance is only apparent for aspect ratio at higher mesh densities. Facial deviation from planarity has been defined as of much less importance for the hump flow than for the rotational flow around the foil.

The relative cost of the mesh analysis is low with respect to the time required for one iteration, as shown in Table 5.2. The computational cost of analysing the flow gradients is very low indeed (approximately 0.75 % of the CPU time for one timestep of the flow solver), while the time required for analysing the geometrical quality of the mesh is about 75% of the flow solver CPU time for one iteration.

Mesh number	CPU time per iteration	CPU time for geometric analysis	CPU time for flow analysis
foil 1	5.92	4.55	0.03
foil 2	15.26	11.46	0.09
foil 3	20.2	15.18	0.16
foil 4	35.7	26.43	0.3

Table 5.2: CPU time usage for mesh quality analysis, in comparison to flow solver CPU time

5.7 Summary

A numerical analysis of the errors present in the flow solver devised has been accomplished, and sources of geometrical distortion error detailed.

The numerical tests carried out have led to the conclusions that :

- pressure gradients provide a suitable definition of flow gradients, and are more sensitive to solution error than velocity gradients;
- geometrical parameters do have a marked effect upon solution accuracy, more so at higher mesh densities;
- of the four geometrical parameters tested skew is the most important, with aspect ratios, misalignment of normals and deviation from planarity being influential in that order.
- the CPU requirements of the analysis are very low (smaller that the time required for one timestep), especially when taking into account the reduction in solution time and increased accuracy possible with a more optimised mesh.

It must be stressed that the values obtained were for the numerical solver detailed in Chapters 3 and 4, and the relative values may vary considerably with another numerical code. What has been presented in this Chapter should be seen as a generic method of assessing the influence of geometrical grid quality and flow gradients upon a given solver. Also of note is the different ratios of influence between the irrotational and the rotational cases, as shown in Figure 5.16, which suggests a dependency of the optimal geometrical mesh upon the flow

regime. This conclusion is shared by Habashi et al [23], where it is stated that "the concept of grid quality in terms of evaluating its skewness is meaningless if dissociated from the equations being solved, as well as from the flow conditions".

6 Mesh adaption

6.1 Introduction

Chapter 5 has detailed a quantitative measure of mesh quality with respect to geometrical parameters. This information can be used to improve the mesh quality and therefore improve the solution accuracy for a given number of data points. In addition the calculation of pressure gradient, presented in Section 5.6, is a clear indicator of discretisation error.

The concepts presented in this Chapter are for the various methods of mesh adaptation possible and how the adaption can be controlled. Both mesh movement and mesh division are incorporated within the flow solver of Chapter 4, and the increase in efficiency and accuracy presented.

6.1.1 Importance of mesh adaption

The discrete modelling of physical continuum by computational methods inevitably introduces a dependence of the solution upon the discrete positioning of data points used, as detailed in Chapter 5.

The general trend in large scale CFD problems is to use fine grids as much as possible to resolve physical flow features. Mesh independence is sought via using finer and finer meshes until the solution no longer changes to any significant amount. With complex three dimensional flow this can result in massive problems, both in memory storage requirements and CPU time. This relatively crude method of gaining mesh independence is due to the fact that meshes are arbitrarily generated, with only the operators knowledge and experience dictating where points should be concentrated. This method may also have undetected underlying geometric pathologies such as cellular twist that will introduce secondary mesh dependencies, just as strong as those being removed [23].

Thus it can be reasonably concluded that wherever possible the user should be removed from the mesh definition; it should instead be carried out via a form of automated adaptive process. Recent years have seen a rapid development of such processes, normally based on some form of *a-posteriori* error [28, 83, 84].

The major advantage of such methods over the arbitrary use of finer meshes is the saving in resources, principally computational time and memory. With the adaption of the computational mesh via quantifiably selected criteria, only the areas requiring refinement are altered. Thus the number of geometrical entities is minimised for a given solution accuracy, resulting in lower memory storage requirements and lower CPU usage in the solution phase. In summary, mesh adaptation increases solution efficiency for a given level of accuracy.

A second reason for the use of mesh adaptation, one that is often overlooked, is in areas on complex geometry. Boundaries with large curvature in close proximity of each other often lead to very poor geometrical grid quality. Such a distribution of mesh points often causes numerical instabilities in the solution. Adaption of the mesh, to better suit the point distribution to the given geometry, reduces such instabilities and creates a better behaved solution.

6.1.2 Methods and control of adaption

The two main methods of grid adaption in use are stretching and splitting, or variations of either. The easiest method - stretching - relocates mesh points, clustering them in areas of high flow gradients. Such methods have been used successfully for a number of years [87, 88, 89], most often for supersonic applications where accurate capture of the shock wave is required. Advantages of this method are the lack of creation or destruction of geometrical entities, while a common problem is excessive mesh movement resulting in highly skewed grids.

Cellular splitting to gain greater resolution is more computationally expensive than relocation of existing entities but avoids creating skew in the mesh. By avoiding such geometrical issues accuracy (particularly in viscous applications) is improved and dissipative errors are reduced [90].

The instigation, and subsequent control, of mesh adaption is normally achieved

by the use of a flow property gradient weighting function. Control volumes are concentrated in areas of high flow gradients and removed from areas of low gradients. More complicated and generalised schemes have been divised, such as that by Brackbill and Saltzman [91] which incorporates grid smoothness, orthogonality and volume variation as well as the basic gradient weighting function.

The work presented uses a mixture of methods to instigate the mesh adaption, combining both geometrical and flow solution parameters as presented in Chapter 5. The actual method of adaption also varies, and is chosen according to the manner in which the geometrical entity was selected. If flow gradients are too large control volumes will be split, whereas if a face has low planarity its vertices will be moved to compensate. Thus the method of adaption will be that best suited to the local problem.

6.2 Literature review

It is now commonly recognised that grid adaption is a manner in which solution accuracy can be improved [20, 81], without recourse to higher order numerical schemes. Other uses of grid adaption are to increase efficiency through mesh optimisation; maintaining the same overall number of control volumes but redistributing them to obtain the best possible solution [23]. The third possible use of grid adaption is within a multi-grid scheme, where recursive sub-division of cells allows the different levels of grids to be generated.

6.2.1 Control of adaption

There are three main ways in which a mesh can be adapted; an error estimator function, geometric limiters or flow solution limiters.

Geometrical limiters have probably been in use for the greatest length of time in some form. The general techniques used in unstructured mesh generation of advancing front and Delaunay triangulation are geometrically controlled mesh adaption [92, 93], Since then more refined techniques have been utilised to increase mesh quality, incorporating edge lengths, smoothness and domain

boundary curvature [33, 85, 89, 94, 95]. These methods are still relatively basic, being dependent upon mesh element size in the main. Further developments, incorporating mesh orthogonality, are now being promoted by Jacquotte, McRae and Laflin, and Hassan and Probert [19, Chapters 33-35], and the relative importance of cell alignment and skew have been commented upon [19, Section 34.6].

The second method of controlling mesh adaption is error estimation. This is where an estimate is obtained by the difference between the computed solution, constant by element, and a reconstructed linear solution in order to provide a measure of the grid quality; that is, the ability to correctly represent the flow characteristics. The concept underlying this process is to have the error constant throughout the domain, thus minimising the global error of the solution [23]. This method has been used successfully by a number of parties [83, 84, 96], with Trépanier et al providing a detailed methodology. The grid management algorithm is described in three separate topics; the geometry induced grid, the flow induced grid and grid re-meshing due to poor quality.

The flow induced grid adaption created by Trépanier et al strives to equalise the local error for all of the computational cells. This is accepted as being the optimal grid for the numerical solution. The error estimated was the integration of the difference between the piecewise computed solution in each element and a piecewise linear solution. This method is described in greater detail by Ilinca, Camarero, Trépanier and Reggio [83]. A full listing of the equations and mathematical requirements is provided, as well as adding extra depth to some aspects of the earlier publication [96]. Once the error has been estimated for all given volumes it has to be spread evenly over the domain. Trépanier et al assume the error to be proportional to grid size, and the characteristics of the next grid are thus obtained by scaling. Dynamic re-meshing is used to satisfy the requirements of the geometry and flow induced grids. Global re-meshing can be expensive so the system used consists of local actions to refine, coarsen, cure and/or smooth the evolving grid. The methods of refinement and coarsening are discussed, and a method for grid curing proposed. This is based upon a quantitative measure of grid quality for a 2D mesh involving cell area and edge lengths -

$$\text{QUALITY} = \frac{4\sqrt{3} \quad \text{AREA}}{|\text{EDGE}_1|^2 + |\text{EDGE}_2|^2 + |\text{EDGE}_3|^2} \tag{6.1}$$

Figure 6.1 details the cellular topology upon which Equation 6.1 operates. Thus

a triangle has a value of 1 if equilateral and 0 if degenerate. A watershed limit of 0.4 was deemed to define a bad triangle.

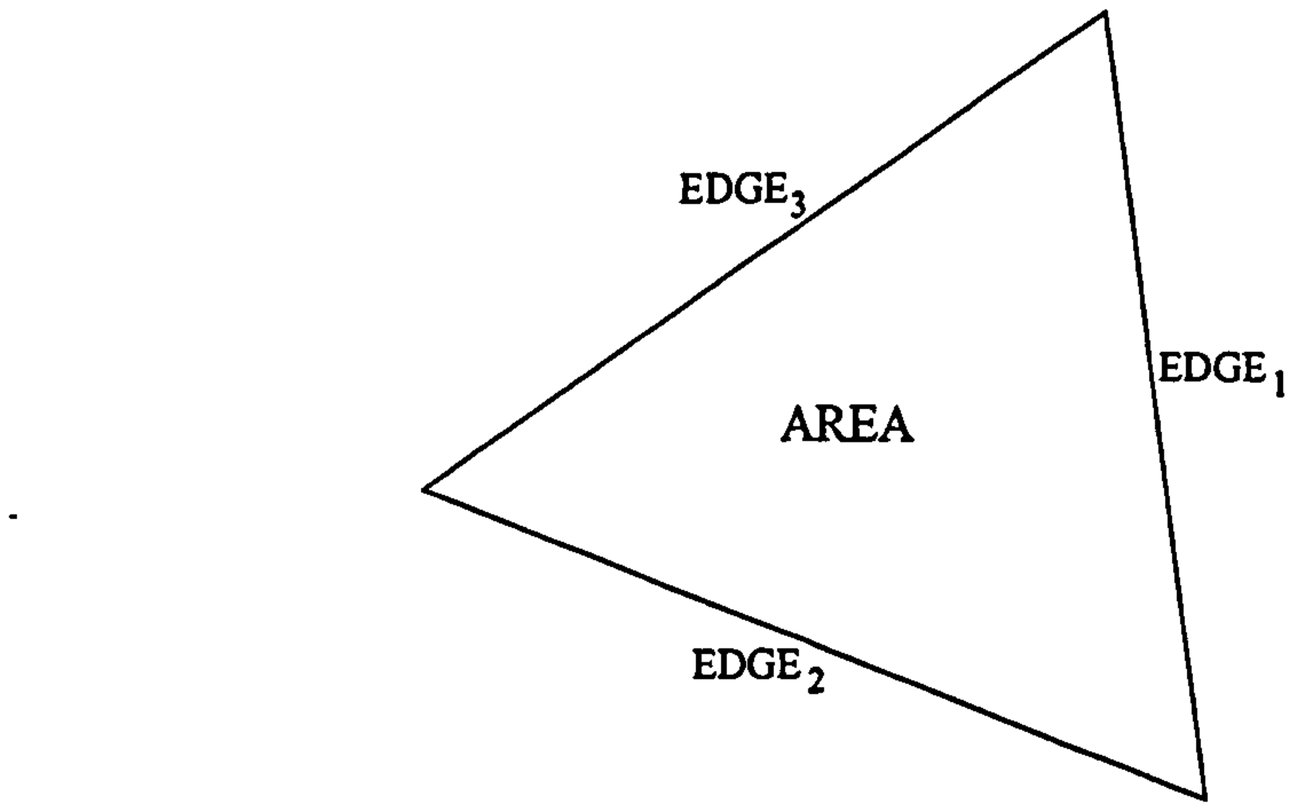


Figure 6.1: Cellular description used by Ilinca et al. to define mesh quality

However the major drawback of these error estimation methods is that they do not correctly take account of the error due to non-uniform grid spacing [81]. Another problem of this technique is the tendency to produce isotropic meshes due to the underlying principle of making the length scale of the elements the same in all directions. This method works satisfactorily for flows possessing large gradients in all directions, but not for highly directional flow features such as wakes and vortices [23]. Habashi et al [23] have more recently devised an error estimator that eliminates most of these problems, but as yet only in two dimensions.

By far the most common method of controlling mesh adaption is the assessment of local flow properties with respect to some predefined limit. Error estimators often use flow properties to define the error, but not as the primary qualifier. Examples of this technique most commonly use the local Mach number or density for compressible flows [97] and pressure or velocity for low Mach number or incompressible flows [98, 99, 100, 101]. Parthasarathy and Kallinderis [98, 99] utilise a cell tree data structure to increase the efficiency of the solver, and allow both cell division and cell merging to allow increased accuracy without increasing the problem size greatly. The advantage of using the local flow gradients to control adaption is that it is a proven and easy to apply technique, however it can cause the grid to become highly non-uniform and contain areas of poor geometrical grid quality.

One of the most recent papers published that incorporates adaptive gridding with the Euler equations is Riemslagh and Dick [101]. The concept of grid adaptation used was to start with a completely uniform grid and locally adapt it to capture interesting flow features; in this research these were deemed to be shock waves, stagnation regions and the trailing edge of the foil (NACA 0012). The criteria used to refine the mesh was based upon the pressure difference over an edge. If

$$|p_i - p_j||EDGE_{ij}| \geq \frac{|p_{max} - p_{min}||EDGE_{ref}|}{S_p} \tag{6.2}$$

where p is the pressure, L the edge length and S a sensitivity parameter, then $EDGE_{ij}$ would be refined by placing an extra node in the centre of the edge. The reference edge length was taken as the foil chord length. This criteria is used to capture shock waves and stagnation regions. A second criteria accounted for the entropy gradient in exactly the same formulation as the pressure gradient in Equation 6.2, with S_p being exchanged for S_s. This criteria was used to trigger shock wave and tangential discontinuity capture. The sensitivity constants were chosen as S_p=250 and S_s=60.

Within the flow model three such adaptations were carried out, with smoothing occurring after each. On the finest multigrid mesh this adaptation caused an increase of nodes from 1944 to 8566 (an increase of 341%) while the coarsest mesh only had increase of 14% on an initial quantity of 158 nodes. This method had as yet only been applied to 2D problems.

6.2.2 Method of adaption

As discussed previously there are two key methods in use presently in order to optimise a given mesh; either refinement of the mesh or redistribution of the mesh. Changing the solution method to a higher order scheme has been used, but has no significant application in field solvers for multi-dimensional problems as yet [19, Part IV].

Mesh refinement requires the addition (or removal) of data points locally in regions of relatively large (or small respectively) error. Problems associated with this method are increased computer time and storage as well as difficulties with the data structure. Unstructured meshes are particularly suited to such a method

[101]. Tree data structures remove many of the data handling problems [84, 99], and the basic algorithms used during this operation lend themselves very easily to creating a multigrid solution to further increase the efficiency of the solution. Cartesian meshes have also been used successfully for mesh refinement [33], but "the method relies on being able to cope with exceptions and is therefore more verbose". In addition, only two dimensional problems have been presented, which suggests that the level of complexity of the algorithms required for realistic shapes will be much greater.

The second method of adapting a mesh is the redistribution of the data points. In this approach points move from areas of relatively low error to areas of large error, thus aiming for an equal distribution of error throughout the domain. The most obvious advantages of this system are the ability to maintain the same mesh connectivity and the lack of requirement to create and destroy entities. It is also inherently possible to use structured grids, thereby utilising the efficiency of such methods. Structured mesh adaption has been employed [88, 89, 94, 97] with good results. An added benefit of this approach is that the grid speed can be mathematically defined simultaneously with the grid movement, thus increasing CPU efficiency [97].

From the study of previous publications it has been concluded that both flow and geometrical quality must be assessed to correctly assess grid quality. A further conclusion is that mesh refinement and redistribution should be used simultaneously to optimise the mesh.

6.3 Control volume splitting

The reason for splitting a control volume is to lower the truncation error created due to the discretisation. As truncation error grows, so also do the flow property differentials between adjacent control volumes. Thus the control of this method has been chosen to be the local flow properties, namely pressure, as detailed in Section 5.6. The manner of control chosen has been to determine the domain mean and standard deviation, and then select those control volumes with a pressure differential greater that 2.365 standard deviations above the mean. The value of 2.365 standard deviations equates to a 1% limit on a normal distribu-

tion curve. Thus approximately 1% of the control volumes will be split at every adaption process.

6.3.1 Geometrical Entity Based Approach

The first approach to splitting an arbitrary polyhedral was based upon selecting a face that was to be split, and then finding a face connected to the first face in the correct direction. This process was continued until a path had been traced around the control volume faces. The selected faces were then split via defining the number of sides, and entities reconnected to define two control volumes. Algorithm 6.1 details briefly each main step in the process. The process was set inside a loop containing all the volumes requiring splitting.

Algorithm : 6.1 (Entity based control volume splitting)

- DEFINE CUTTING PLANE NORMAL
- DEFINE EDGE THAT IS TANGENTIAL TO CUTTING NORMAL
  1. FIND THE FACES THAT ARE CONNECTED TO THAT EDGE
  2. IF MORE THAN 2 FACES IN LIST DEFINE ARRANGEMENT
  3. WHILE NEW EDGE IS NOT ORIGINAL EDGE
     (A) DEFINE EDGE OPPOSITE CURRENT EDGE
     (B) FIND OTHER FACE(S) ATTACHED TO NEW EDGE
     (C) IF MORE THAN 2 FACES IN LIST DEFINE ARRANGEMENT
     (D) MAKE THE NEW EDGE THE CURRENT EDGE
- FOR EACH FACE IN CUTTING LIST
  1. DEFINE CORNERS OF FACE
  2. DEFINE MID-POINT OF SIDE UPON WHICH CUTTING EDGE EXISTS
  3. CREATION OF CHILD FACES
     (A) DEFINE NUMBER OF SUBFACES ON ORIGINAL FACE ⇒ BOTH CHILD FACES WILL HAVE SAME NUMBER OF SUBFACES
     (B) ASSESS ORDER OF EDGES ON FACES ⇒ INSERT NEW NODES IN CORRECT PLACE

4. Ensure new nodes and edges are not repeated on other faces to be cut

- Resetting construction nodes
- Resetting node to node connections

This method met with some success, and proved to work for most cases for a number of levels of adaption. A example of this was the adaptation of the mesh around a Wigley hull form, as detailed in Wright [102].

Due to this study occurring before the data presented in Section 5.5 had been calculated the relative influence of the various geometrical parameters was unknown. As a result, the geometrical parameters were given equal weighting, as detailed in Equation 6.3

$$\begin{aligned} \text{QUALITY} = \; & 1.0\text{deviation from planarity} \\ & +1.0\text{skew} \\ & +1.0\text{aspect ratio} \\ & +1.0\text{misalignment of normal} \\ & +1.0\text{volume} \\ & +1.0\text{ratio of volume to adjacent CV volume} \end{aligned} \tag{6.3}$$

As can be seen the influence of volume, and hence discretisation error, was taken into account in this study as well as distortion error via the other geometrical parameters.

The method of determining those control volumes out with acceptable limits was as detailed at the beginning of this Section; the mean for the domain was taken, and those control volumes out with 2.365 standard deviations of the mean were noted. Both volumes of very low and very high quality were listed - the overall error of the solution is determined by the poorest quality area, hence the solution in very high grid quality zones will be over shadowed by the neighbouring zone of poor quality. If the time until solution convergence can be quicken via

the lowering of the grid quality in such areas (i.e. by merging control volumes together) then the whole procedure is made more efficient.

Figure 6.2 show the volumes of unacceptable mesh quality, as defined by geometrical calculations *before* any flow solution calculations have occurred. As can be seen, in this multi-block style generated mesh that geometrical variance is closely linked to the block-to-block geometry. Figure 6.3 shows the adapted control volumes and subsequent control volumes created (having been split); the number of control volumes falling short of the required standard fell from 368 (3.2% of the initial total of 11448 control volumes) initially to just 96 within one level of adaptation. It should be noted that because of the present simplified manner of cell fusion (merely re-joining children of the same parent) the grid can not be made any more coarse than the initial grid. The volumes of grid quality outside the set limits after one level of refinement can be seen in Figure 6.4; clearly the volumes of low geometrical quality have been reduced.

Table 6.1 confirms this visual interpretation with numerical values of grid quality . The standard deviation has significantly dropped, as has the maximum (relating to poor quality).

Mesh	Minimum	Mean	Maximum	Std Deviation
Initial	0.034614	1.21657	6.946052	1.003025
Adapted	0.034614	1.21376	4.926340	0.917516

Table 6.1: Grid quality of Wigley hullform mesh before and after adaptation

However, problems start to appear with this algorithm due to the arbitrary nature of the control volumes and the allowance by the algorithm for the production of polyhedral volumes. Particular areas of concern were those of defining edge sides and corners; a generic definition of a 'corner' that would operate for a polygonal face proved troublesome for example. As entities become more abstracted from regular, symmetric shapes the algorithm has greater and greater difficulty splitting the faces in a symmetric manner, and the path which is 'traced' around the control volume does not necessarily meet with the starting point. Indeed, after three levels of adaptation the algorithm often became insufficient and broke down, leaving un-connected construction nodes and faces. Attempts were made to cover such exceptions but the code became extremely protracted and it was felt that as arbitrary polyhedrals were to be allowed there would always, eventually, be some case that exceeded the algorithm. Hence it was decided that

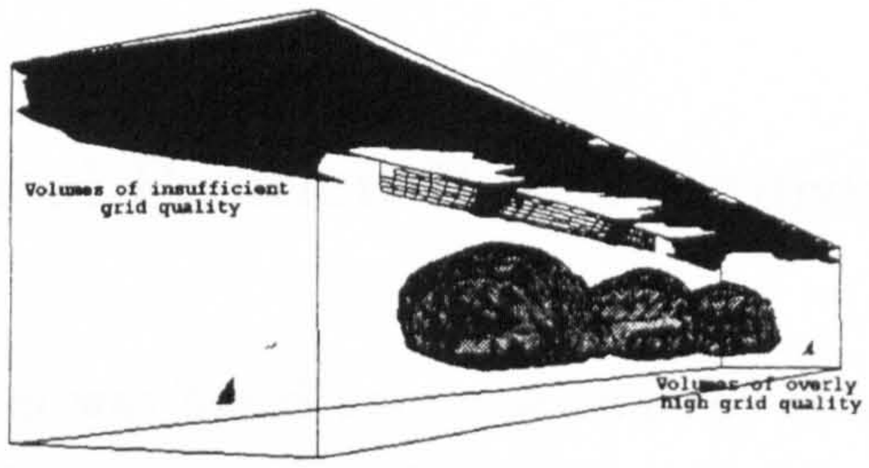


Figure 6.2: Volumes with unacceptable quality in a mesh around a Wigley hullform

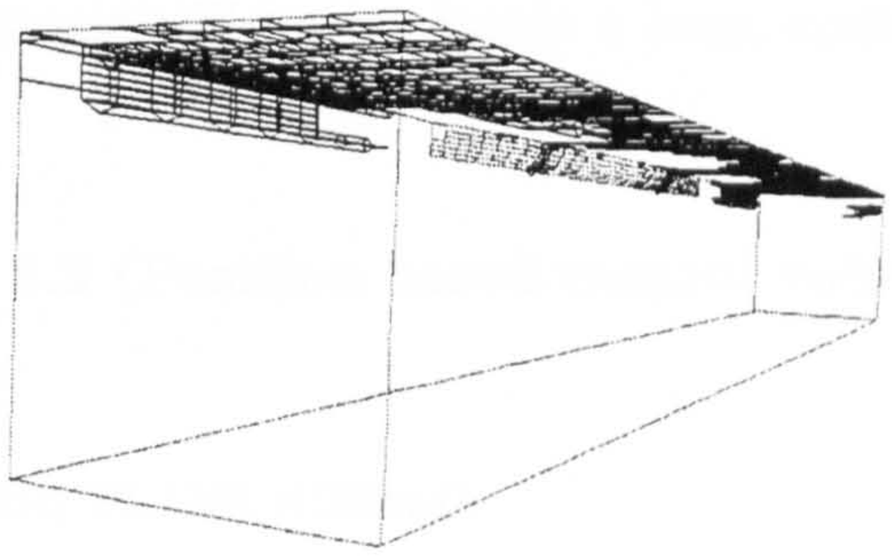


Figure 6.3: Adapted areas of Wigley hullform mesh

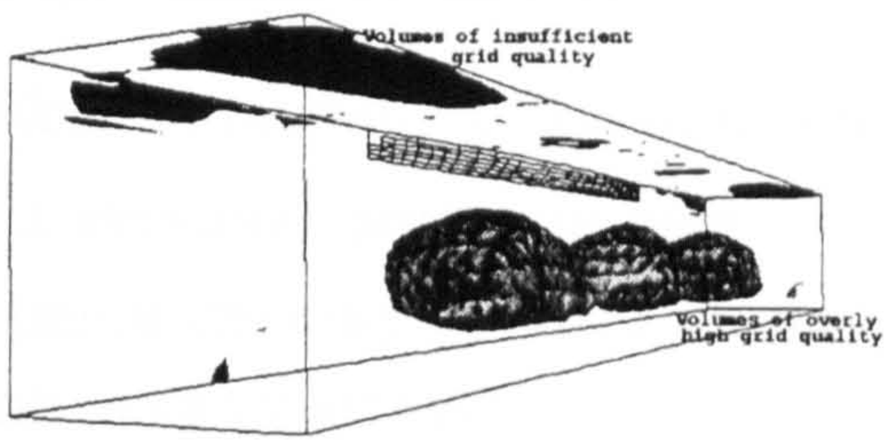


Figure 6.4: Volumes with extreme quality in the Wigley hullform mesh after adaptation

a more appropriate splitting technique for an arbitrary volume was required.

6.3.2 Geometrical Position Based Approach

The preferred algorithm was based upon the concept of a splitting plane. This passes through the volumetric centroid of the control volume, with its normal being defined by the geometrical grid quality algorithm. All the edges on each face attached to the control volume are checked to see if they pass through the plane. This produces a list of faces to be split, and the style in which they require splitting. Because of the non-reliance upon geometrical entity orientation this algorithm is reliable for successive generations of control volumes and degeneration of the volume topology. Algorithm 6.2 details each main step in the process for splitting one control volume. As with the entity based algorithm detailed in Section 6.3.1, the process will be set inside a loop, containing all the volumes requiring splitting.

Algorithm : 6.2 (Position based control volume splitting)

- Define cutting plane normal
- Define volumetric centroid of volume as point on cutting plane
- If (node on boundary) ⇒ Replace flow node with a construction node
- Collect faces and edges to be cut
  1. For all faces attached to control volume
     (a) For all internal edges on face
     - If (edge crosses cutting plane) ⇒ Place face and edge in cutting lists

     (b) For all external edges on face
     - If (edge crosses cutting plane) ⇒ Place face and edge in cutting lists

     (c) If (no edges cross cutting plane) ⇒ Determine which side of plane face is and place in list A or list B

- Order edges and link faces
    1. Count faces and edges
    2. Mark start and finish mark for face in edge list
    3. List repeat edges (all external edges must be listed by two faces)
    4. Check if cutting plane crosses over faces limits more than twice
    5. Check (and mark) parallel cut faces
        (a) If all splitting nodes are pre-existing
        (b) If all splitting nodes are external
        (c) If nodes are joined together in one continuous chain
            $\Rightarrow$ Mark face as to be a parallel cut
- If (number of faces > stack size) $\Rightarrow$ Reset face pointer stack
- Split the edges
    - Normalised position of cutting plane, X, crosses edge at
        1. $0.0 \leq X < 0.2 \Rightarrow$ Use first node
        2. $0.2 \leq X < 0.8 \Rightarrow$ Create new node
        3. $0.8 \leq X < 1.0 \Rightarrow$ Use second node
- Reset the original faces and spawn new face. For each face
    1. Determine which side of cutting plane all construction nodes lie; place in appropriate list (list A and list B)
    2. If (face on boundary & open) $\Rightarrow$ Add flow node replacement construction node to appropriate list
    3. If (face is internal & enclosed) $\Rightarrow$ Remove central construction node from appropriate list
    4. Add splitting nodes for face to both lists
    5. Place nodes in order
        (a) Define angle of node with respect to first node in list
        (b) Sort list of nodes for angle $= 0 \rightarrow 2\pi$

6. If (number of construction nodes in list ≥ 4) add central node
7. Transfer all information into face structure. Original face reconstructed from list A, new face created from list B

- Reset tail of construction node list
  1. Mark first construction node in list
  2. Reset beginning of list to second member
  3. Place first member at end of list
  4. Set this to be the end of the list, not a looped list
  5. Store the ID of the construction node for use by the new data node
  6. Reset global ID of construction node

- Spawn the new flow node
  1. Set up new memory for child flow nodes
  2. Define characteristic length of control volume
  3. Count number of external faces
  4. If (number of external faces $= 0$) $\Rightarrow$
     New node position = centroid of new volume
  5. If (number of external faces $= 1$) $\Rightarrow$
     New node position = centroid of new external face
  6. If (number of external faces ≥ 2) $\Rightarrow$
     (a) Find common construction nodes
     (b) If (number of common nodes $= 1$) $\Rightarrow$
         Convert construction node to flow node
     (c) If (number of common nodes $= 2$) $\Rightarrow$
         Make flow node $\frac{1}{2}$ way between nodes
     (d) If (number of common nodes $= 3$) $\Rightarrow$
         Convert construction node closest to average position
     (e) Remove converted construction node from list of construction nodes defining face if required

- Reorder nodes on faces
    1. Place nodes in order with respect to flow node
- Make splitting face
    1. Collect all splitting nodes
    2. Place nodes in order
    3. If (number of construction nodes ≥ 4) $\Rightarrow$ make central construction node
- Reset status of flow nodes
    1. Flag parent as DEAD
    2. Flag children as ALIVE
- Reset the geometrical entity linking
    1. Reset face to flow node links
    2. Count number of construction nodes attached to flow nodes
    3. Reset construction node to flow node links
    4. Reset flow node to flow node links

6.3.3 Benefits of position based approach

This new algorithm has the advantage of only working upon the edge and node entities, not on the face as a whole. This reduces the number of possible situations greatly; an edge has direction, yet contains only two nodes, while a face can have any number of nodes greater than three to define it, hence creating an infinite number of combinations.

The other major improvement of this algorithm is the use of placing the nodes in order by angle. Previous algorithms tried to maintain the existing node connectivity and ordering but this caused a great number of problems and produced verbose coding. Instead the use of a list containing the required construction nodes, and then sorting the list by angle has proven to be efficient in both length and complexity of code as well as CPU usage. The only variable in this method

is the selection of a point that is defined as being the origin. It is for this reason that the reordering occurs twice. To define the place of the new flow node the construction nodes have to be in order, but the final order of the nodes can only be specified once the flow node position is declared. Thus the first sort places the construction nodes in cyclical order, and the second sort organises them with respect to the flow node.

6.4 Control volume merging

The reasons for merging control volumes together are the exact opposite of splitting a volume; the local error is appreciably less than the norm of the local error over the domain. This means that the solution is being inefficient; the increased accuracy in this area will be negated by the surrounding lower accuracy and hence the CPU resources required for the area are being squandered. As with the control of the volume splitting, local flow properties have been utilised. When the pressure differential between two adjacent control volumes falls below a set limit (adjusted to the required accuracy level), then they are merged together.

6.4.1 Initial method of merging control volumes

To date, the method of merging control volumes together is limited to those volumes that have previously been split from the same parent. While this is limiting, if the initial mesh is suitably coarse then it should not be restricting to the solution efficiency. The basic premise of the approach is to go back a level of the flow node tree data structure, flagging the children as dead, and the parent as alive. The code has been set up to allow eight levels of the tree to be in existence at any one time. Due to the use of a linked list within the tree structure a parent flow node may have more than two children. Hence one or more levels of children can be removed, and the link reset to the required level of children, if deemed necessary. Algorithm 6.3 details the approach used to merge volumes. As with the splitting algorithm (Algorithm 6.2) this will be placed inside a loop that contains a reference to all control volumes requiring merging.

Algorithm : 6.3 (Control volume merging)

- FIND ALL CONTROL VOLUMES OF THE SAME PARENT AND GENERATION
- RESET STATUS OF FLOW NODES AND FACES
  1. FLAG CHILDREN AS DEAD
  2. FLAG PARENT AS ALIVE
  3. FLAG SPLITTING FACES AS DEAD
- RESET THE GEOMETRICAL ENTITY LINKING
  1. RESET FACE TO FLOW NODE LINKS, ALL POINTING TO PARENT NODE
  2. RESET CONSTRUCTION TO FLOW NODE LINKS, ALL POINTING TO PARENT NODE
  3. RESET FLOW NODE TO FLOW NODE LINKS, ALL POINTING TO PARENT NODE

6.4.2 Influences and effects

The most noticeable effect control volume merging has upon a solution is its convergence time. This can be reduced drastically, for the same accuracy of solution, by merging of control volumes in the far field.

It is worth noting that efficiency improvement is not linear due to the maintenance of the present face structure; this results in more than one face between two adjacent control volumes. An algorithm to merge such groups of faces is being developed, but, as with the destruction of levels of the flow node tree, the instances of use is a subject of development. If faces are merged at the same time as the control volumes, then memory and solution time CPU are decreased but mesh adaptation CPU time is increased. The most efficient balance between merging faces and leaving the current structure is case dependant and not resolved as yet.

An important point to be noted about this methodology is that because of the allowance for more than one face between adjoining control volumes, and

the inherent subdivision of the control volumes into tetrahedra, the problem of hanging nodes and subsequent loss of conservation are negated [20, page21].

6.5 Entity stretching and convection

The redistribution of mesh nodes has been used by others to improve solution accuracy via re-location to areas of high flow gradients [23]. However for this work it has been decided that the volume splitting and merging detailed in Sections 6.3 and 6.4 will provide sufficient flexibility and range for accuracy requirements. Instead node redistribution has been used to improve solution convergence rates and stability. This has been achieved by redistributing the construction nodes according to the local grid quality assessment.

As listed in Chapter 5, skew, aspect ratio, deviation from planarity and misalignment of the normal are all assessed and combined - weighted with respect to Figure 5.16 via Equation 5.12 - to define the local geometrical grid quality. This quality rating has been given a limiter and if a given volume has a value greater than this limit then two processes are initiated.

The first approach to improving the geometrical quality of the mesh is a movement of faces to improve orthogonality, hence lowering facial skew and misalignment of the normals. The second and third processes involve the movement of individual construction nodes to increase planarity. Algorithms 6.4, 6.5 and 6.6 detail these three processes. Figure 6.5 shows a representation of a face, and details the labelling of entities.

Algorithm : 6.4 (Construction node relaxation)

- For each Construction node
  1. Define mean position of surrounding flow nodes
  2. Move construction node to this position

Algorithm : 6.5 (Central construction node movement)

- FOR EACH INTERNAL FACE
  1. DEFINE FLOW NODE TO FLOW NODE VECTOR, $N_A\vec{N}_B$
  2. FOR EACH EDGE CONSTRUCTION NODE
     - DEFINE BASE CONSTRUCTION NODE TO EDGE CONSTRUCTION NODE, $CN_B\vec{C}N_E$
     - DEVIATION FROM PLANARITY, $DP = N_A\vec{N}_B \cdot CN_B\vec{C}N_E$
     - MOVE BASE CONSTRUCTION NODE TO LOWER DEVIATION FROM PLANARITY $\Rightarrow CN_B =$ DAMPING FACTOR $\times DP$

Algorithm : 6.6 (Reduction of deviation from planarity)

- DEFINE FACE NORMAL, $\vec{n_F}$
- USING LEAST SQUARES FIT, MOVE BASE CONSTRUCTION NODE TO MINIMISED DEVIATION FROM PLANARITY OF SUB FACES
- REDEFINE FACE NORMAL, $\vec{n_F}$
- MOVE EACH OF THE EDGE CONSTRUCTION NODES TO BRING IT INTO PLANE $\Rightarrow P_{new} = P_{old} + \left(O\vec{P}_{old} \cdot \vec{n_F}\right)\vec{n_F}$

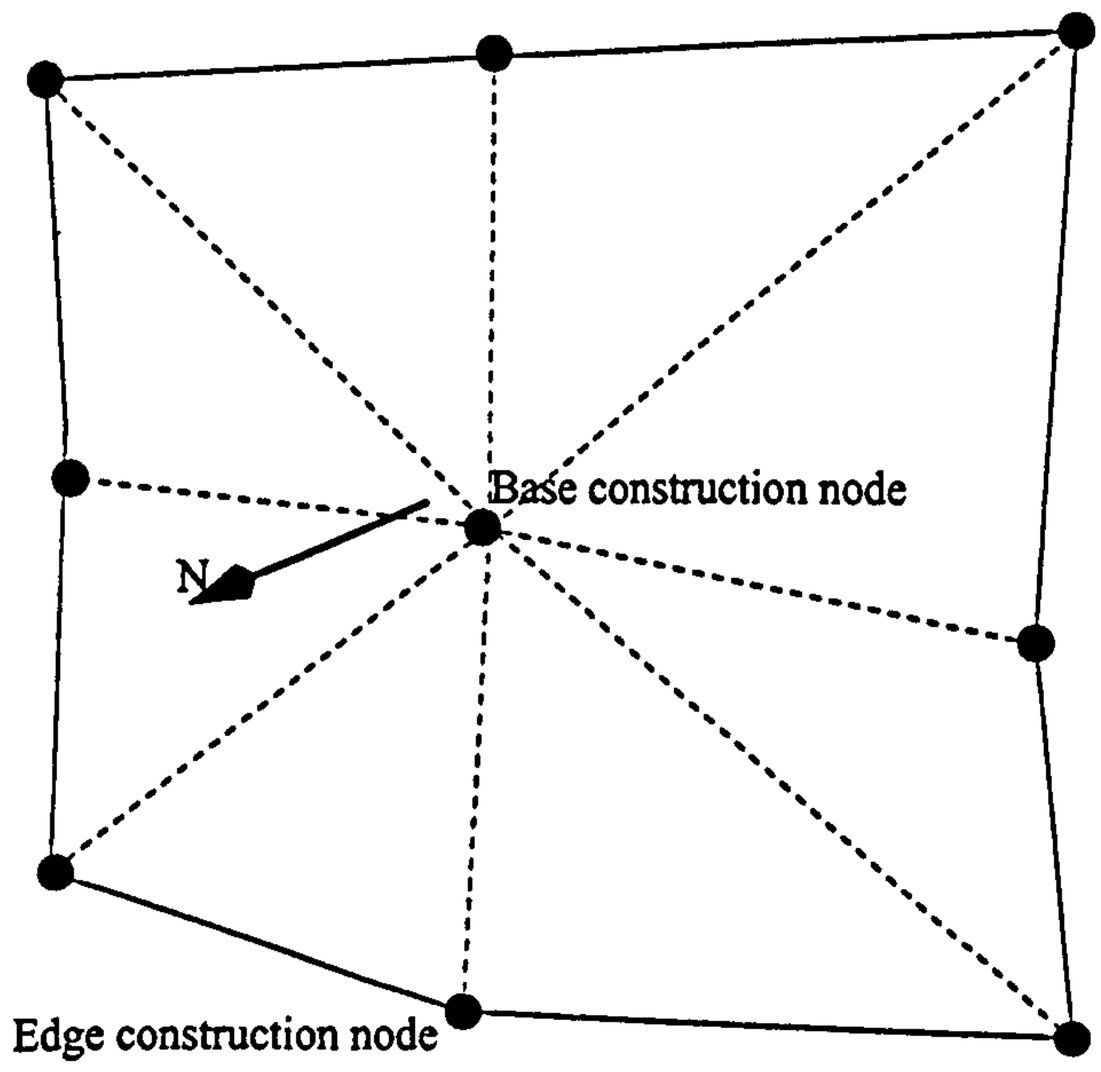


Figure 6.5: Node and face definition

There is a damping factor in Algorithm 6.5 because the movement of the base construction node was found to be too large, and the whole algorithm non linear, when the full deviation distance was used. A value of 0.1 has been used for the damping factor in all the cases presented.

It can be observed that the relevant node movement to make one face planar may not be the correct movement to make a co-joining face also planar. However, provided the planes are not the same, they will cross and with the introduction of third plane a unique point that lies on all three planes can be found. In theory more than three plane can have a unique co-located point, but it is not a mathematical certainty. Thus when moving around the faces attached to a given control volume a variation of Algorithm 6.6 is used. Due to the developmental nature of the work, and in aid of robustness, only the combination of two faces at any one given time are going to be considered. Algorithm 6.7 details the manner in which the node adjoining two faces is moved. For nodes attached to three or more faces an iterative process is used where the second face becomes the first and the next face becomes the second, until all the faces are considered. While this process is iterative, only one cycle through the list of construction nodes has been implemented, due to the computational time requirements of the process (see Table 6.6).

Algorithm : 6.7 (Multiple face deviation from planarity reduction)

- DEFINE THE BASE NODE POSITION AS C
- DEFINE ORIGINAL NODE POSITION, P, AS P^0
- DEFINE 1^{st} FACE NORMAL, $\vec{n_F^1}$
- $\Delta^1 = C - P^0$
- $\delta^1 = \Delta^1 \cdot \vec{n_F^1}$
- $P^1 = P^0 + \delta^1 \vec{n_F^1}$
- DEFINE 2^{nd} FACE NORMAL, $\vec{n_F^2}$
- $\Delta^2 = C - P^1$
- $\delta^2 = \Delta^2 \cdot \vec{n_F^2}$
- $P^2 = P^1 + \delta^2 \vec{n_F^2}$
- NEW NODE POSITION, $P_{new} = P^2$

6.6 Validation and verification of adaptation algorithms

Using the algorithms of adaptation described in Section 6.3 and 6.4 the number of control volumes can be increased or decreased as the underlying mesh quality dictates. The use of the data structure, in particular the binary tree structure defined in Chapter 2.5.3 and shown in Figure 2.16, aids this process. A process of inheritance of flow data properties is used during fission of control volumes and volume weighted averaging of properties is used for fusion.

Conservation of mass, momentum and energy are maintained due to the manner of calculating the flux vectors on a face based algorithm. If two hexahedra are adjacent and one split, then the second volume will have two interfaces on the joining plane, one face joining it directly to both of the children of the first hexahedra.

6.6.1 Arbitrary splitting of volumes

Algorithms 6.2 and 6.3 have been used with success to split and merge volumes of a variety of shapes in an arbitrary plane.

As a test example the six faced control volume presented in Figure 6.6 was created within a larger cubic mesh. The different coloured faces represent different boundary types, and the faces are presented in an 'exploded' view format to indicate the sub faces and positions of the construction nodes.

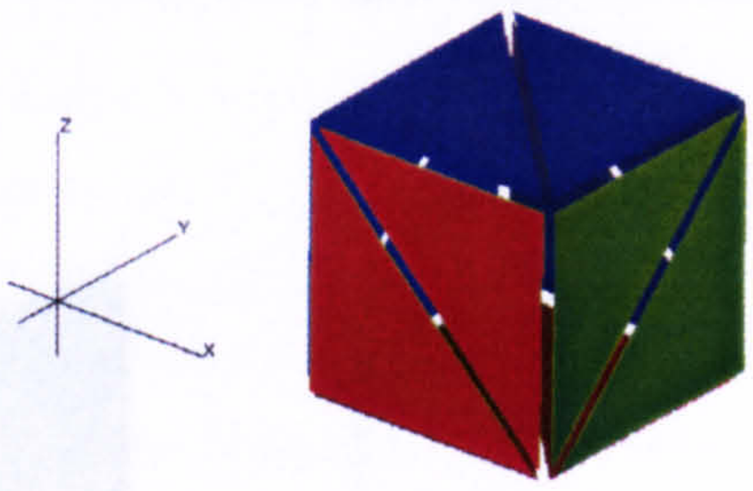


Figure 6.6: Typical hexahedral control volume. Different coloured faces represent different boundary types.

This volume was then split in a number of manners to validate the algorithms, three results of which are presented in Figures 6.7 - 6.9. The only variation in the input to the splitting algorithm was the definition of the cutting plane, defined by a (normal) vector.

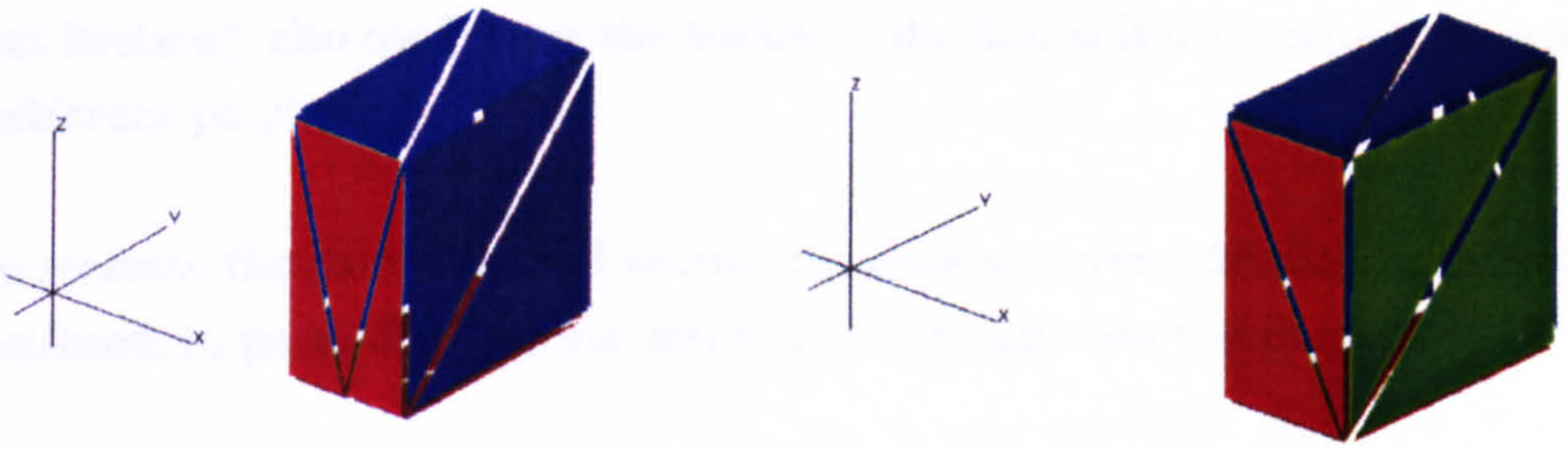


Figure 6.7: Children of volume with splitting plane normal (1, 0, 0)

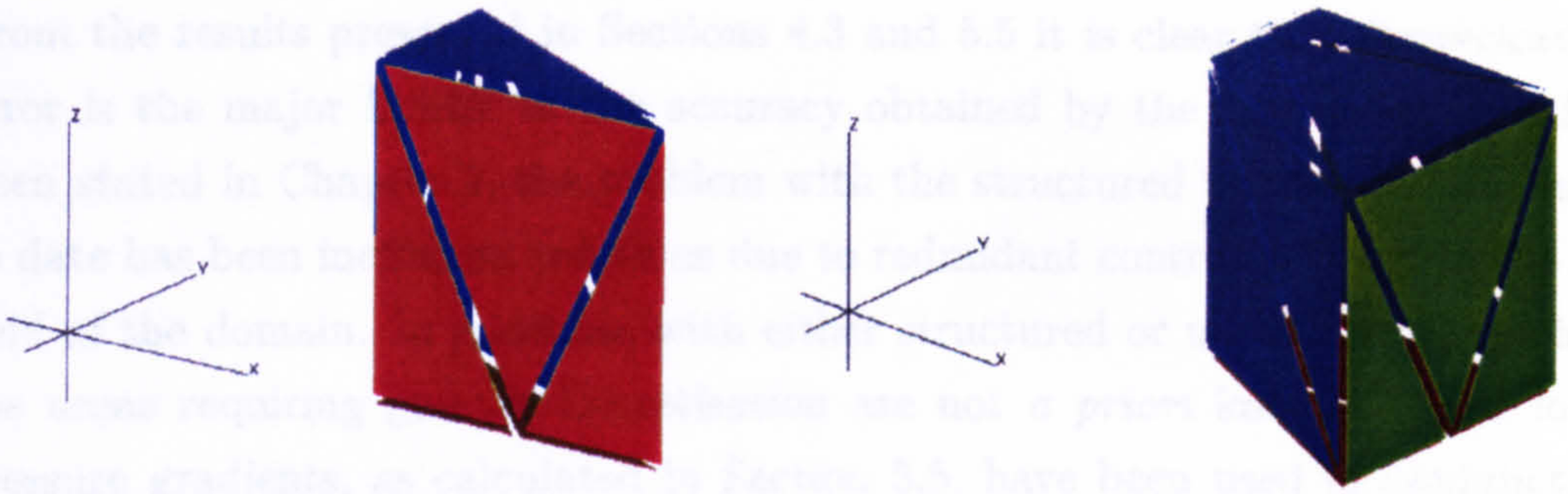


Figure 6.8: Children of volume with splitting plane normal(0.577, 0.816, 0)

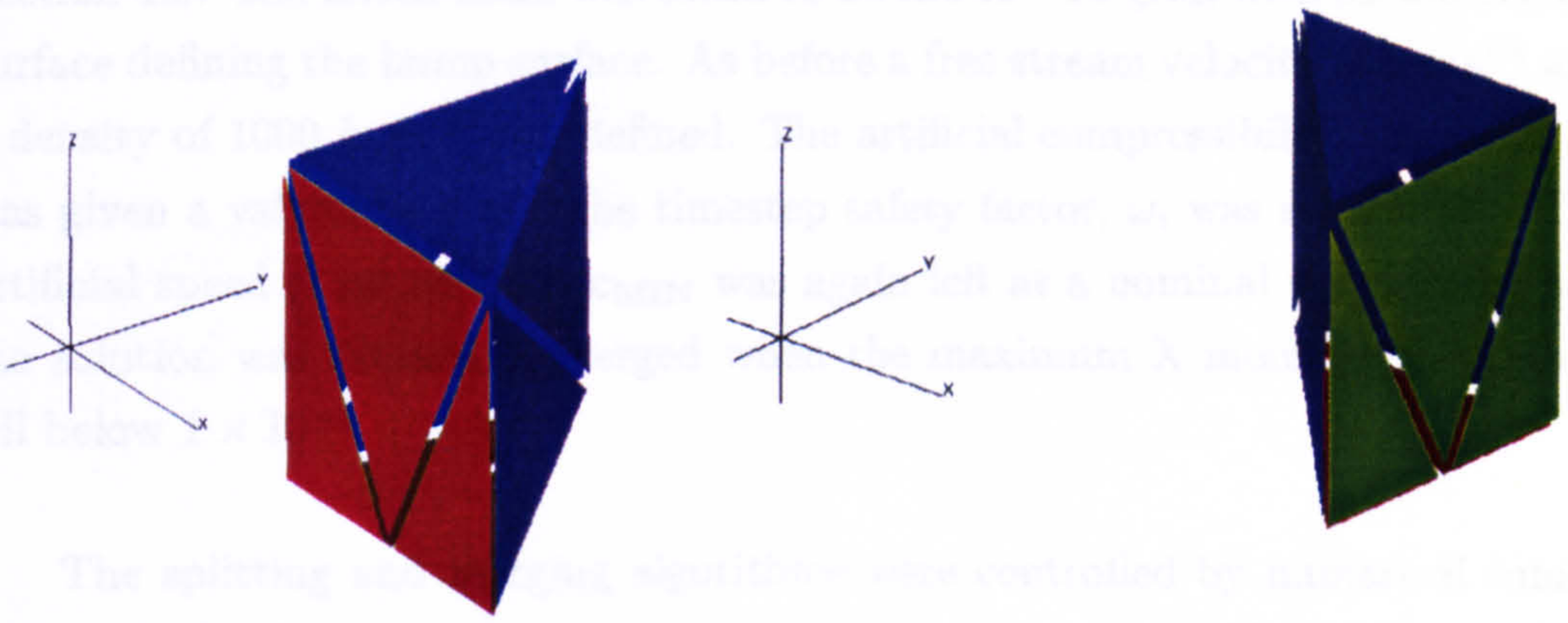


Figure 6.9: Children of volume with splitting plane normal(0.707, 0.707, 0)

It can be observed that a variety of polyhedra are created by these splits, thus confirming the arbitrary nature of the splitting algorithm and, when these control volumes were numerically tested within the flow solver (as presented in the next Section), also confirming the ability of the flow solver algorithm to cope with arbitrary polyhedrals.

To recreate the parent control volume the data structure detailed in Section 2.5 is utilised, in particular the tree structure, to re activate the parent.

Further sub divisions of the control volumes, in particular those detailed in Figure 6.7, have been carried out to a level of twelve successive sub divisions without any corruption of the mesh. It is thus felt that Algorithm 6.2 is suitably robust for use with complex and arbitrarily shaped domains.

10% arc hump mesh adaption

From the results presented in Sections 4.3 and 5.5 it is clear that discretisation error is the major limiter in the accuracy obtained by the flow solver. As has been stated in Chapter 2, the problem with the structured format of mesh used to date has been inefficient solutions due to redundant control volumes in the far field of the domain. In addition, with either structured or unstructured meshes, the areas requiring greater discretisation are not *a priori* known. Thus local pressure gradients, as calculated in Section 5.5, have been used in conjunction with Algorithms 6.2 and 6.3 to increase solution accuracy while maintaining as high a solution efficiency as possible.

To test this process it has been applied to the 10% arc hump, as defined in Section 4.3. The initial mesh was taken to be the 58×15 grid, with an underlying surface defining the hump surface. As before a free stream velocity of 5 ms^{-1} and a density of 1000 kgm^{-3} was defined. The artificial compressibility constant, $\mathcal{K}$, was given a value of 1.5 and the timestep safety factor, ω, was set to 0.65. The artificial speed of sound limit $\mathbf{c_{MIN}}$ was again left at a nominal value of 0.1 and the solution was deemed converged when the maximum X momentum residual fell below 1×10^{-8}.

The splitting and merging algorithms were controlled by numerical limits. The control volumes with a pressure gradient across it outside 2.365 standard deviations of the mean were split or merged depending on which side of the mean they lie.

The adaption process was executed every 600 iterations for a total of 2400 iterations (i.e. 4 levels of adaptation).Figure 6.10 presents the 'new' construction volumes after all 4 levels of adaption. The total number of control volumes increased from 2610 to 2881 between the original mesh and the final adapted one. In addition to these volumes a number of the original volumes will be greatly reduced in size. As an example, Figure 6.11 shows a control volume near the leading edge of the hump in the original mesh and in the final, adapted mesh. As well as being reduced in size, the volume has been altered in shape considerably; this is due to the manner of splitting the control volumes by a planar surface. The other six 'children' of the original control volume have been hidden to clarify the image.

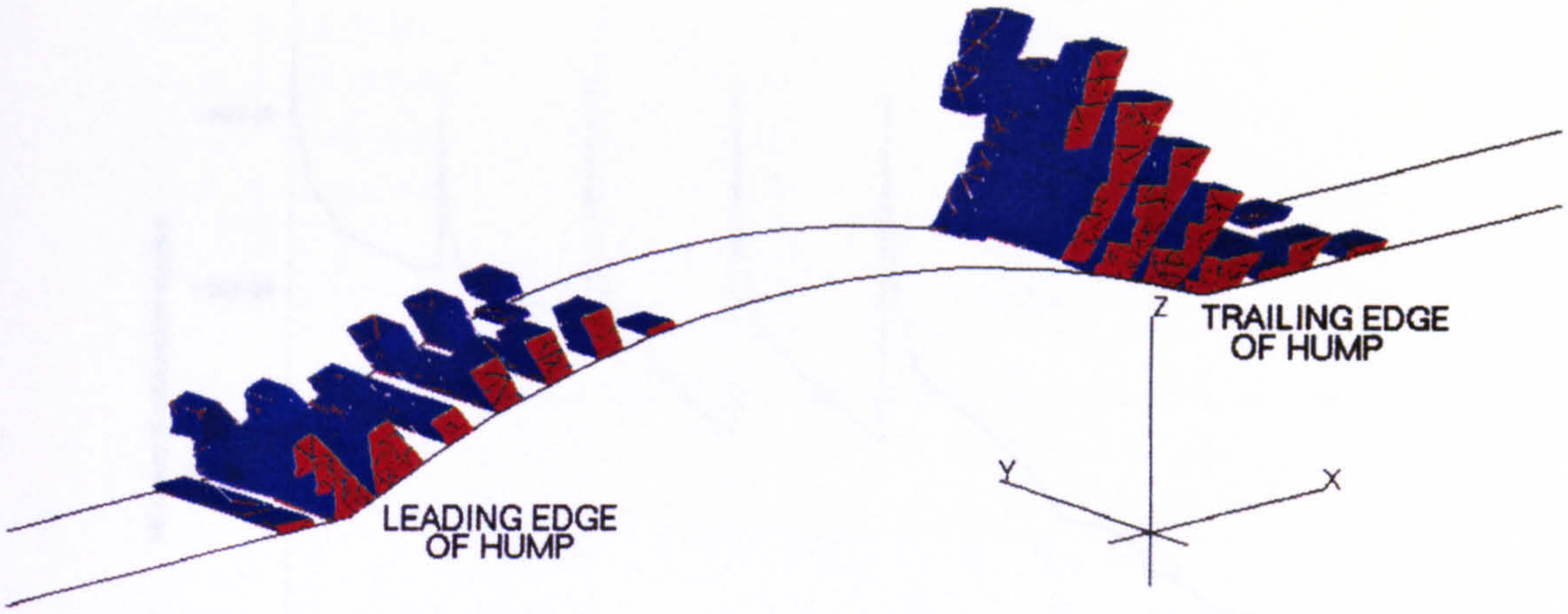


Figure 6.10: Children control volumes with new IDs

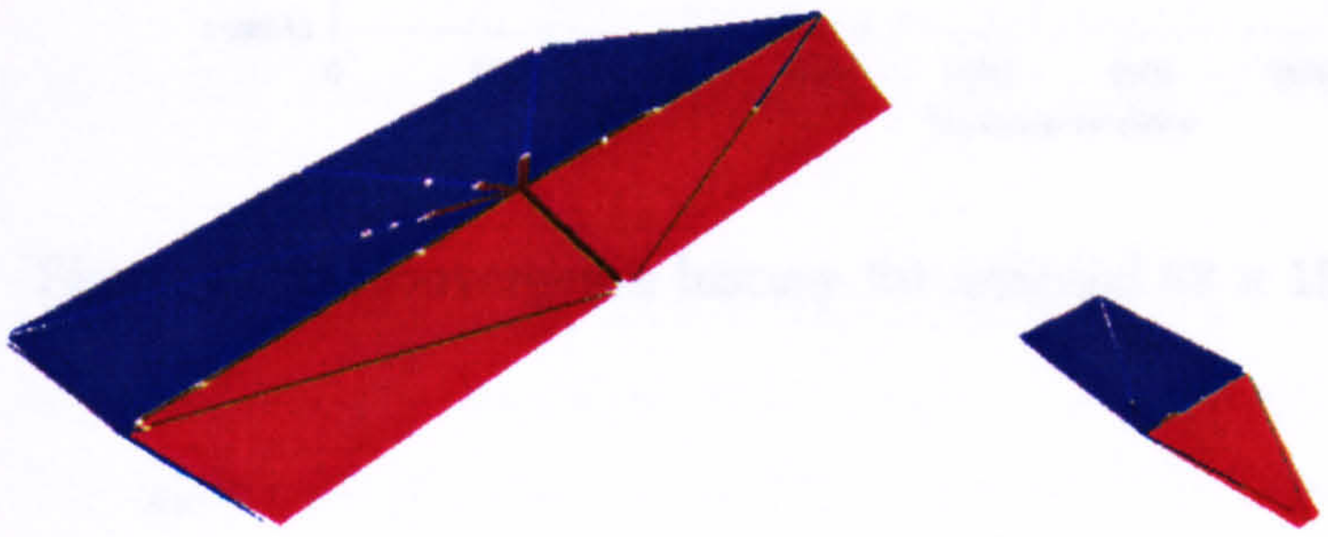


Figure 6.11: Parent control volume and 4th generation child

The convergence history of the solution is shown in Figure 6.12 while Table 6.2 presents the CPU time per iteration on a 500 MHz Pentium III processor, with a Linux operating system fully optimised GNU compiled C++ code, for each variation of the mesh. The total number of iterations at each level, and the resultant total CPU for the solution are also listed in Table 6.2. Table 6.3 presents the forces generated by the flow over the hump, in comparison to the finest mesh modelled in Section 4.3 (268×67), and the initial mesh (58×18). The pressure coefficient over the hump surface for the adapted mesh is shown in Figure 6.13, presented with the result for the 58×18 mesh and the trends of the maxima and minima, as shown in Figure 4.10.

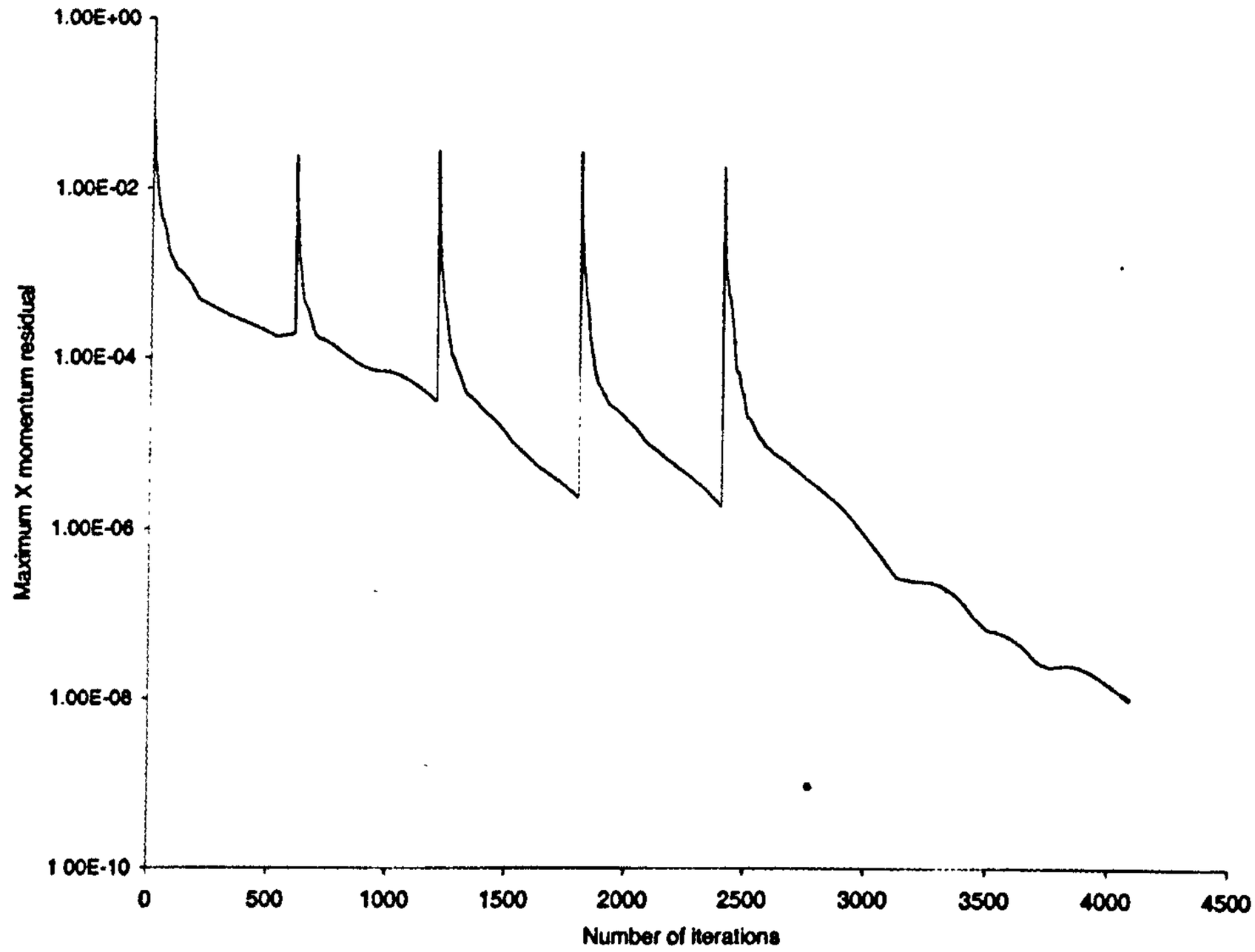


Figure 6.12: Convergence history for adapted 58×15 mesh on 10% arc hump

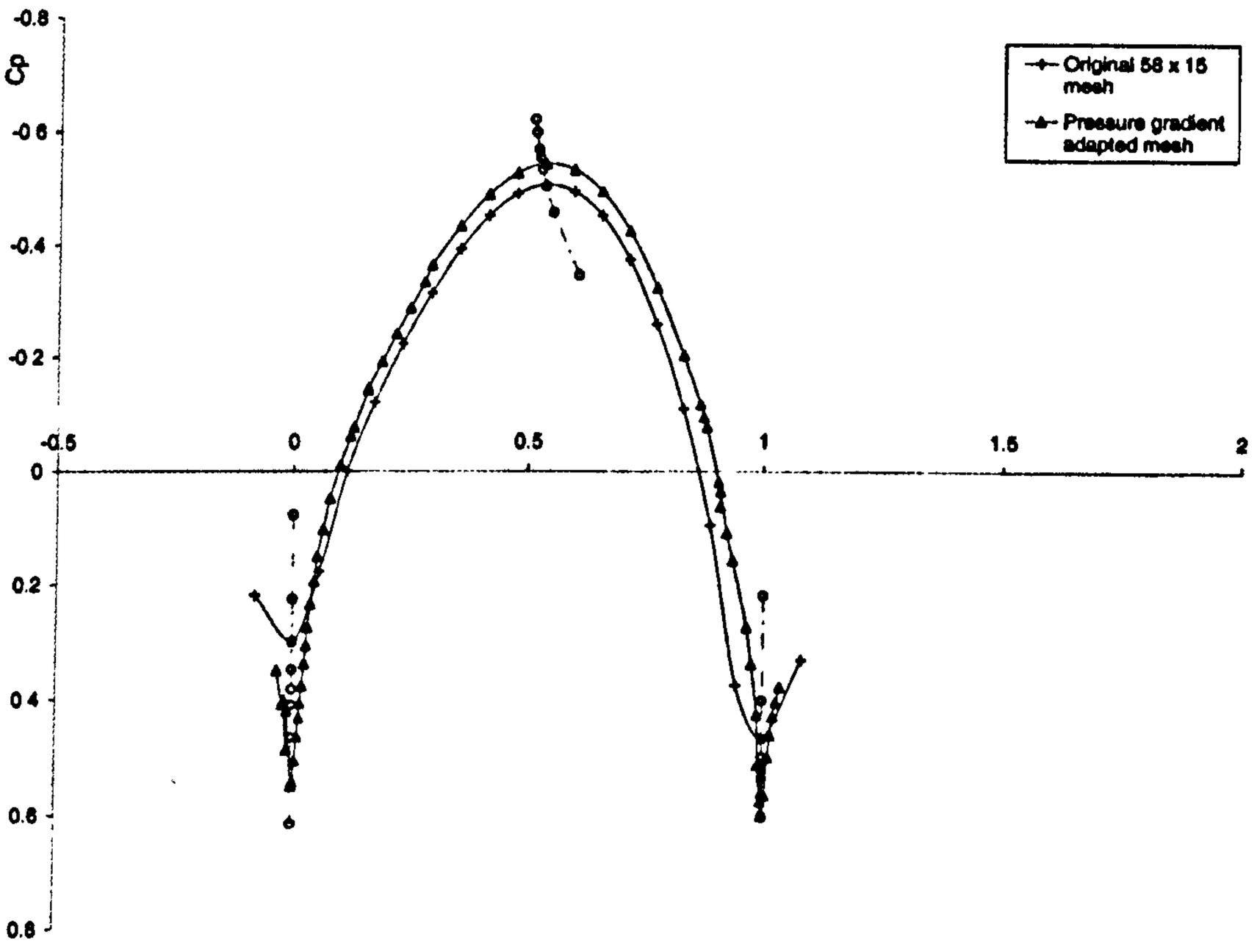


Figure 6.13: Pressure coefficient on 10% arc hump for original and adapted mesh

Mesh	$\frac{H_1}{H_i}$	CPU time per iteration (s)	Number of Iterations	Total CPU time for convergence
58×15	4.62	1.48	3231	4781.9
Initial mesh	4.62	1.48	600	
With 1 adaption		1.43	600	
With 2 adaptions		1.46	600	
With 3 adaptions		1.50	600	
With 4 adaptions		1.60	1692	6233.2
178×54	1.51	14.51	10799	156693
268×67	1.0	31.4	19165	601781

Table 6.2: CPU time usage for adapted mesh over 10% arc hump

Mesh number	Force in X direction	Force in Z direction
hump 3	0.007374	-0.150628
adapted hump 3	0.00706	-0.2533
hump 8	0.005571	-0.26821

Table 6.3: Forces imposed upon 10% arc hump for adapted and original meshes

The results of Table 6.2 show that the overall CPU time for the mesh is noticeably more than that of the original mesh (30% difference in CPU time), but when the increase in accuracy of the forces and pressure field (Table 6.3 and Figure 6.13 respectively) is noted then it can be justifiably be stated that the efficiency of the solver is vastly increased. The discretisation error of the solution with the adapted mesh is equivalent to that of a structured mesh with $h_i/h_1 \approx 1.25$, when comparing the forces generated with those presented in Figure 5.5. Table 6.2 shows that a structured mesh of that density would take in the region of 425000 CPU seconds. There is therefore an efficiency increase in the region of 60 orders of magnitude.

6.6.2 Geometrical entity convection and manipulation

The algorithms detailed in Section 6.5 were used in conjunction with the geometrical grid quality parameters as defined in Chapter 5 to quantifiably measure, define and attempt to improve the grid quality of two of the meshes tested in Section 5.5. The relative success of these algorithms was taken to be the grid quality after adaptation and the flow solution accuracy of the adapted mesh.

10% arc hump

The first test case was the 58×15 mesh over the 10% arc hump, with the form of distortion being misalignment of face normals with the node to node vector (a relative movement of 30% was chosen). This case was chosen because it gave a suitable poor mesh (thereby causing a stringent check of the algorithm) while not causing any contravention of the Global Conservation Laws.

The distorted mesh was used with the uniform flow detailed in Section 4.3, as with the flow models detailed in Section 5.5.1. The difference in this model was that the grid quality assessment and geometrical adaption routines were switched on. At the beginning of the solution phase the geometrical grid quality was assessed, and those areas with an excessively low grid quality were adapted according to Algorithms 6.5 and 6.7, as well as the process detailed in Algorithm 6.4 being applied to all areas of the mesh. The planarity algorithms are only applied to specific areas because of their large (with respect to Algorithm 6.4) CPU requirements.

Grid quality is defined by Equation 5.12. The relative weightings of the geometric parameters ($\beta_1 - \beta_4$ in Equation 5.12) are presented in Table 6.4. These values were taken from the trends and average values of the results obtained in Section 5.5. As with the flow solution measures, the mean and standard deviation of the grid quality were calculated, and those control volumes more than 2.365 standard deviations greater than the mean were flagged.

Geometrical parameter	Weighting
Aspect ratio	1.5
Skew	8.0
Misalignment of normal	0.75
Deviation from planarity	0.75

Table 6.4: Weighting of geometrical parameters for quality analysis of meshes

The quality of the three different meshes (before distortion, after distortion and after adaption of the distorted mesh) is shown in Figures 6.14 - 6.16. As noted in Section 5.5, a grid quality value of zero would be ideal, and as the number increases the geometrical mesh quality degrades. Table 6.5 presents the statistical data for the geometrical quality of the three meshes.

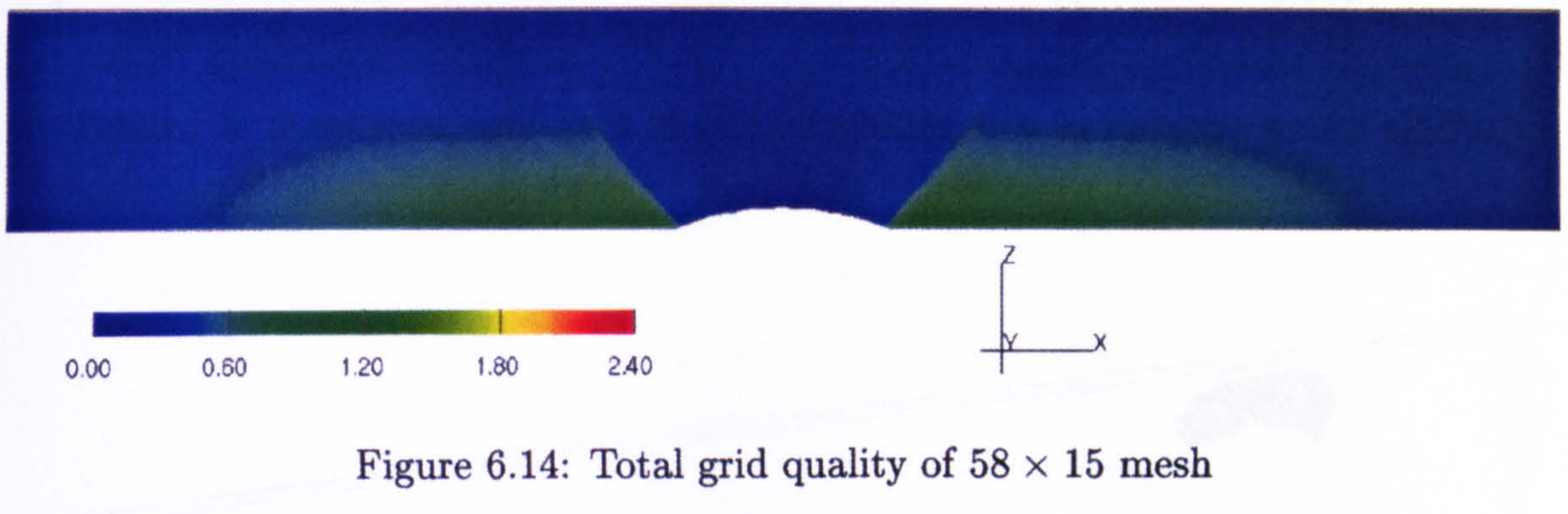


Figure 6.14: Total grid quality of 58×15 mesh

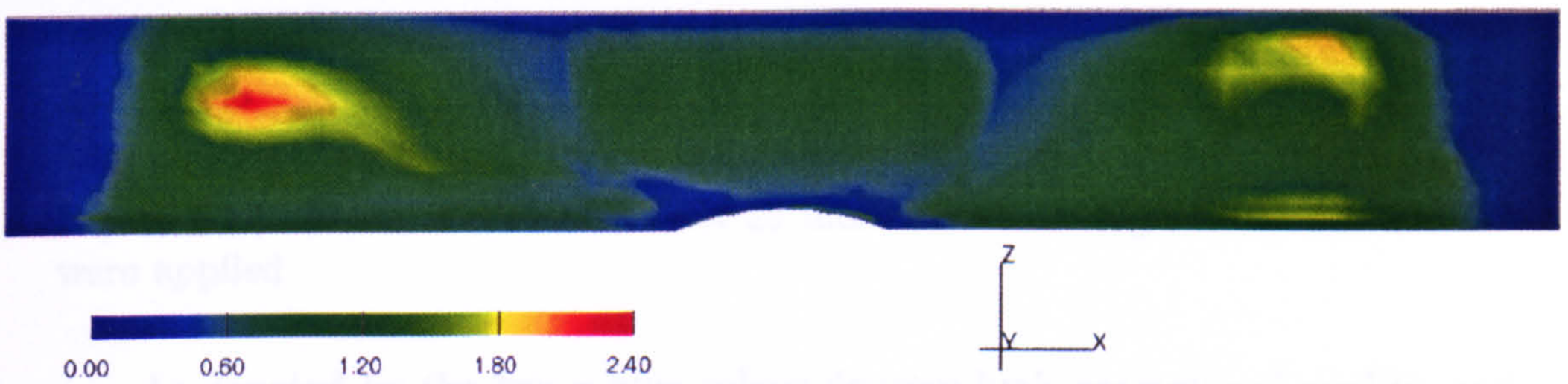


Figure 6.15: Total grid quality of 58×15 mesh with mesh distortion

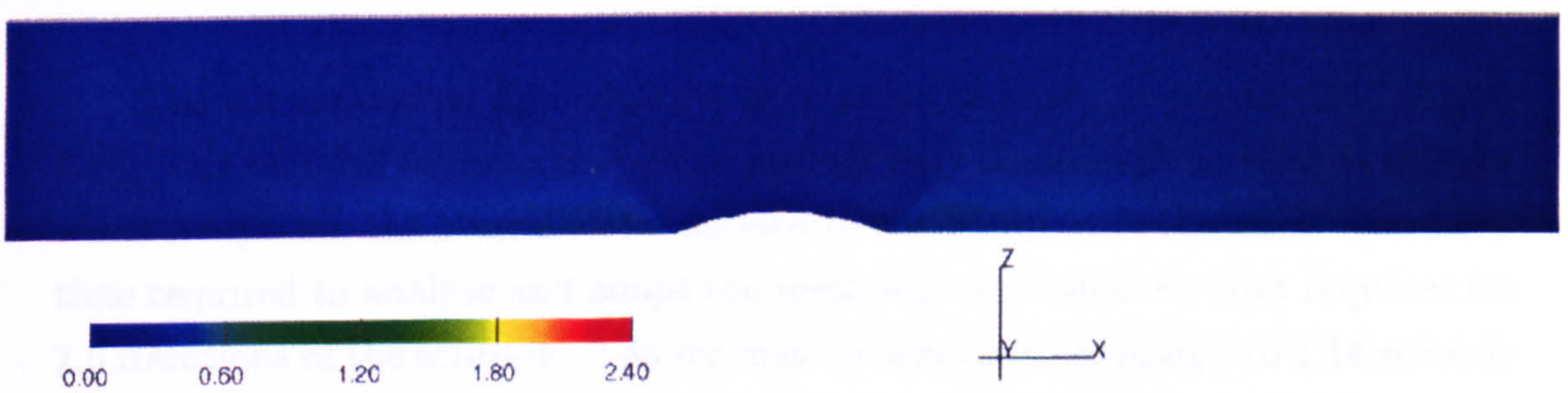


Figure 6.16: Total grid quality of distorted 58×15 mesh after adaption

Mesh	Mean quality value	Maximum quality value
Original 58 × 15 mesh	0.316	0.546
Distorted 58 × 15 mesh	0.796	2.381
Adapted distorted 58 × 15 mesh	0.257	0.494

Table 6.5: Statistical definition of geometrical quality in 10% arc hump meshes

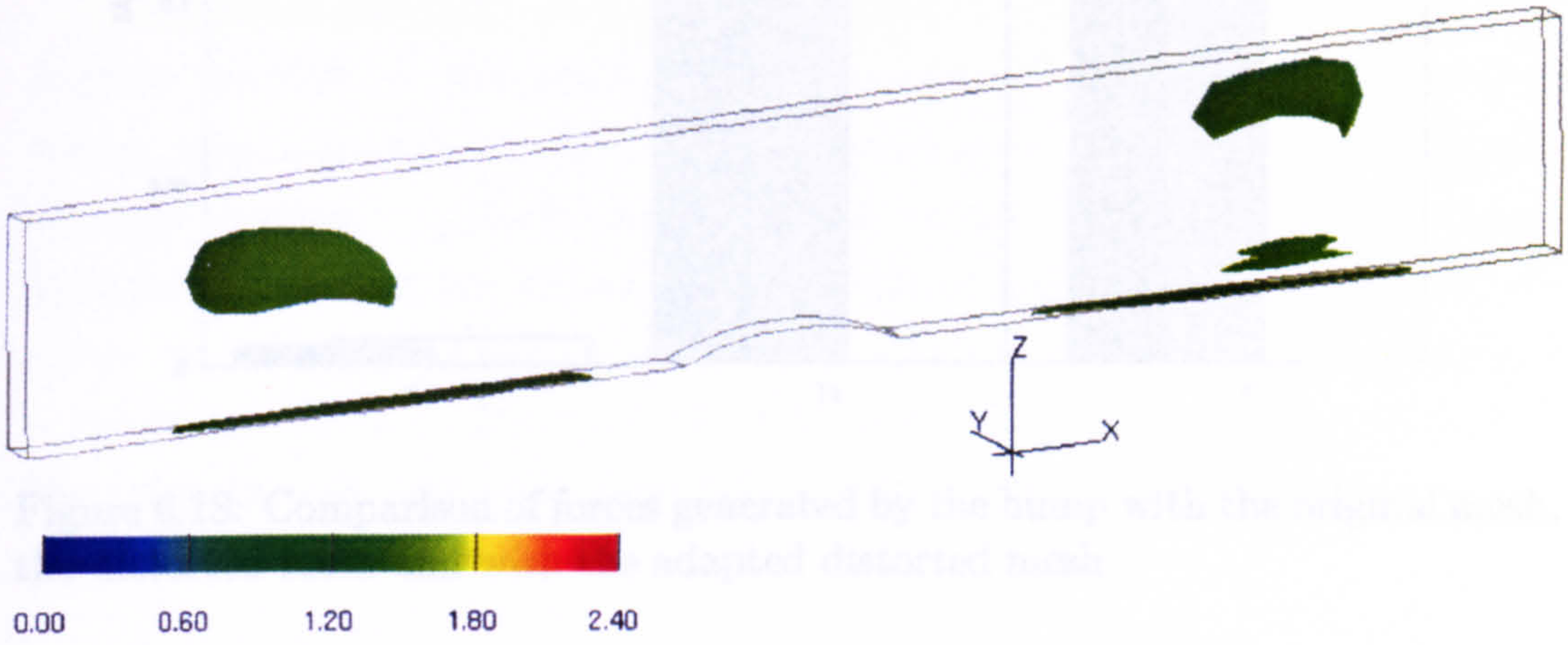


Figure 6.17: Areas of distorted 58 × 15 mesh to which Algorithms 6.5 and 6.7 were applied

As denoted by the key a blue colour denotes high geometrical quality, and red areas of poor quality. The lowering of grid quality by the distortions and the re-capture of high geometrical quality after application of the adaption algorithms can clearly be seen in Figures 6.15 and 6.16 respectively. While the process detailed in Algorithm 6.4 was applied throughout the domain, the planarity algorithms (Algorithms 6.5 and 6.7) were applied only in the areas denoted in Figure 6.17.

The corresponding improvement in solution accuracy is presented in Figure 6.18. The removal of distortion error, leaving only discretisation error is evident when comparing the overall forces applied to the hump from the fluid. The CPU time required to analyse and adapt the mesh was equivalent to that required for 2.5 iterations of the solution: 2.48 seconds for adaption compared to 1.14 seconds for one solution iteration on a Pentium III 650 MHz processor using the Linux operating system. When it is considered that the converged solution takes in the region of 3000 iterations to converge for the 58 × 15 mesh with good geometrical quality (see Figure 4.7), and much longer for one with poor geometrical quality,

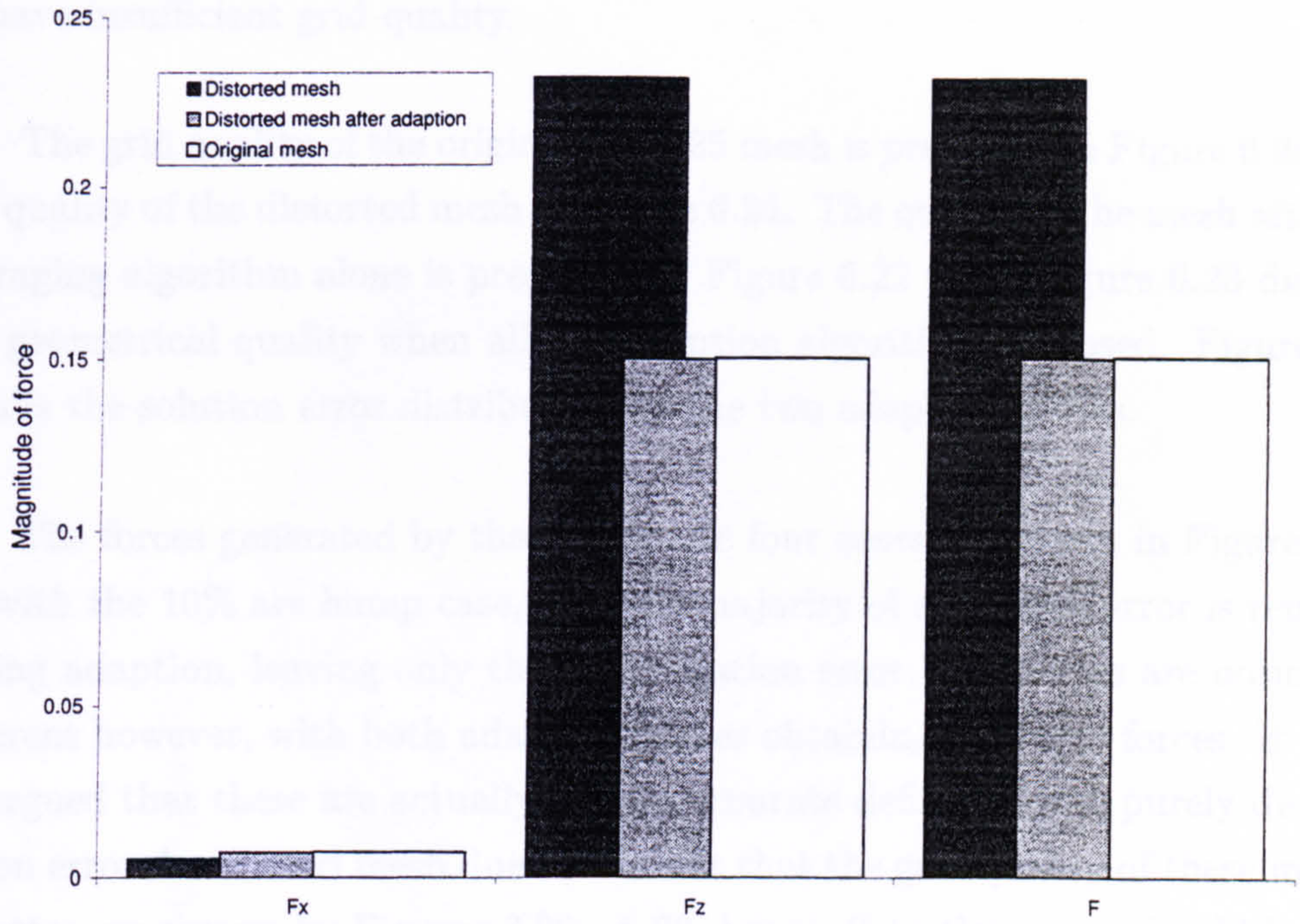


Figure 6.18: Comparison of forces generated by the hump with the original mesh, the distorted mesh and with the adapted distorted mesh

the computational cost of this process can be deemed to be very low.

NACA 0012 foil

The second case study of the geometrical adaption routines was the 343×25 mesh around a NACA 0012 foil at 8 degrees angle of attack, again with misalignment of face normals with the node to node vector at a factor of 30%. To maintain comparability the same flow parameters and solution method were used as in Section 5.5.2, but with the geometrical grid quality assessment and adaption modules switched on. The same geometrical parameter weightings used for the 10% arc hump case (listed in Table 6.4) were applied in this case, and again those control volumes with a grid quality greater than 2.365 standard deviations of the mean were noted. A variation used for this case study was the use of the averaging algorithm (Algorithm 6.4) by itself, and then with the extra planarity improvement algorithms for a different solution case to test the relative effectiveness of the two forms of geometrical grid quality enhancement. Again the construction node position averaging algorithm was applied throughout the domain, while the planarity algorithms were only applied to those control volumes that were deemed

to have insufficient grid quality.

The grid quality of the original 343×25 mesh is presented in Figure 6.20, and the quality of the distorted mesh in Figure 6.21. The quality of the mesh after the averaging algorithm alone is presented in Figure 6.22 while Figure 6.23 displays the geometrical quality when all the adaption algorithms are used. Figure 6.24 details the solution error distribution on the two adapted meshes.

The forces generated by the foil for the four cases are shown in Figure 6.19. As with the 10% arc hump case, the vast majority of distortion error is removed during adaption, leaving only the discretisation error. The forces are noticeably different however, with both adapted meshes obtaining the same forces. It could be argued that these are actually a more accurate definition of a purely discretisation error dominated mesh due to the fact that the grid quality of these meshes is better, as proven by Figures 6.20 - 6.23, hence distortion error should be less. The original mesh was never declared to have discretisation error alone.

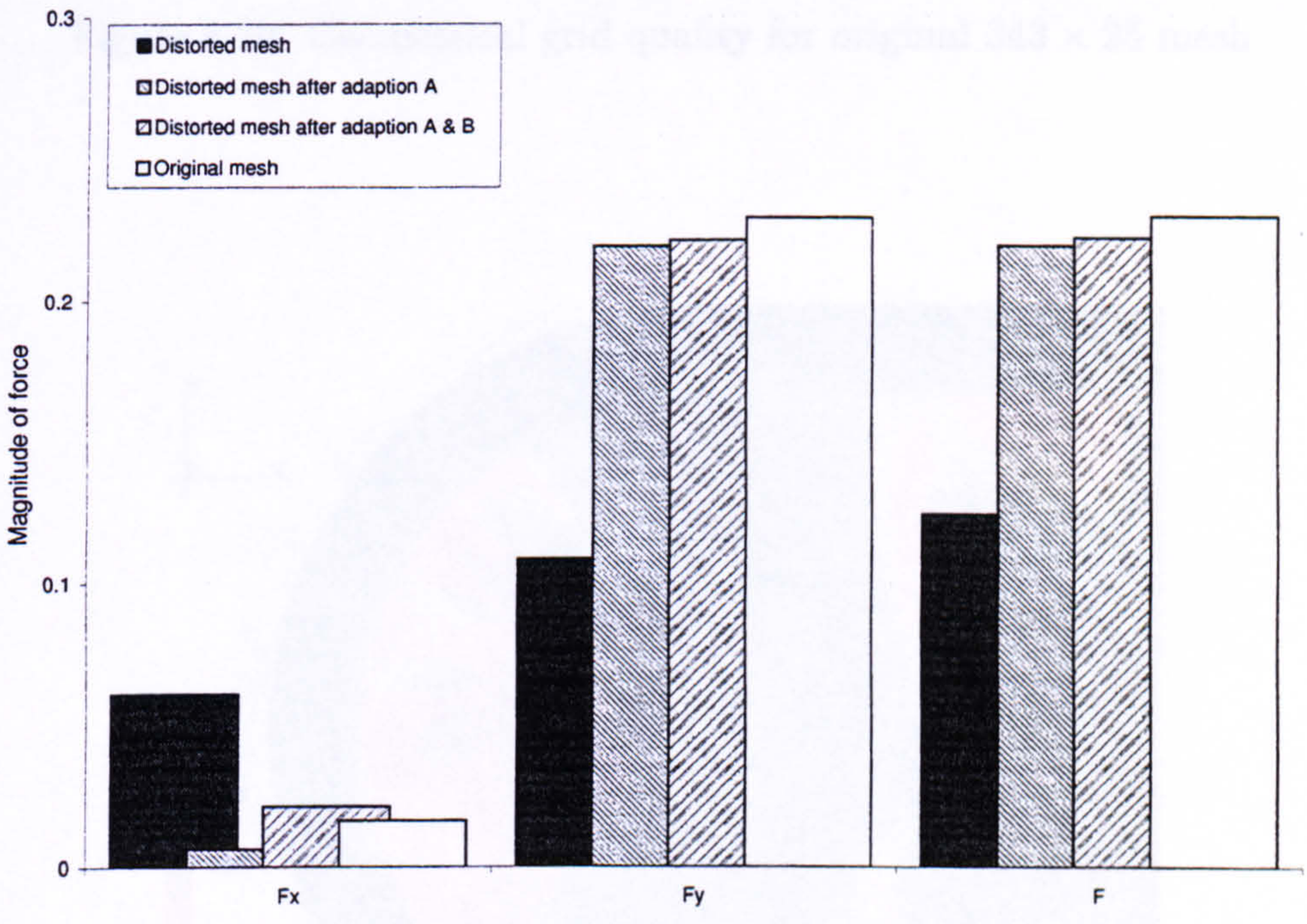


Figure 6.19: Comparison of forces generated by foil with the original mesh, the distorted mesh and the two adaptions of the distorted mesh.

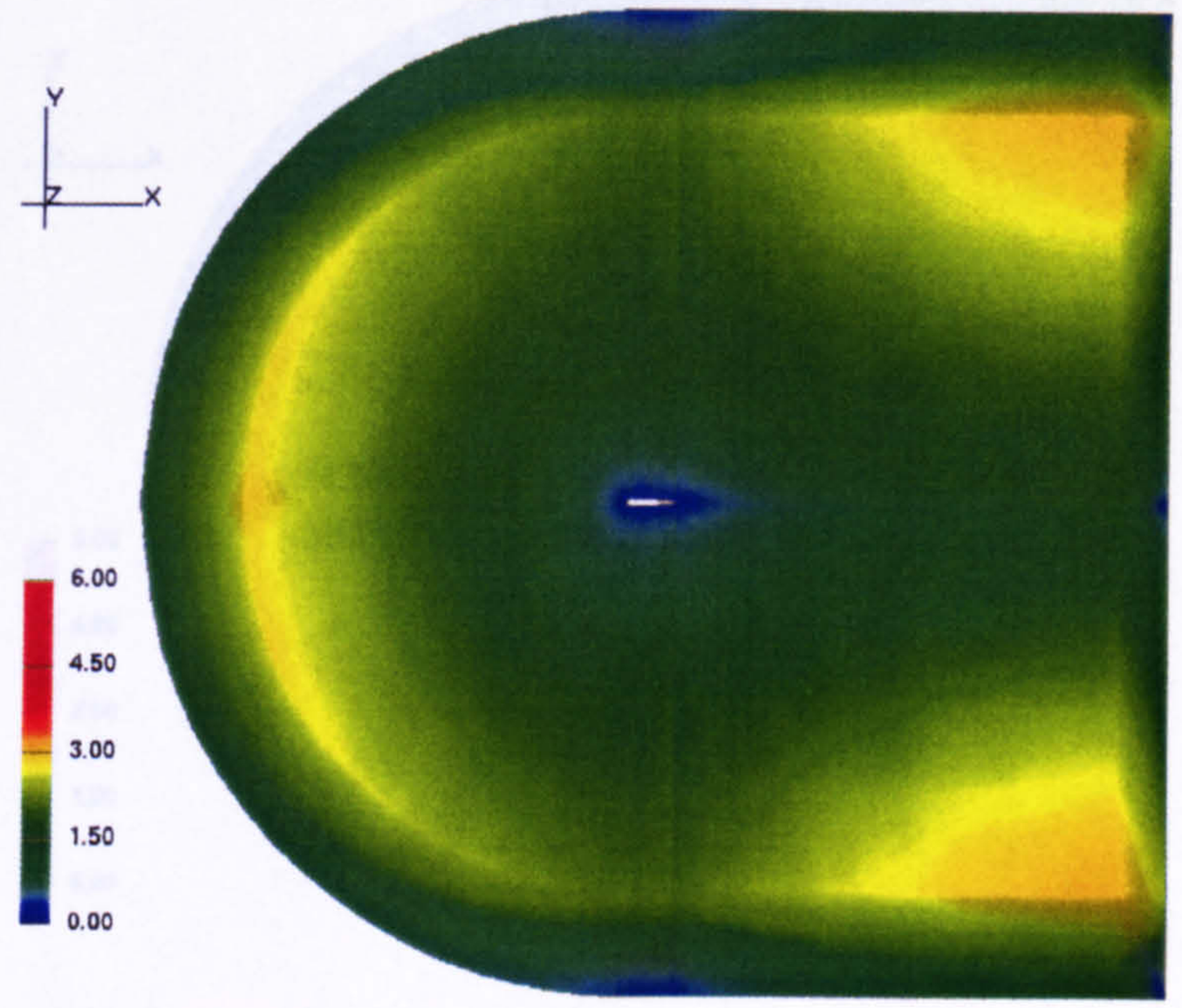


Figure 6.20: Geometrical grid quality for original 343 × 25 mesh

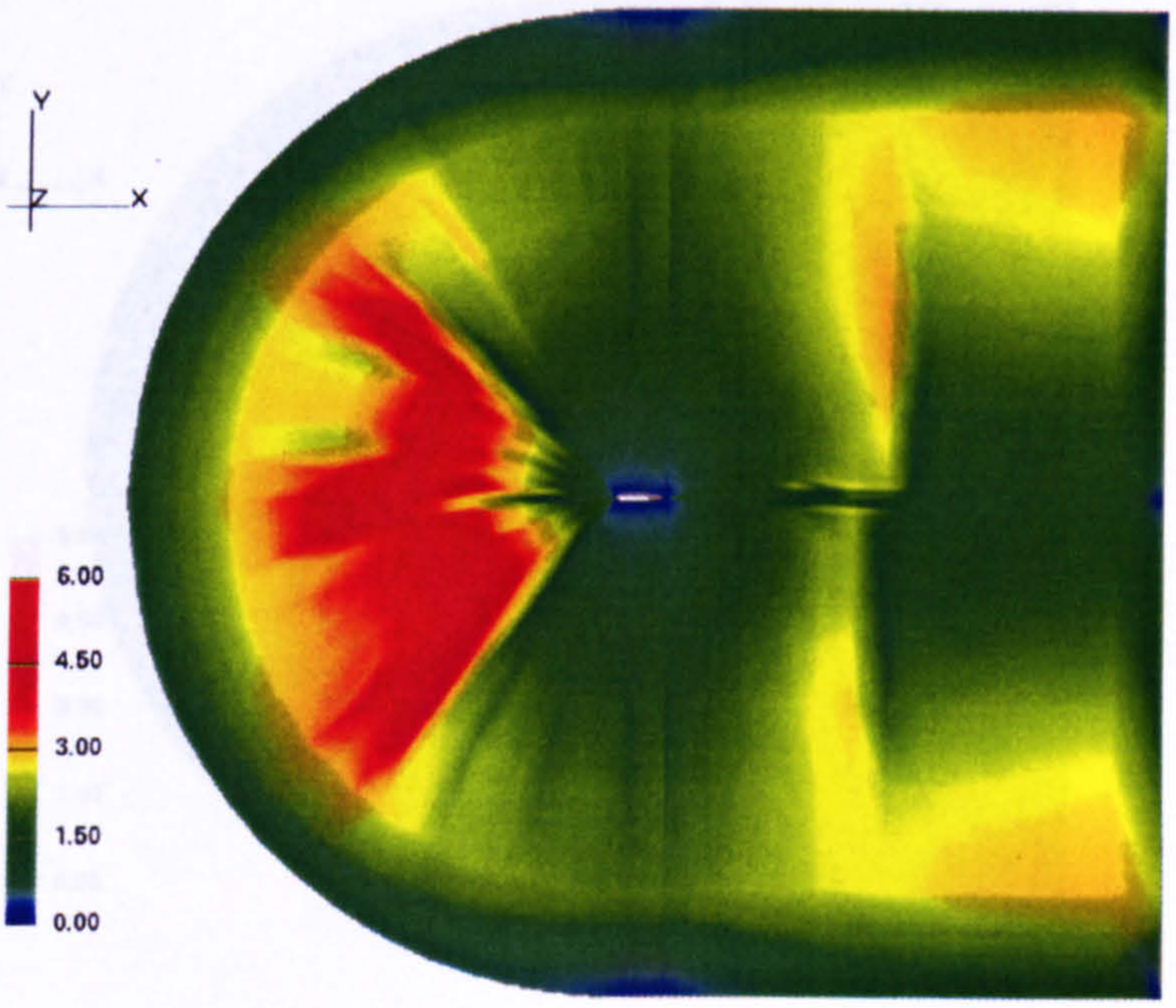


Figure 6.21: Geometrical grid quality for 343 × 25 mesh after distortion

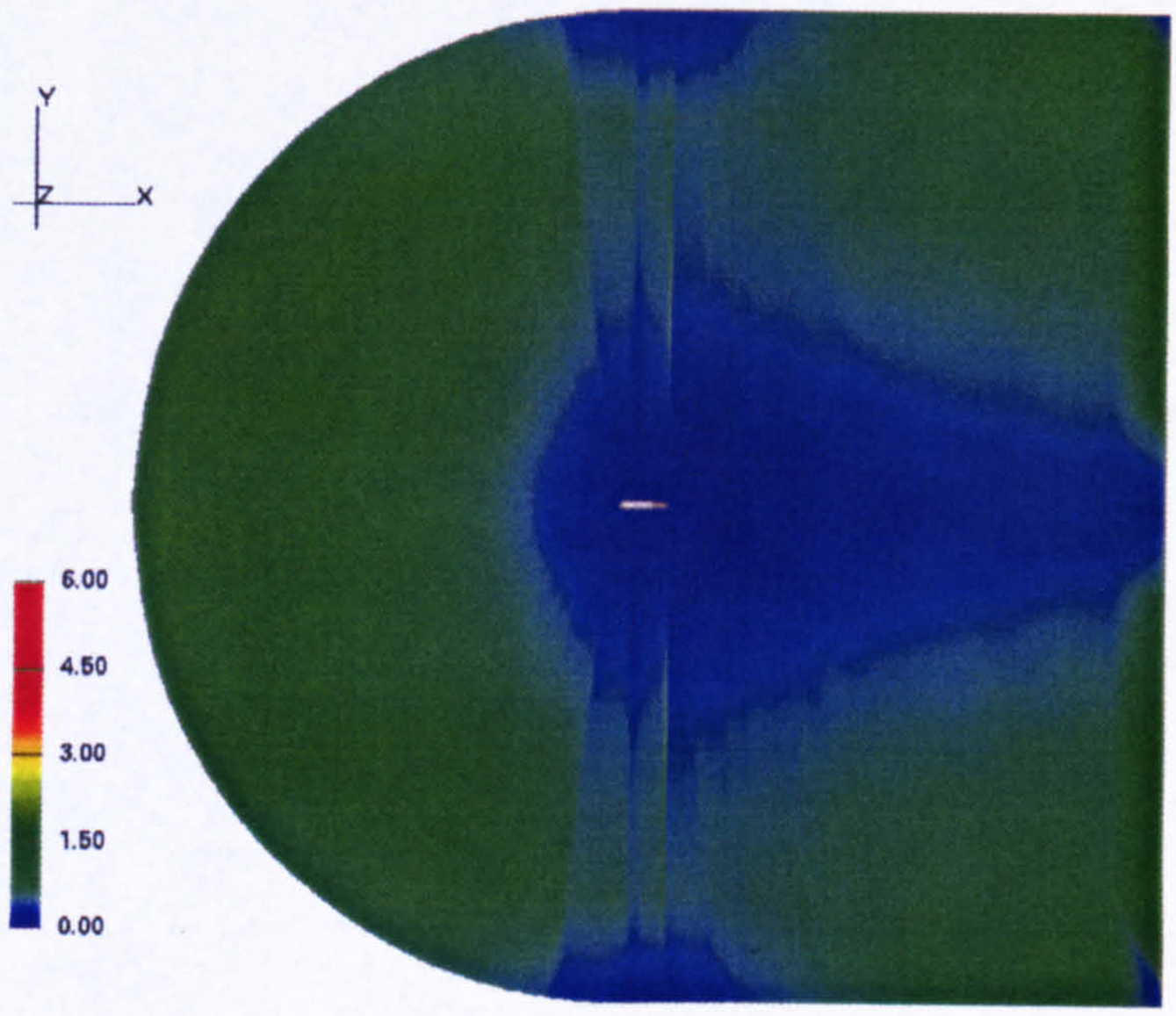


Figure 6.22: Geometrical grid quality for distorted mesh after adaption via Algorithm 6.4

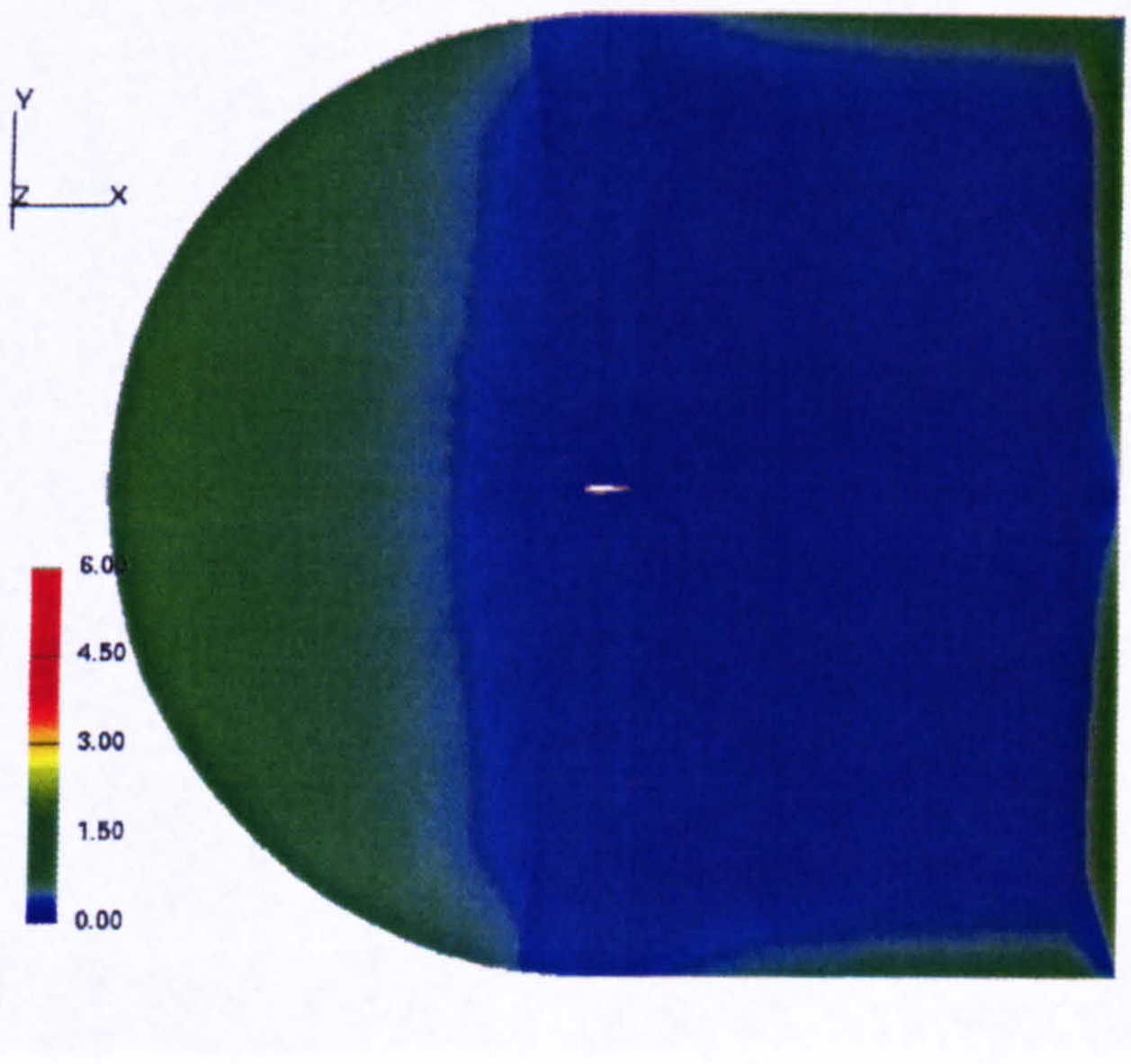


Figure 6.23: Geometrical grid quality for distorted mesh after adaption via Algorithms 6.4, 6.5 and 6.7

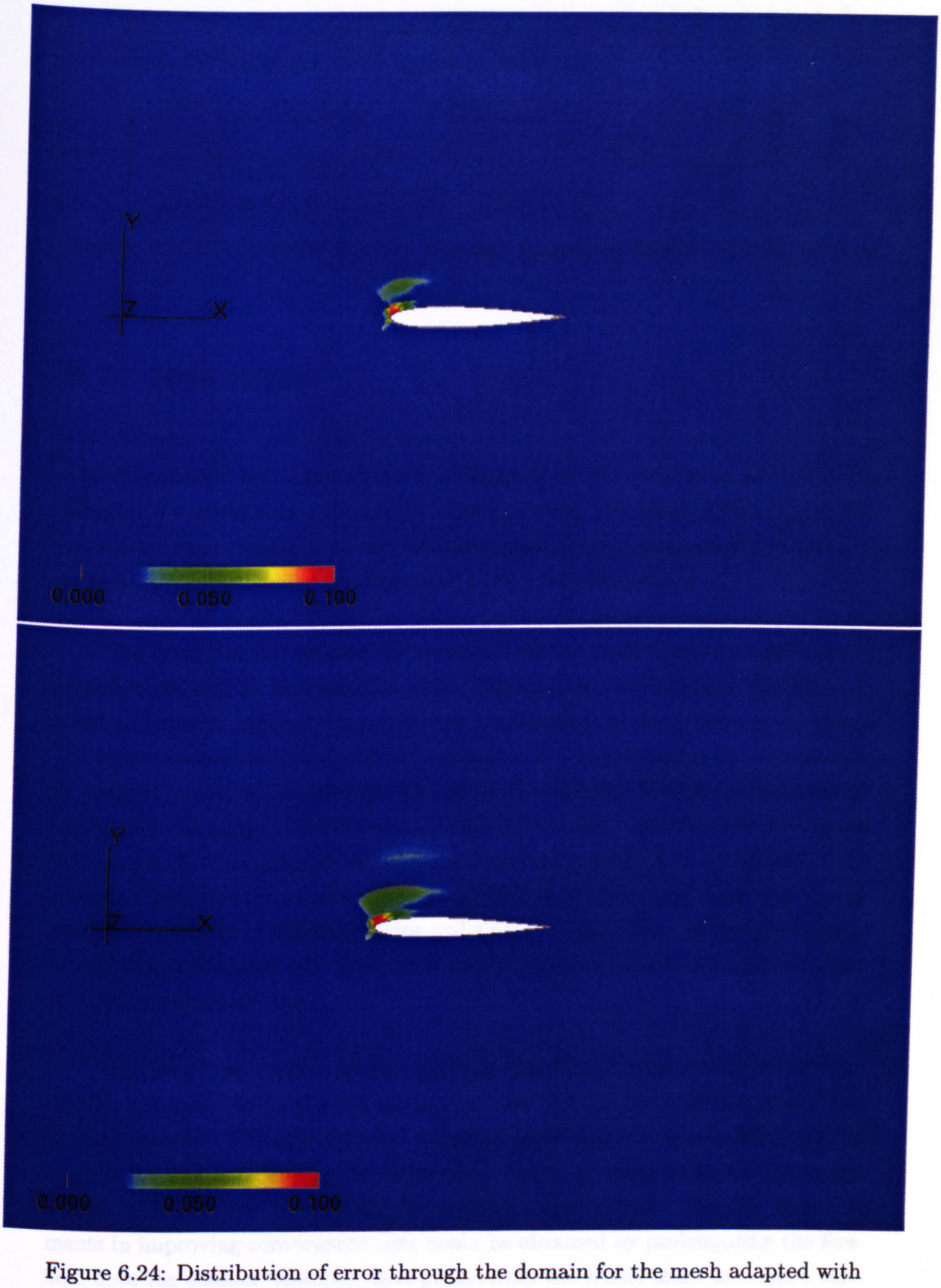


Figure 6.24: Distribution of error through the domain for the mesh adapted with Algorithm 6.4, and with all algorithms respectively.

The CPU time required to analysis and adapt the mesh for the two different combinations of algorithms are shown in Table 6.6.

Type of adaption	CPU time for analysis (s)	CPU time per solution iteration (s)
Algorithm 6.4	102	10.8
Algorithms 6.4, 6.5 and 6.7	262	10.8

Table 6.6: CPU times for different adaption techniques for NACA 0012 foil mesh

6.7 Summary

Algorithms have been created and implemented within the flow code detailed in Chapter 4 to manipulate the geometrical quality of the mesh. These algorithms have been incorporated with the measurements of quality detailed in Chapter 5 to produce an automated quality control and adaption process.

The results of this process are presented for the manipulation of geometrical quality alone and for flow solution error. Figures 6.18 and 6.19 show the improvement in solution accuracy possible with the removal of distortion error. While the cases investigated were relatively extreme, it is worth noting the comparison of the grid quality of the original 343×25 mesh about the NACA0012 foil and the quality for the adapted mesh (with all algorithms used), as presented in Figures 6.20 and 6.23. The adapted solution has improved quality in the majority of the domain. This therefore leads to the conclusion that the initial mesh had an appreciable quantity of distortion error and that the application of the geometrical adaption algorithms is expedient when considering the relative time for adaption to total convergence time.

The use of the control volume splitting algorithm has proven similarly successful, reducing the local error throughout the domain by reduction of the discretisation error. The case study investigated showed an increase in efficiency for a given level of accuracy in the order of 60 orders of magnitude, or conversely a noticeable increase in accuracy for a given solution time. Further improvements in improving convergence time could be obtained by prolongating the flow field onto the smaller 'children' data points from the initial mesh in a more conservative manner rather than just transferring the same values to the children

as implemented in Section 6.6.1. Such a method is implemented successfully in Section D.

There are still a number of matters requiring further study and development, such as the limits in Algorithm 6.2 that determine whether an edge is cut or a pre-existing node is utilised. The robustness of the algorithms is also an area for further development, allowing a larger range of complex meshes to be adapted.

7 Validation test cases

The purpose of this chapter is to model a realistic three dimensional geometry, and to allow the adaption and adaption control procedures to operate upon an initial non-optimised mesh.

7.1 SPAR buoy

7.1.1 Model definition

The geometry selected is akin to that of a SPAR buoy, as used in offshore oil production, except square in section. The overall domain size has been limited from that of an external flow to one of restricted dimensions to create a more complex flow field. The structure is 1m by 1m in cross section and 2m deep, in a domain 8m long, 4m wide and 4m deep. Figure 7.1 presents a wireframe of the domain.

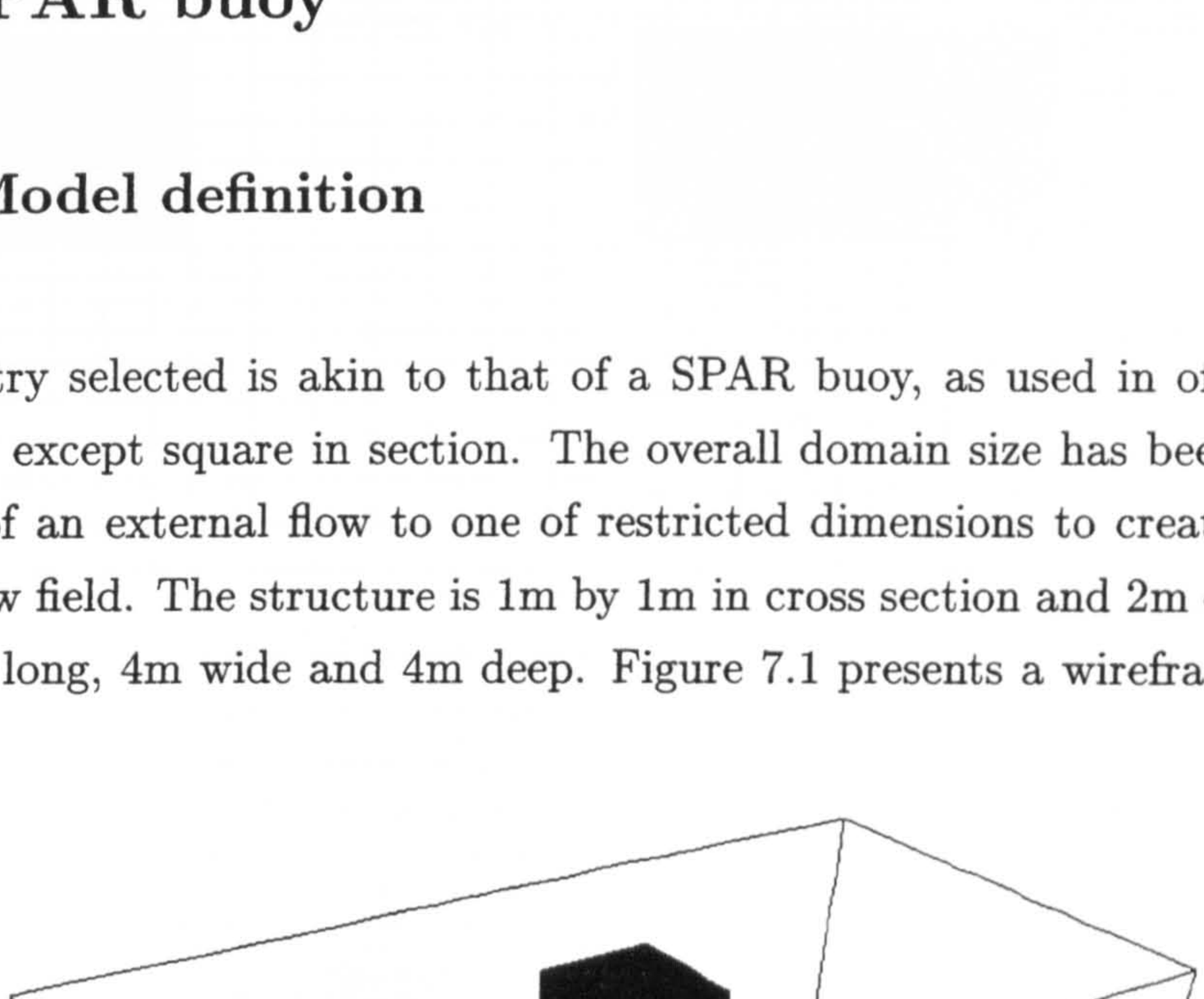


Figure 7.1: Wire frame of domain around truncated square section prism

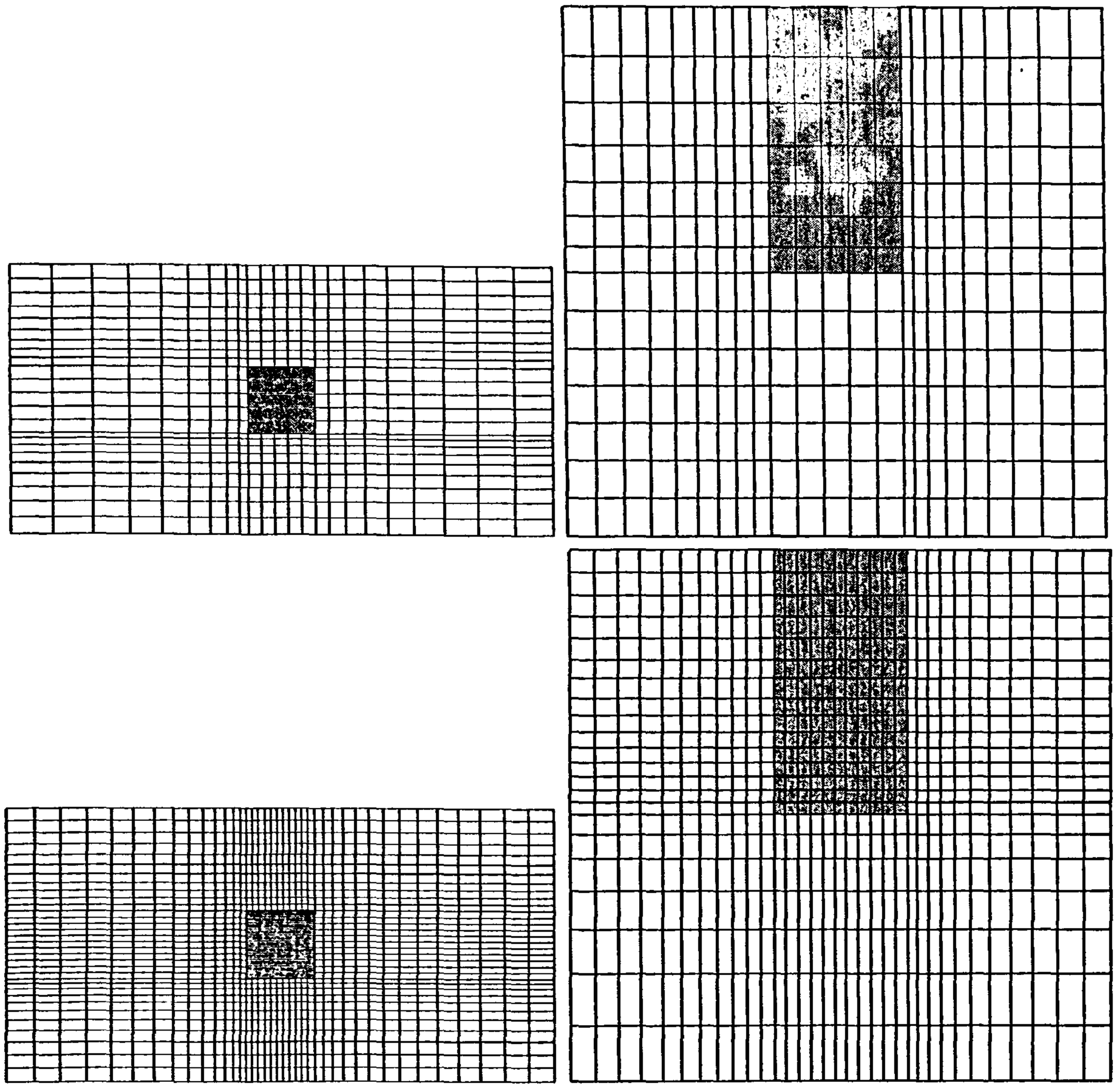


Figure 7.2: Original meshes surrounding structure in the XY and XZ planes

The initial mesh was defined to be 26 × 26 × 16, with the mesh on the structure being 6×6×8 in the x, y and z planes respectively. There was weighting towards the surface of the structure in the xy plane. For comparison purposes a denser mesh was also defined. This mesh was 42 × 36 × 24, with the surface definition being 12 × 12 × 16 in the x, y and z planes. This increased number of control volumes (29780 compared to 8528) was felt to be a realistic comparison for the increased number of control volumes created upon the structure during the adaption process. Figure 7.2 displays both meshes in the xy and xz planes.

The free stream flow field was defined to have a velocity of $5ms^{-1}$ in the X direction and zero vertical and lateral flow. The fluid was declared to have a density of 1000 kgm^{-3}. The artificial compressibility constant, $\mathcal{K}$, was given a value of 1.5, and the timestep safety factor, ω, was set to 0.5. The artificial speed of sound limit $\mathbf{c_{MIN}}$ was specified to be 1.05.

The coarser mesh was set to be adapted by both the pressure gradient field and the local geometrical properties with the appropriate adaption algorithm every 500 iterations. The cutting plane for the splitting algorithm was defined to be that of the local flow direction except in control volumes adjacent to boundaries, where it was defined to be normal to the wall. As comparison the coarse and fine meshes were run without any adaption. The convergence criteria for all three cases was defined to be when the maximum X momentum residual fell below 1×10^{-8}.

7.1.2 Results

The extra control volumes created in the coarse mesh are shown in Figure 7.3. The direct result of these extra control volumes is a lowering of the flow property differential between adjacent control volumes. This is verified by Figure 7.4, which presents the pressure differential across each control volume for the coarse mesh without adaption and after 1500 iterations with adaption (after the third level of adaption). The two iso-surfaces show the volumes of the domain with the change in pressure between control volumes greater that 0.1 and 0.25 (non dimensional).

The pressure around the structure at the base and at the tip, for all three

Mesh	Number of nodes	CPU time per iteration (s)
$26 \times 26 \times 16$ mesh	8528	9.75
With 1 adaption	8720	8.75
With 2 adaptions	8916	9.12
With 3 adaptions	9122	9.66
With 4 adaptions	9403	10.6
With 5 adaptions	9703	11.55
$42 \times 36 \times 24$ mesh	29780	39.8

Table 7.1: CPU time usage for different meshes

cases, is presented in Figure 7.5. Due to the symmetry of the flow results are presented for one side of the structure only. The streamwise velocity field for the fine and coarse adapted meshes at a variety of heights is presented in Figure 7.6, and the vertical velocity around the structure for these meshes is presented in Figure 7.7.

The convergence history for the three cases is shown in Figure 7.8, and Table 7.1 lists the CPU time per iteration for each of the meshes utilised.

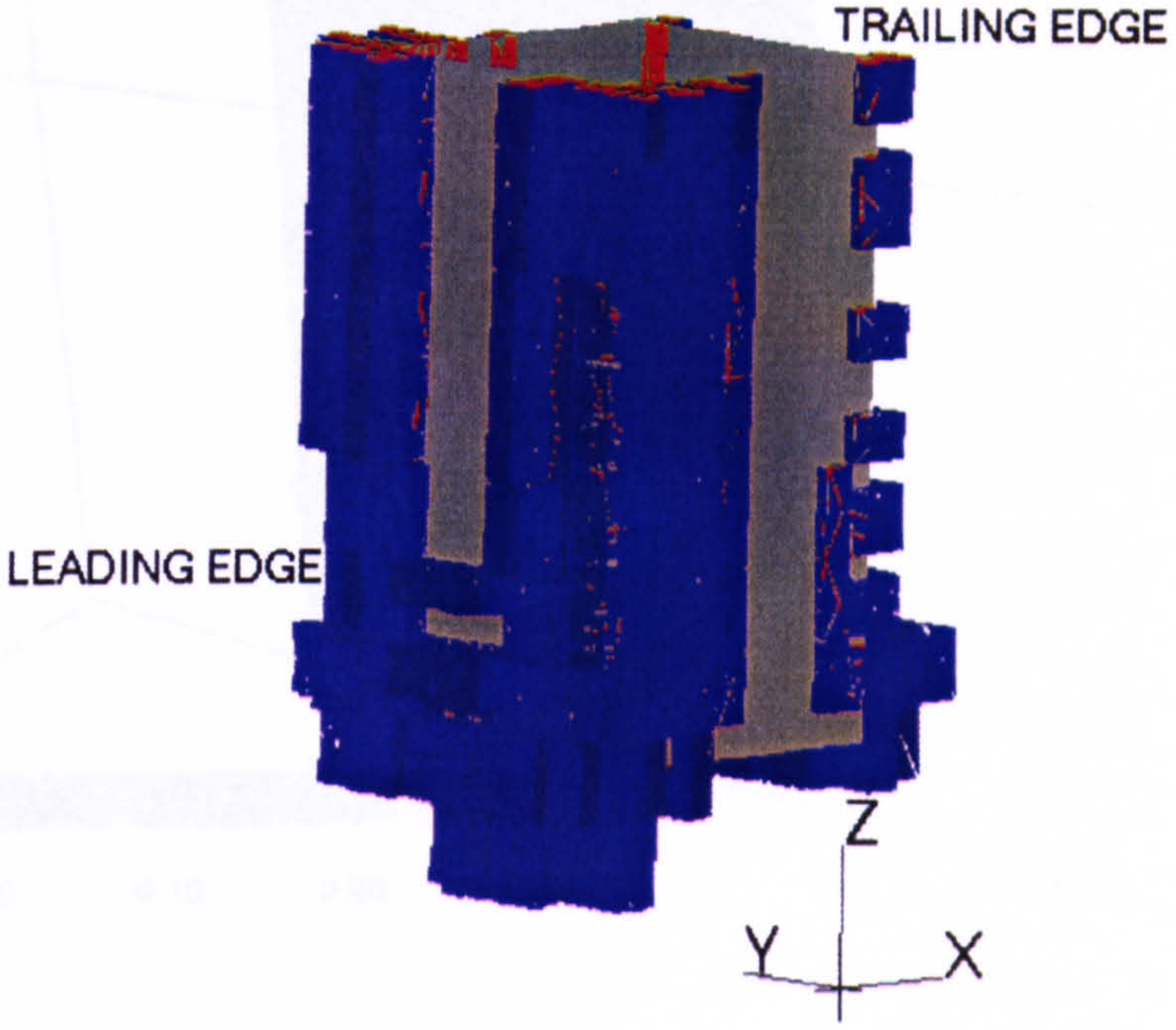


Figure 7.3: Control volumes created during solution of flow

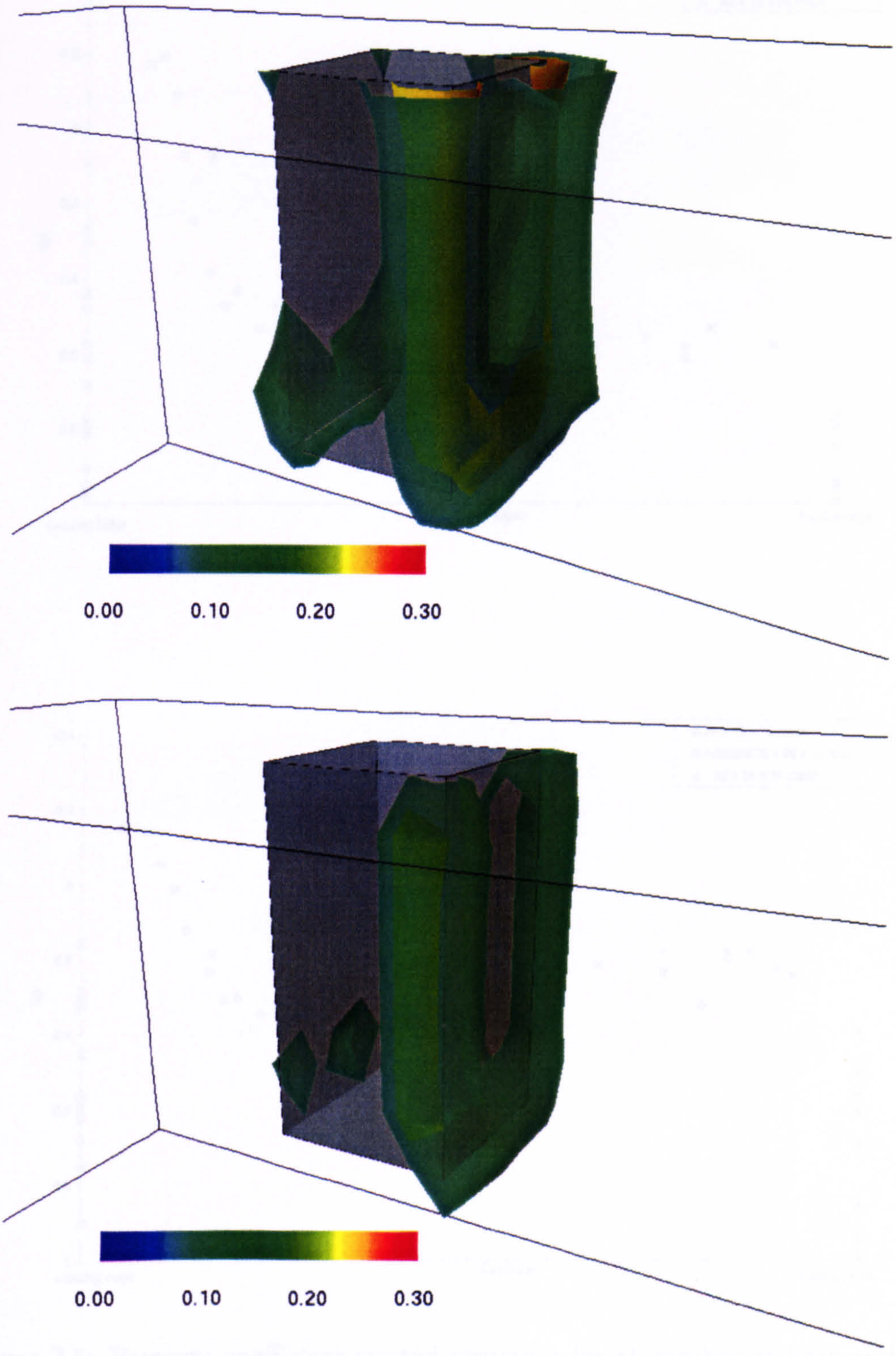


Figure 7.4: Volumes of domain space with pressure differential values over 0.1 and 0.25 for unadapted and adapted meshes respectively

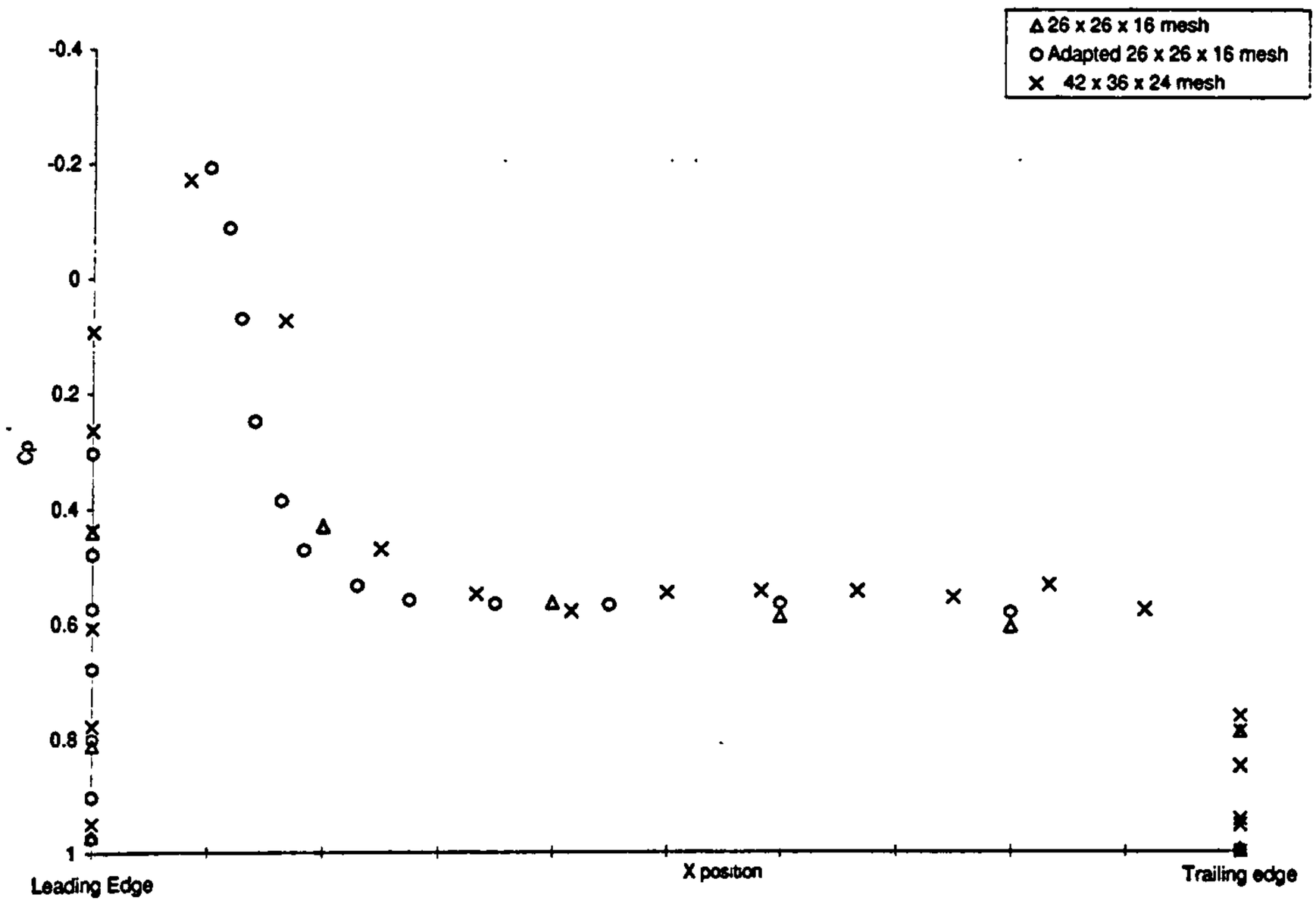


Figure 7.5: Pressure coefficient around structure for all meshes at base and tip respectively

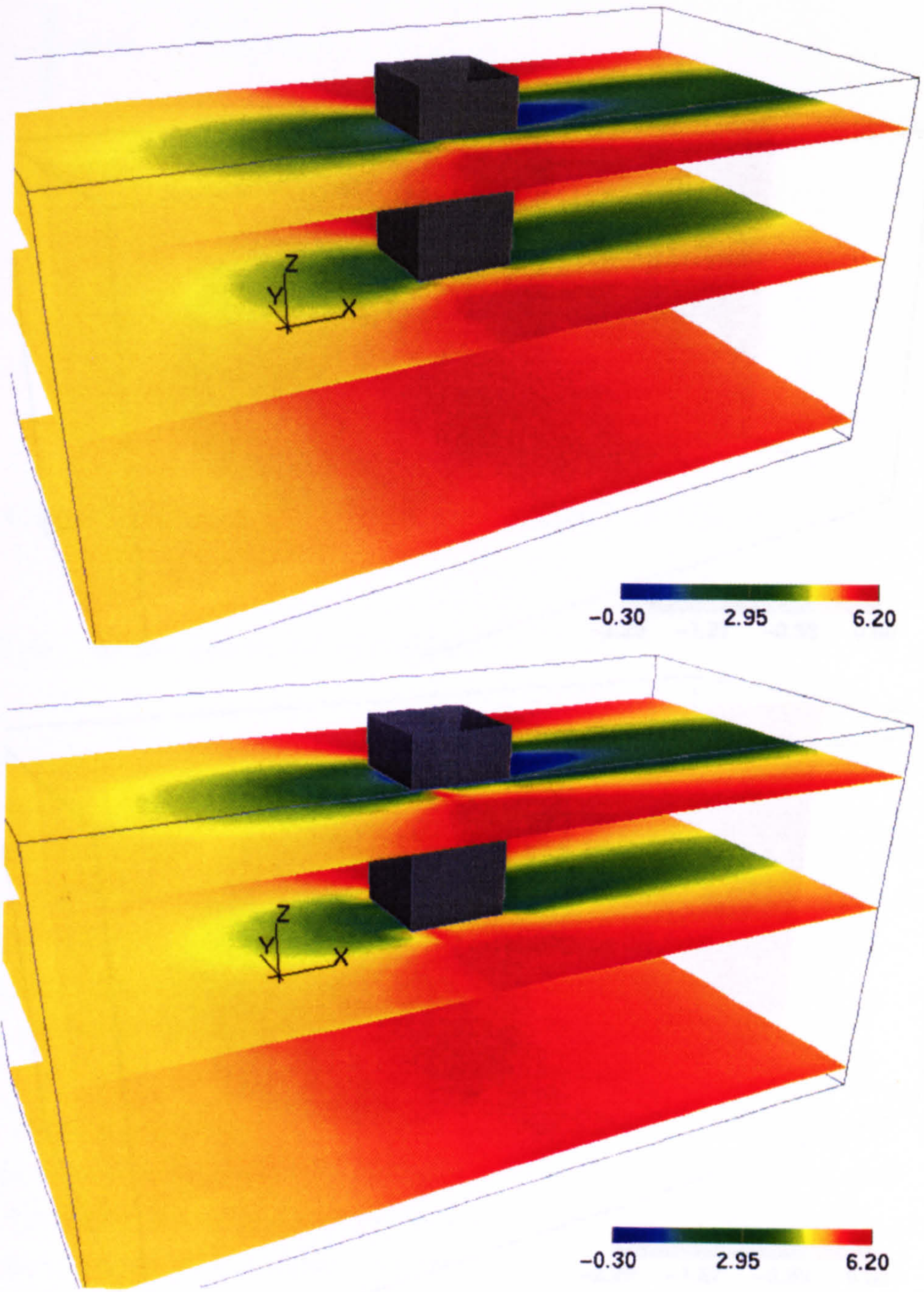


Figure 7.6: Streamwise velocity at three vertical positions for $42 \times 36 \times 24$ and adapted $26 \times 26 \times 16$ meshes respectively

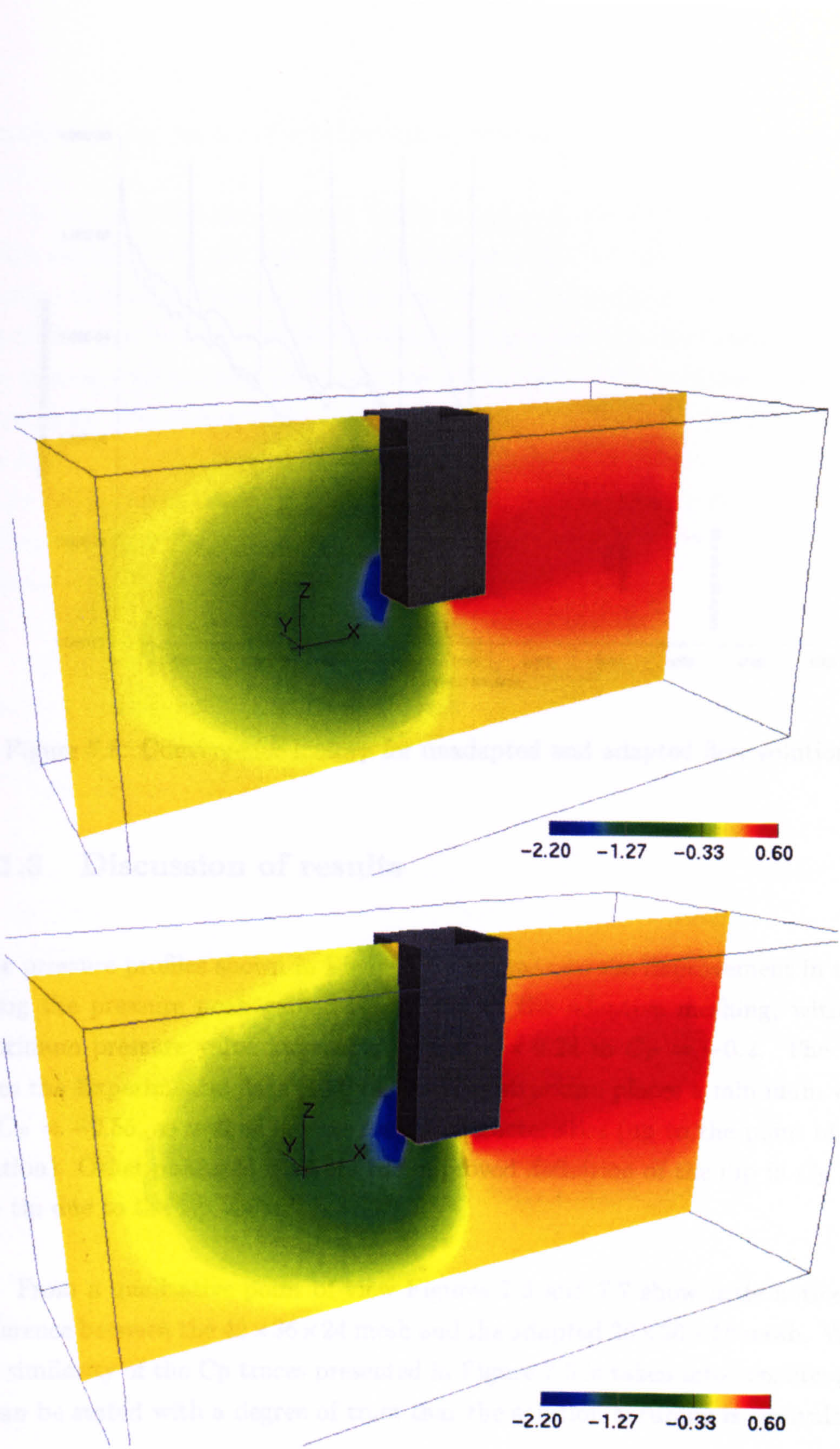


Figure 7.7: Vertical velocity profile for $42 \times 36 \times 24$ and adapted $26 \times 26 \times 16$ meshes respectively

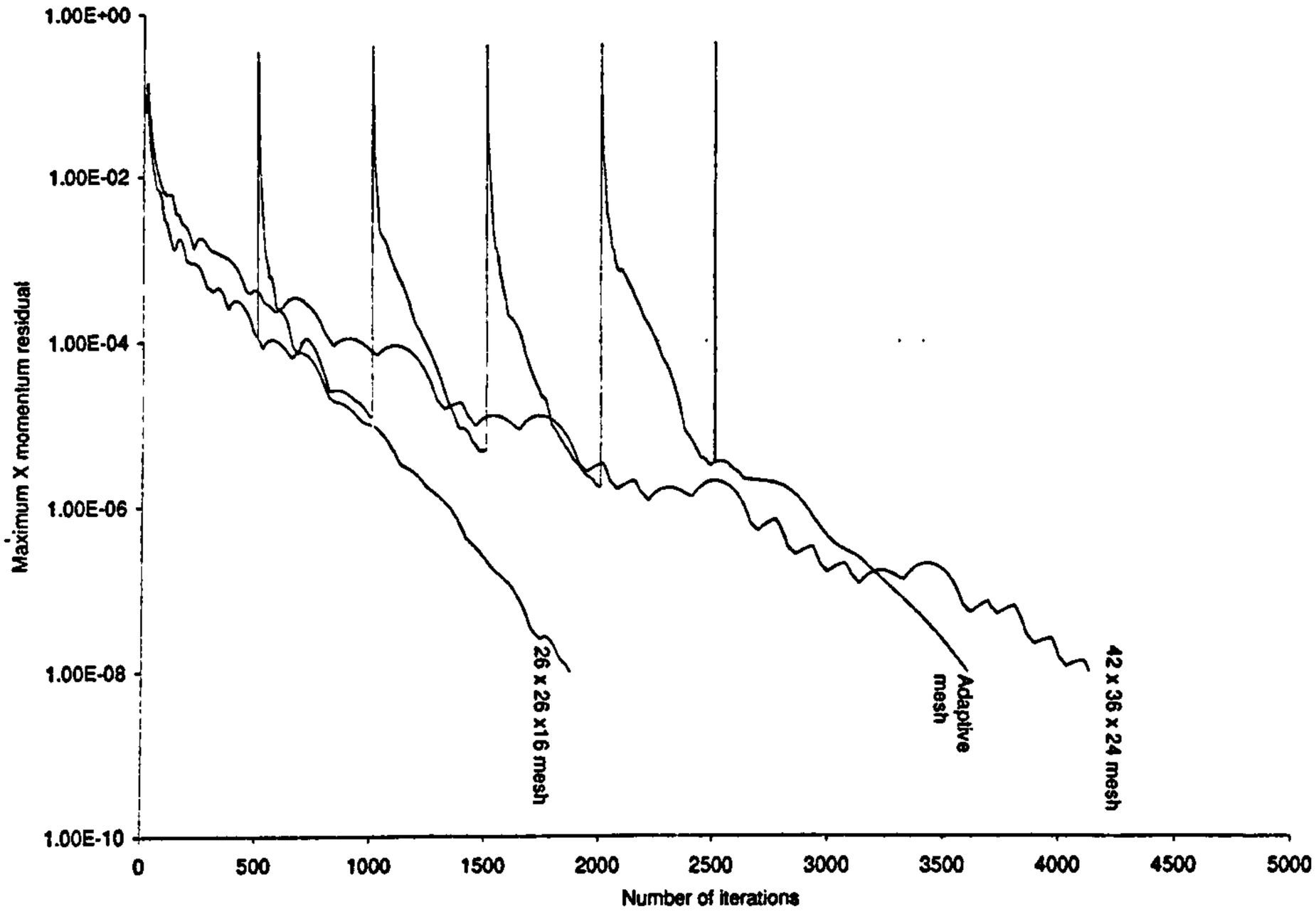


Figure 7.8: Convergence history for unadapted and adapted flow solutions

7.1.3 Discussion of results

The pressure profiles shown in Figure 7.5 demonstrate the improvement in modelling the pressure peak gained by the use of the adaptive meshing, with the maximum pressure value increasing from $C_P = 0.24$ to $C_P = -0.2$. The data from the Experimental data [103] of a similar structure places a minimum value of $C_P = -0.55$, as well as having similar characteristics (up to the point of separation). Other points to note are the improved definition of the dip in C_P near the tip due to the tip vortex.

From a qualitative point of view Figures 7.6 and 7.7 show little noticeable difference between the $42 \times 36 \times 24$ mesh and the adapted $26 \times 26 \times 16$ mesh. When the similarity of the Cp traces presented in Figure 7.5 is taken into consideration it can be stated with a degree of trust that the solution accuracy is equivalent.

The adapted mesh does take considerably longer to converge (3612 iterations compared to 1851 iterations) than the unadapted $26 \times 26 \times 16$ mesh. A great deal of this is due to the large errors introduced during the transference of information to the finer mesh. As commented in Section 6.7 this should be improved

considerably by the use of a conservative prolongation scheme.

It can be noted that in both Tables 6.2 and 7.1 the CPU time per iteration drops slightly after the first level of adaption, even though there are a greater number of control volumes. This is due to the presence of a routine within the algorithm that removes redundant construction nodes (i.e. face vertices that do not increase the geometrical resolution of the face). The initially constructed 'dual mesh' (see Figure 2.11) contains a considerable number of such nodes, and are hence removed. With the simpler face definition the algorithm is noticeably faster. It is therefore advisable to include this routine in the solver as a matter of course, before the solution phase is started, to optimise the mesh and increase efficiency.

Table 7.1 also shows that the CPU time requirements, per node, remains relatively constant after adaption. The $42 \times 36 \times 24$ mesh requires 1.33×10^{-3} seconds per node, while after 5 levels of adaption the coarser mesh requires 1.19×10^{-3} seconds per node.

The increase in solution accuracy gained is equivalent to that of more than double the initial mesh density in all directions. The initial mesh had 182 control volumes on the surface of the structure, which increased to 765 with adaption, while the denser mesh had 842 nodes on the surface. With only 1175 extra control volumes being created (equivalent to 13% of the original) in a total of 8528, the increased accuracy is equivalent to that gained by adding 21252 (250 % more) control volumes in a structured mesh format. The total solution time for the adapted mesh is 41190 seconds and 144493 seconds for the $42 \times 36 \times 24$ mesh; a 75 % reduction in time. In addition to this increase in efficiency in CPU time, the memory requirements of the adapted mesh are 2388 K Bytes and 8340 K Bytes for the $42 \times 36 \times 24$; a drop of 70%. A purely structured mesh of the same size would require 714 K Bytes, so the memory for the adapted mesh is still greater by a factor of 3.35.

Figure 7.3 shows that the vast majority of control volume splits were in the region of the edges of the structure, which eventually led to some problems within the computational code. The algorithm was able to cope with situations where large (in the order of 32) numbers of control volumes were adjacent to one control volume in a given direction, but the computational code began to incur

stack overflows. The solution to this problem is the creation of a more complex control volume selection process, with limits upon the relative size of adjacent control volumes. A more refined cutting plane direction algorithm would also help alleviate the problem.

7.2 Summary

A three dimensional flow has been modelled, with mesh discretisation defined via an automated mesh adaption control process. Results show a marked improvement in the capture of the pressure peaks in flow by the use of control volume splitting, while the geometrical adaption processes ensured mesh degradation did not occur.

Difficulties were encountered with the lack of restrictions placed upon the relative size of adjacent control volumes, which resulted in memory allocation problems. Solutions to this problem would most likely contain limiters upon adjacent control volume size.

Overall solution efficiency, in terms of both memory requirements and CPU time usage, has been increased considerably. For an equivalent level of accuracy the CPU time required has been dropped by 75% and the memory requirements have dropped by 70 %. The memory requirements are still greater than those required by a purely structured mesh by a factor of 3.35.

8 Conclusions and further work

The aim of this work was to investigate methods of mesh adaption, and their use to maximise solution efficiency via automated control of the mesh definition. This mesh definition was to be contained within a framework that would eventually allow hydrodynamic flows around complex geometries to be modelled. The major achievements of this work are:

Flow Solver A flow solver capable of modelling three dimensional incompressible inviscid flow has been created that operates on arbitrary polyhedral control volumes. A face based algorithm allows the conservative formulation of the Euler equations to be solved upon the arbitrary geometrical structure utilised while a first order upwinding formulation of artificial compressibility has been implemented to model incompressible flow. Steady flow cases around a 2D hump and 2D foil have been validated and flow around a 3D truncated square cylinder has been modelled. A current restriction with the present scheme is the poor accuracy associated with a first order spatial discretisation scheme.

Measurement of grid quality The measurement of solution accuracy and geometrical grid quality in order to obtain distortion error dependencies has quantified the relative importance of four key geometrical properties. In normalised order of importance they were: geometrical skew (82 %); aspect ratio (8%); misalignment of facial normals (2%); and deviation from planarity (6%). For simple flows each of these properties varied linearly with solution accuracy but as the flow became more complex non-linear cross coupling appeared to be present. A method of least squares fitting has been developed where geometrical errors were induced and the solution error measured, which was used to determine the relative importance of the geometrical parameters. From this analysis an overall quantitative value of geometrical grid quality can be obtained for a given mesh. This form of measurement has been used to measure the quality upon three geometries and for a variety of mesh densities.

Mesh Adaption Two forms of mesh adaption have been implemented and shown

to improve solution accuracy and increase efficiency. A generalised cellular splitting algorithm has been created that allows the splitting of an arbitrary polyhedra in an arbitrary cutting plane, or the rejoining of two 'children' to re-create the original control volume. Three other algorithms have been implemented to stretch and convect the mesh entities. The splitting algorithm has been used to decrease discretisation error and the convection algorithms have been used to decrease distortion error. The lowering of discretisation error has been used in two cases - flow over a 2D hump and around a 3D structure - while the control and lowering of distortion error has been used in three cases - flow over a 2D hump, around a 2D foil and around a 3D structure. The use of the splitting algorithm has increased the efficiency of the solver by up to 60 orders of magnitude in one case study. The algorithms to reduce distortion error have been proven to remove induced errors that were causing up to 50% error in the global solution.

8.1 Additional Observations

Throughout this work observations have been made about the techniques and strategies employed to achieve the overall aims. Some of the more important observations are outlined below.

- Arbitrary facial and cellular definition allows the accurate definition of a given geometry. This definition places no geometrical restrictions upon any given control volume, nor upon cellular connectivity.

- Unstructured data can be used to systematically adapt a mesh applying different adaption strategies. The subsequent unstructured/hybrid meshes can improve the efficiency of cell placement within the global mesh or to improve the quality of structured meshes without the necessity of a complete re-mesh.

- The data structure utilised uses only nodes and faces to completely define the finite volume discretisation, which results in a 37% increase in memory efficient from a previous structure that used a complete connectivity of nodes, edges, faces and cells.

- The use of geometrical entities with the connectivity list and the arbitrary control volume definition eliminates the problem of 'hanging nodes'. There can still be a sharp increase in the size of adjacent control volumes, which could cause instabilities in the solution, but such traits have not become evident in the test cases used to date.

- The capability of correctly defining the mathematical flux through an arbitrary volume for artificial compressibility has been achieved via the derivation of a generalised flux vector that allows arbitrary shape, size and spatial orientation as well as moving meshes.

- Pressure gradients provide a suitable definition of flow gradients, and are more sensitive to solution error than velocity gradients.

- A grid dependency study that considers discretisation alone may lead to erroneous conclusions, as geometrical properties can strongly influence both local and global solution results.

- The adaption of less than 10% of the total number of control volumes can considerably improve the solution accuracy, as detailed in Section 6.6.1 and Chapter 7.

8.2 Further work

This final section outlines the areas in which there is scope for further work. Some of the proposals are aimed at improving the accuracy, range of application and efficiency of solutions obtained, whereas others aim at improving the flexibility of the adaptive process.

Flow solver accuracy The highest priority is to improve the accuracy of the flow solver. This can be achieved through the use of a higher order scheme; indeed Hirsch [15] notes that, "first order schemes are not sufficiently accurate for practical purposes and the passage to second order schemes is essential". Problems with higher order schemes are primarily instabilities caused by discontinuities as well as difficulties in creating an efficient stencil on an unstructured mesh. Rycroft [20] met with some success in creating a

higher order stencil for such arbitrary meshes, and details several possible techniques that could be utilised.

Viscous flows The major development of the code required is the ability to model viscous flows. There are three basic levels of viscous flow solvers currently in use: Reynolds Averaged Navier Stokes (RANS), Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS). The simplest, RANS flow solvers, model all turbulence with empirical models which typically use some measure of the distance to the wall to determine the eddy viscosity, and hence the Reynolds stresses. Such solvers however lack accuracy once a flow has separated from the body. As such flows are common place in realistic marine situations a great deal of the computational expense can be wasted upon inaccurate solutions. At the other end of the scale are DNS solvers, which solve the entire flow equations without averaging or approximation. The scale of geometries successfully modelled by such methods to date limits DNS to numerical experimentation of small scale turbulence, and not large $\mathcal{R}e$ problems. Probably the current framework would be most suitably employed within a LES code. Such methods calculate eddies and viscous effects down to a given size, and models the remainder (which typically contains somewhere in the region of 5% of the energy field). The key issue is that the smallest eddy calculated is defined by the grid size. Thus in normal LES codes the $\mathcal{R}e$ that is capable of being calculated is set by the mesh size. However with the framework developed within this work the situation can be reversed; the size of the mesh can be set by the required eddy size for a given problem via the control volume splitting algorithm.

Free surface and unsteady solutions Other flow solver developments that are also required for the modelling of the marine environment are the incorporation of a free surface boundary condition and the modelling of unsteady flows in the three dimensional flow solver. Of the two tasks the incorporation of the free surface is expected to be the more arduous due to the requirements of accuracy and numerical stability. The process will be aided by the adaptive meshing framework created however; the geometrical quality measures will aid both accuracy and stability in areas of highly non-linear wave forms, and the splitting/merging algorithms will maintain solution efficiency alongside the accuracy requirements near the free surface. The use of underlying surface definition will also greatly benefit the model

by allowing complex surface breaking geometries. That is, data points on a body at the free surface will be able to track across the body surface while still maintaining connectivity to the free surface.

Unsteady flows can be modelled via the use of pseudo-time, as detailed in Section 4.1.2. The numerical stability and convergence of such a scheme is not a concern after the two dimensional studies undertaken. What is the major consideration however is the accuracy levels attainable with this method. The spatial accuracy of the basic flow solver will still be the over riding factor upon the accuracy of such unsteady simulations, and as such remains the major concern in the search for greater solution accuracy via the mathematical model.

Grid quality definition and quantification Success has been achieved in the quantification of the influence of certain geometrical properties upon solution accuracy, but the level of accuracy of the values is a matter for debate. A larger study, of more range and of a greater number of test cases, would give greater clarity to the relative influence of the geometrical properties. A further study of interest would be to investigate the effects and influence of the other geometrical (and flow) properties detailed in Chapter 5, although it is strongly expected that the other geometrical parameters will be of secondary importance and will be highly cross coupled with the properties already tested.

Mesh adaption The work has achieved automated adaption of the geometrical mesh, incorporating both geometrical and flow solution measures of the solution error and selecting the best corresponding method of adaption. This work should be developed to incorporate more complex merging and splitting routines. One area for expansion is the merging of two (or more) first generation control volumes, or control volumes of different branches and generations in the binary tree data structure. This will provide much greater flexibility within the code, and allow even greater efficiency by not being restricted to any minimum number of control volumes in a given region. The splitting algorithm is fairly robust, but somewhat inefficient in areas of very high flow gradients where a given control volume will require multiple splittings. The development of an algorithm that can split the control volume into a given number of children (N say) would be of great benefit. The algorithm would require little more data or processes than required currently. Instead of one cutting plane there would be (N-1), and

the faces and face edges could be processed in a similar manner as presently. The basic techniques, algorithms and computational routines are present that will allow rapid development.

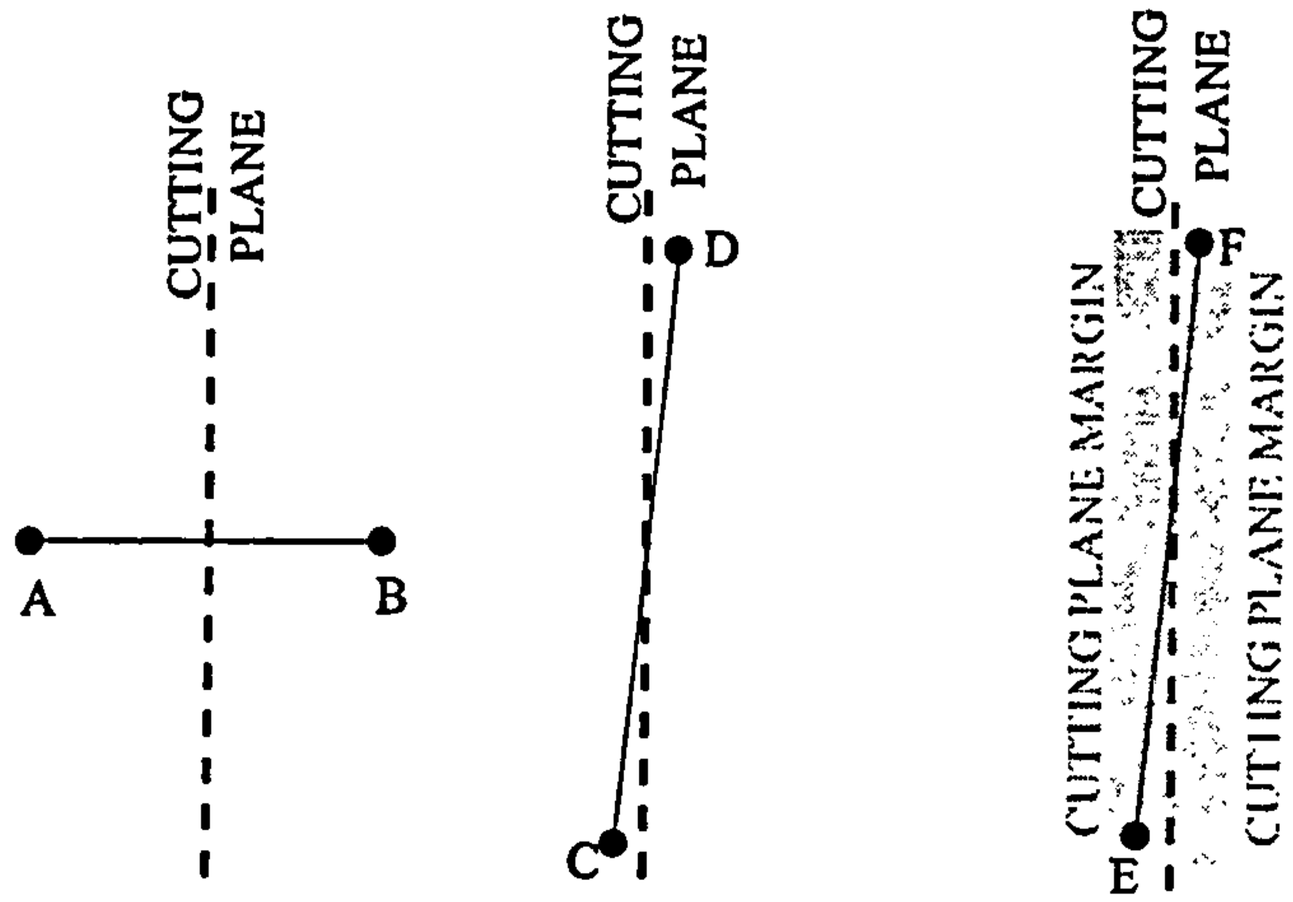


Figure 8.1: Incorporation of more intuitive selection process in splitting algorithm

The only area in which the current splitting algorithm is lacking is in the decision of how to manipulate an edge that crosses the cutting plane. Figure 8.1 demonstrates the problem. Under the current algorithm edges AB and CD are dealt with in the same manner. Because the edge crosses the cutting plane at a normalised position of 0.5 (on the edge) both would have a new node created, and the nodes A, B, C, D placed in the corresponding face list. If an area on either side of the cutting plane were also considered, as in the case of edge EF, then the edge could be selected as effectively running parallel to the cutting plane and dealt with accordingly. This would result in fewer new nodes being created, saving memory and algorithm CPU time, but also provide the splitting algorithm with a more intuitive, robust and flexible technique of edge selection.

Efficiency of solver To further improve the flow solver efficiency there are two topics that would be most beneficial; parallelisation and the development of multigridding techniques.

Parallel solutions of adapted unstructured meshes such as those used in this work has been achieved with "close to optimal" performance by Rycroft [20] on a small number of processors. Topics suggested for future development were the maintenance of performance on highly parallel solutions and dynamic load balancing, particularly in areas of mesh adaption. With the relative price of computational hardware continuing to drop the benefits of

increasing efficiency via the use of multiple processors become increasingly important.

The investigations detailed in Appendix D show that the use of multigridding is greatly beneficial to solution efficiency with minimal computational costs. The levels of efficiency attainable are well documented elsewhere [51, 104], and show that the optimal number of grids is well in excess of the two that have been utilised here. Few problems are envisaged with increasing the number of meshes greatly, although efficiency of the algorithm could be improved by resetting the binary tree links in a more optimal manner to avoid the referencing of deactivated entities, the reference of which is reset via a number of routines. As commented in Section D.4, the various parameters also need greater study and optimal values defined.

The final area that the author believes could greatly enhance the capabilities and efficiency of the solver is the combination of the multigridding techniques with the adaptive meshing. Presently the adaptive meshing is not used with the multigridding, but it should be possible within the framework developed. Concerns would be the conservation of mass and momentum and the maintenance of control over the process, ensuring that control volume splitting was only occurring upon the correct level of the mesh, and the retention of the correct entity connectivity. With such refinement of the flow solver framework, algorithms and routines developed by this work the overall efficiency and accuracy of the flow field modelling could be greatly improved.

References

[1] V. C. Patel, H. C. Chen, and S. Ju. "Computations of ship stern and wake flow and comparisions with experiment". *Journal of ship research*, 34(3):179–193, September 1990.

[2] F. Stern, Y. Toda, and H. T. Kim. "Computation of viscous flow around propeller-body confiurations: Iowa axisymmetric body". *Journal of ship research*, 35(2):151–161, June 1991.

[3] Y. Kodama. "Computation of ship's resistance using an NS solver with global conservation - flat plate and Series 60 ($C_B = 0.6$) hull". *JSNA*, 172:21–28, December 1992.

[4] H. Miyata and H. Akimotoand F. Hirosima. "CFD performance prediction simulation for hull-form design of sailing boats". *Marine Science and Technology*, 2(4):257–267, 1997.

[5] H. Karadeniz. "Wave-current and fluid-structural interaction effects on the stochastic analysis of offshore structures". *International Journal of Offshore & Polar Engineering*, 3(2):107–114, June 1993.

[6] M. Caponnetto, A. Castelli, P. Dupont, B. Bonjour, P. L. Mathey, S. Sanchi, and M. L. Sawley. "Sailing Yacht Design Using Advanced Numerical Flow Techniques". In 14^{th} *Chesapeake sailing yacht symoposium*, pages 13–33. SNAME, 1999.

[7] J.D. Anderson. *"Fundementals of Aerodynamics"*. McGraw Hill Inc., 1991. ISBN 0-07-100767-9.

[8] S. F. Hoerner. *"Fluid Dynamic Drag"*. The Author, 1965.

[9] C. Hirsch. *"Numerical Computation Of Internal And External Flows"*, volume 1. J. Wiley and sons, 1988. ISBN 0-471-92385-0.

[10] O. Reynolds. "An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels". In *Philisophical Transactions of the Royal Society*, volume 174, 1883.

[11] O. Reynolds. "On the dynamic theory of incompressible viscous fluids and the determination of the criterion". In *Philisophical Transactions of the Royal Society*, volume 186, 1895.

[12] N. Rott. "Note on the History of the Reynolds number". In *Annual Review of Fluid Mechanics*, volume 22, pages 1–11, 1990.

[13] W. Froude. "Experiments on the Surface-Friction experienced by a Plane moving through water". 42nd Report by the British Society for the Advancement of Science, 1872. 118-124.

[14] W. Froude. "Report to the Lords Admiralty on Experiments for the Determination of the Frictional Resistance of Water on a Surface, under various conditions, performed at Chelston Cross, under the Authority of their Lordships". 44th Report by the British Society for the Advancement of Science, 1874. 249-254.

[15] C. Hirsch. *"Numerical Computation Of Internal And External Flows"*, volume 2. J. Wiley and sons, 1988. ISBN 0-471-92452-0.

[16] S. J. P. Watson. "Experiences with validation of CFD for marine hydrodynamics". In *Technical papers of 1st MARNET CFD Workshop, CIMNE Barcelona*, November 1999.

[17] T. G. Thomas. Verbal communication, February 2000. Aerodynamic and Flight Mechanics Group, School of Engineering Sciences, University of Southampton.

[18] J.D. Anderson. *"Computational Fluid Dynamics - The basics with applications"*. McGraw Hill Inc., 1995. ISBN 0-07-113210-4.

[19] J. F. Thompson, B. K. Soni, and N. P. Weatherill, editors. *"Handbook of Grid Generation"*. CRC Press, 1999. ISBN 0-8493-2687-7.

[20] N. C. Rycroft. *"An adaptive three dimensional, finite volume, Euler solver for distributed architecture using arbitrary polyhedral cells"*. PhD thesis, University of Southampton, 1998.

[21] R. Haimes, S. D. Connell, and S. A. Vermeersch. "Visual grid quality assesment for 3D unstructured meshes". AIAA 1993; internet.

[22] B. J. Gribben, K. J. Badcock, and B. E. Richards. "Towards Automatic Multiblock Topology Generation". Aero Report 9826, Aerospace Engineering Dept, University of Glasgow, August 1998.

[23] W. G. Habashi, J. Dompierre, Y. Bourgault, M. Fortin, and M. G. Vallet. "Certifiable Computational Fluid Dynamics Through Mesh Optimization". *AIAA Journal*, 36(5):703–711, May 1998.

[24] H. Miyata, A. Kanai, T. Kawamura, and J-C. Park. "Numerical simulation of three-dimensional breaking waves". *Journal Marine science and technology*, 1:183–197, 1996.

[25] R. Azcueta, S. Muzaferija, and M. Perić. "Computation of water and air flow around ships". Paper derived from internal report, 1998.

[26] S. Muzaferija and M. Perić. "Computation of free-surface flows using interface tracking and interface capturing methods". Paper derived from internal report, 1998.

[27] H. Zhang, M. Reggio, J. Y. Trepanier, and R. Camarero. "Discrete form of the gcl for moving meshes and its implementation in cfd schemes". *Computers & fluids*, 22(1):9–23, 1993.

[28] R. G. Hindman. "Generalized coordinate forms of governing fluid equations and associated geometrically induced errors". *AIAA Journal*, 20:1359, 1982.

[29] I. Demirdzic and M. Perić. "Space conservation law in finite volume calculations of fluid flow". *Int. Journal of Numerical Methods in Fluids*, 8:1037, 1988.

[30] J. F. Thompson. "A Reflection on Grid Generation in the 90s: Trends, Needs and Influences", 1996. http://www.ERC.MsState.Edu/ joe/gridconf/.

[31] R. R. Cosner. "Future Requirements in Surface Modelling and Grid Generation". In *Proceedings of the Surface Modelling, Grid Generation and Related Issues in Computational Fluid Dynamics Workshop*, NASA Conference Publication 3291, page 751, NASA Lewis Research Centre, Cleveland, OH, May 1995.

[32] D. K. Clark, M. D. Salas, and H. A. Haasan. "Euler calculations for multielement airfoils using cartesian grids". *AIAA Journal*, 24:353–359, 1987.

[33] J. J. Quirk. "An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies". *Computers and Fluids*, 23(1):125–142, 1994.

[34] M. Beddhu, M. Y. Jiang, D. L. Whitfield, L. K. Taylor, and A. Arabshahi. "CFD validation of the free surface flow around DTMB Model 5415 using Reynolds Averaged Navier-Stokes Equations". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 373–393, May 1998.

[35] C. W. Lin, S. Percival, and L. Fisher. "Viscous flow computations on an appended ship by Chimera RANS scheme". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 161–180, May 1998.

[36] A. Masuko. "Numerical simulation of viscous flow for complex geometries using overlapped grid method". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 181–197, May 1998.

[37] C. Yang and R. Löhner. "Fully nonlinear ship wave calculation using unstructured grid and parallel computing". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 125–150, May 1998.

[38] M. Aftosmis, D. Gaitonde, and T. S. Tavares. "Behaviour of linear reconstruction techniques on unstructured meshes". *AIAA Journal*, 33(11):2038–2049, 1995.

[39] N. P. Weatherill. "Mixed structured-unstructured meshes for aerodynamic flow simulation". *Aeronautical Journal*, 94(934):111–123, 1990.

[40] J. A. Shaw and A. J. Peace. "The modelling of aerodynamic flows by solution of the Euler Equations on mixed polyhedral grids". *International Journal of Numerical Methods in Engineering*, 35:2003–2029, 1992.

[41] D. Sharov and K. Nakahashi. "Hybrid prismatic/tetrahedral grid generation for viscous flow applications". *AIAA Journal*, 36(2):157–162, 1998.

[42] S. Kano and K. Nakahashi. "Flow Computationa Around Delta Wings Using Unstructured Hybrid Grids". *Journal of Aircraft*, 36(2):374–379, March 1999.

[33] J. J. Quirk. "An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies". *Computers and Fluids*, 23(1):125–142, 1994.

[34] M. Beddhu, M. Y. Jiang, D. L. Whitfield, L. K. Taylor, and A. Arabshahi. "CFD validation of the free surface flow around DTMB Model 5415 using Reynolds Averaged Navier-Stokes Equations". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 373–393, May 1998.

[35] C. W. Lin, S. Percival, and L. Fisher. "Viscous flow computations on an appended ship by Chimera RANS scheme". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 161–180, May 1998.

[36] A. Masuko. "Numerical simulation of viscous flow for complex geometries using overlapped grid method". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 181–197, May 1998.

[37] C. Yang and R. Löhner. "Fully nonlinear ship wave calculation using unstructured grid and parallel computing". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 125–150, May 1998.

[38] M. Aftosmis, D. Gaitonde, and T. S. Tavares. "Behaviour of linear reconstruction techniques on unstructured meshes". *AIAA Journal*, 33(11):2038–2049, 1995.

[39] N. P. Weatherill. "Mixed structured-unstructured meshes for aerodynamic flow simulation". *Aeronautical Journal*, 94(934):111–123, 1990.

[40] J. A. Shaw and A. J. Peace. "The modelling of aerodynamic flows by solution of the Euler Equations on mixed polyhedral grids". *International Journal of Numerical Methods in Engineering*, 35:2003–2029, 1992.

[41] D. Sharov and K. Nakahashi. "Hybrid prismatic/tetrahedral grid generation for viscous flow applications". *AIAA Journal*, 36(2):157–162, 1998.

[42] S. Kano and K. Nakahashi. "Flow Computationa Around Delta Wings Using Unstructured Hybrid Grids". *Journal of Aircraft*, 36(2):374–379, March 1999.

[43] R. J. Gomez and E. C. Ma. AIAA paper 94-1859-CP, 1994. Presented at 12th Applied Aerodynamics Conference, Colarado Springs.

[44] E. Turkel. "Accuracy of schemes with non-uniform meshes for compressible fluid-flows". *Applied numerical Mathematics*, 2(6):529–550, 1986.

[45] R. C. Swanson and R. Radespiel. "Cell Centered and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations". *AIAA Journal*, 29(5):697–703, May 1991.

[46] A. M. Wright and S. R. Turnock. "Efficient multi-level adaptation methods for unstructured polyhedral computational meshes". Ship Science Report 109, Department of Ship Science, University of Southampton, 1999.

[47] N. C. Rycroft and S. R. Turnock. "Hybrid cell finite volume Euler solutions of flow around a main-jib sail using an IBM SP2". In *Proceedings of Parallel CFD 97, Manchester*, May 1997.

[48] N. C. Rycroft and S. R. Turnock. "3-D Multiblock Grid Generator; FLEXIMESH". Ship Science Report 101, Department of Ship Science, University of Southampton, 1997.

[49] K. D. Squires, T. Hino, and Y. Kodama. "Application of the fractional step method to free surface flow problems". *Papers of ship research Institution*, 27(3), July 1990.

[50] J. J. Kim, H. T. Kim, and S. H. Van. "RANS simulation of viscous flow and surface wave fields around ship models". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 105–123, May 1998.

[51] J. Farmer, L. Martinelli, and A. Jameson. "Fast multigrid method for solving incompressible hydrodynamic problems with free surfaces". *AIAA Journal*, 32(6):1175–1182, June 1994.

[52] R. Azcueta, I. Hadzic, S. Muzaferija, and M. Perić. "Computation of Flows with Free Surfaces". In *Technical papers of 1st MARNET CFD Workshop, CIMNE Barcelona*, November 1999.

[53] S. R. Turnock and A. M. Wright. "A fully adaptive free surface definition algorithm suitable for modelling unsteady wave flow around floating structures". In *Proceedings of* 3^{rd} *ASME/JSME Joint Fluids Engineering Conference*, July 1999. San Fransisco, California.

[54] P. D. Lax. "Weak solutions of non-linear hyperbolic equations and their numerical computation". *Comm. Pure and Applied Mathematics*, 7:154–193, 1954.

[55] W. E. Milholen II, N. Chokani, and J. Al-Saadi. "Performance of Three-Dimensional Compressible Navier-Stokes Codes at Low Mach numbers". *AIAA Journal*, 34(7):1356–1362, July 1996.

[56] T. Hino. "Computation of Free Surface Flow around an Advancing Ship by the Navier-Stokes equations". In *Proceedings 5th International Conference on Numerical Ship Hydrodynamics*, pages 103–117, 1989. National Academy Pres, Washington D.C.

[57] H. Miyata, T. Sato, and N. Baba. "Difference solution of a viscous flow with free-surface wave about an advancing ship". *Journal of Computational Physics*, 72:393–421, 1987.

[58] Y. Tahara, F. Stern, and B. Rosen. "An Interactive Approach for Calculating Ship Boundary Layers and Wakes for Nonzero Froude Number". *Journal of Computational Physics*, 98(1):33–53, 1992.

[59] J. H. Ferziger and M. Perić. *"Computational Methods for Fluid Dynamics"*. Springer, 1997. ISBN 3-540-59434-5.

[60] A. J. Chorin. "A numerical method for solving incompressible viscous flow problems". *Journal of Computational Physics*, 2:12–26, 1967.

[61] A. Rizzi and L.E. Eriksson. "Computation of inviscid incompressible flow with rotation". *Journal of Fluid Mechanics*, 153:275 – 312, 1985.

[62] B. Engquist and A. Majda. "Absorbing boundary conditions for the numerical simulation of waves". *Mathematical of computation*, 31(139):629–651, July 1977.

[63] J. D. Ramshaw and V. A. Mousseau. "Accelerated artificial compressibility method for steady state incompressible flow calculations". *Computers and Fluids*, 18(4):361–367, 1990.

[64] J. M. R. Graham. "Marine CFD final workshop; Status of Programme". Concise summary of all six projects undertaken within the EPSRC funded directed programme into marine CFD, 1999.

[65] R. H. Ni. "A multiple-grid scheme for solving the Euler equations". *AIAA Journal*, 20(11):1565–1571, November 1982. presented as AIAA paper81-1025.

[66] E. Turkel. "Preconditioning techniques in Computational Fluid Dynamics". In *Annual Review of Fluid Mechanics*, volume 31, pages 385–416, 1999.

[67] A. Jameson, W. Schmidt, and E. Turkel. "Numerical solution of the euler equations by finite volume methods using Runge Kutta time stepping schemes". AIAA Paper 81-1259, 1981.

[68] A. L. Gaitonde. "A dual time method for the solution of the unsteady Euler equations". *Aeronautical Journal*, pages 283 – 291, October 1994.

[69] D. L. Whitfield and L. K. Taylor. "Unsteady three-dimensional incompressible Euler and Navier-Stokes solver for stationary and dynamic grids". AIAA Paper 91-1650, June 1991. Presented at Honolulu conference.

[70] T. H. Pappou and S. Tsangaris. "Develpment of an artificial compressibility Methodology using Flux Vector Splitting". *Int. Journal for Num. Meth. in Fluids*, 25:523–545, 1997.

[71] A. M. Wright and S. R. Turnock. "Convective cell approach for solving incompressible Euler flows: Explicit flux vector splitting scheme using artificial compressibility". Ship Science Report 117, Department of Ship Science, University of Southampton, 1999.

[72] W. K. Anderson, J. L. Thomas, and B. Van Leer. "A comparision of finite volume flux vector splitting for the Euler equations". AIAA paper 85-0122, January 1995.

[73] S. R. Turnock. *"Prediction of Ship Rudder-Propeller interaction using parallel computations and wind tunnel tests"*. PhD thesis, University of Southampton, 1993.

[74] N. Gregory and C. L. O'Reilly. "Low speed aerodynamic characteristics of NACA0012 aerofoil section, including the effects of upper-surface roughness simulating hoar frost". Technical Report 1308, National Physics Laboratory: Aerodynamics Division, January 1970.

[75] I. H. Abbot and A. E. Von Doenhoff. *"Theory of wing sections"*. Dover Publications Inc., 1949. ISBN 0-486-60586-8.

[76] Various. "Special section: Credible Computational Fluid Dynamics simulations". *AIAA Journal*, 36(5):665–764, 1998.

[77] P. J. Roache. "Quantification of uncertainty in Computational Fluid Dynamics". In *Annual Review of Fluid Mechanics*, volume 29, pages 123–160, 1997.

[78] D. Thompson, editor. *"The 9th Concise Oxford Dictionary of Current English"*. BCA, by arrangment with Oxford University Press, 1996. ISBN 0-000000-091299.

[79] L. Eça and M. Hoekstra. "On the Numerical Verification of Ship Stern Flow Calculations". In *Technical papers of 1st MARNET CFD Workshop, CIMNE Barcelona*, November 1999.

[80] S. R. Turnock. "Interpretation of CFD results for use in ship hyrdodynamic design". In *Proceedings of The International CFD Conference (CFD 99)*, June 1999. Ulsteinvik, Norway.

[81] N. K. Yamaleev. "Minimization of the Truncation Error by Grid Adaption". ICASE Report 99-46, NASA, November 1999.

[82] F. Stern, R. V. Wilson, H. W. Coleman, and E. G. Paterson. "Verification and Validation of CFD Simulations". In *Proceedings of 3rd ASME/JSME Joint Fluids Engineering Conference*, July 1999. Paper no FEDSM99-6913.

[83] A. Ilinca, R. Camarero, J. Y. Trepanier, and M. Reggio. "Error estimator and adaptive moving grids for finite volumes schemes". *AIAA Journal*, 33(11):2058–2065, November 1995.

[84] J. Oliger and X. Zhu. "Stability and error estimation for component adaptive grid methods". *Applied Numerical Mathematics*, 20:407–426, 1996.

[85] A. Rassineux. "Generation and optimisation of tetrahedral meshes by advancing front technique". *International Journal for numerical Methods in Engineering*, 41:651–674, 1998.

[86] J.J. Bertin and M.L. Smith. *"Aerodynamics for engineers"*. Prentice Hall, 1989. ISBN 0-13-018243-5.

[87] A. Lawal. "Adaptive grid method for convection-diffusion equations". *Int. J. Heat Mass Transfer*, 33(8):1633–1641, August 1990.

[88] Y. Tu and J. F. Thompson. "Three-dimensional solution-adaptive grid generation on composite configurations". *AIAA Journal*, 29(12):2025–2027, December 1991.

[89] J. H. Kwon and H. K. Jeong. "Solution-adaptive grid generation for compressible flow". *Computers and Fluids*, 25(6):551–560, June 1996.

[90] Y. Kallinderis. "A finite volume Navier-Stokes algorithm for adaptive grids". *International Journal of Numerical Methods in fluids*, 15:193–217, 1992.

[91] J. U. Brackbill and J. S. Saltzman. "Adaptive zoning for Singular Problems in Two Dimensions". *Journal of Computational Physics*, 46(2):342–368, 1982.

[92] A. Bowyer. "Computing Dirichlet tessellations". *The Computer Journal*, 24(2):162–166, 1981.

[93] D. F. Watson. "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes". *The Computer Journal*, 24(2):167–172, 1981.

[94] P. R. Eiseman. "Alternating direction adaptive grid generation". *AIAA Journal*, 23(4):551–560, April 1985.

[95] C. Özturan. " Worst case complexity of parallel triangular mesh refinement by longest edge bisection". ICASE Report 96-56, NASA, 1996.

[96] J. Y. Trepanier, M. Reggio, M. Paraschivoiu, and R. Camarero. "Unsteady Euler solutions for arbitary moving bodies and boundaries". *AIAA Journal*, 31(10):1869–1876, October 1993.

[97] C. B. Allen. "Grid adaptation for unsteady flow computations". *Proc Institution of Mechanical Engineers*, 211, 1997.

[98] V. Parthasarathy and Y. Kallinderis. "New multigrid approach for three-dimensional unstructured, adaptive grids". *AIAA Journal*, 32(5):956–963, May 1994.

[99] V. Parthasarathy and Y. Kallinderis. "Directional viscous multigrid using adaptive prismatic meshes". *AIAA Journal*, 33(1):69–78, January 1995.

[100] D. J. Mavriplis. "Multigrid solution strategies for adaptive meshing problems". ICASE Report 95-14, NASA, 1995.

[101] K. Riemslagh and E. Dick. "A multigrid method with unstructured adaptive grids for steady Euler equations". *Journal of Computational and Applied Mathematics*, 67:73–93, 1996.

[102] A. M. Wright. "Automated adaptation of spatial grids for flow solutions around marine bodies of complex geometry", 1999. Submitted for transference from the degree of Master of Philosophy to Doctor of Philosophy.

[103] I. P. Castro and A. G. Robins. "The flow around a surface-mounted cube in uniform and turbulent streams". *Journal of fluid mechanics*, 79:307–335, 1977.

[104] C. F. Ollivier-Gooch. "Multigrid Acceleration of an Upwind Euler Solver on Unstructured Meshes". *AIAA Journal*, 33(10), October 1995.

[105] P.Wesseling. " An introduction to multigrid methods". ICASE Report 95-11, NASA, 1995.

[106] S. Muzaferija. "Advanced techniques in viscous flow CFD". In *Proceedings of 31st WEGEMT School on CFD for ship and offshore design*, May 1999.

A Forms of the Euler Equations

The Euler equations describe the conservation of mass, momentum and energy, with the omission of viscous and thermal conduction effects resulting in the description of what is known as inviscid flow. Three different forms of the equations are commonly used to define the system of equations, revealing their mathematical properties. These properties in turn impose the type of numerical approach that can be used to solve the equations. This section concentrates upon the definition and transformation between these forms of the Euler equations; the conservative, primitive and characteristic forms.

A.1 Conservative Form of Euler Equations

The Euler equations can most easily be defined as a set of conservative laws and as such can be derived using an arbitrary volume, Ω, fixed in space. Variation of a scalar quantity per unit volume U, bounded by a closed surface S can be expressed as a summation of internal and surface sources, and the transport across the bounding surface. The last of these mechanisms is termed a 'flux'.

Due to the removal of viscous terms all diffusive terms are also neglected, leaving the system of equations dominated by the convective transport of mass and momentum.

The mathematical expression of this is

$$\frac{\partial}{\partial t}\int_{\Omega} U d\Omega = -\oint_{S} \vec{F}\cdot d\vec{S} + \int_{\Omega} Q_v d\Omega + \oint_{S} \vec{Q}_S \cdot d\vec{S} \qquad \text{(A.1)}$$

where $\vec{S}$ is outward facing, and Q_V and Q_S are the volume and surface sources respectively, as shown in Figure A.1.

The conservation of mass momentum and energy can be expressed using the general integral conservation law given in Equation A.1, generating a set of five coupled partial differential equations for three dimensional flow. Equation A.2

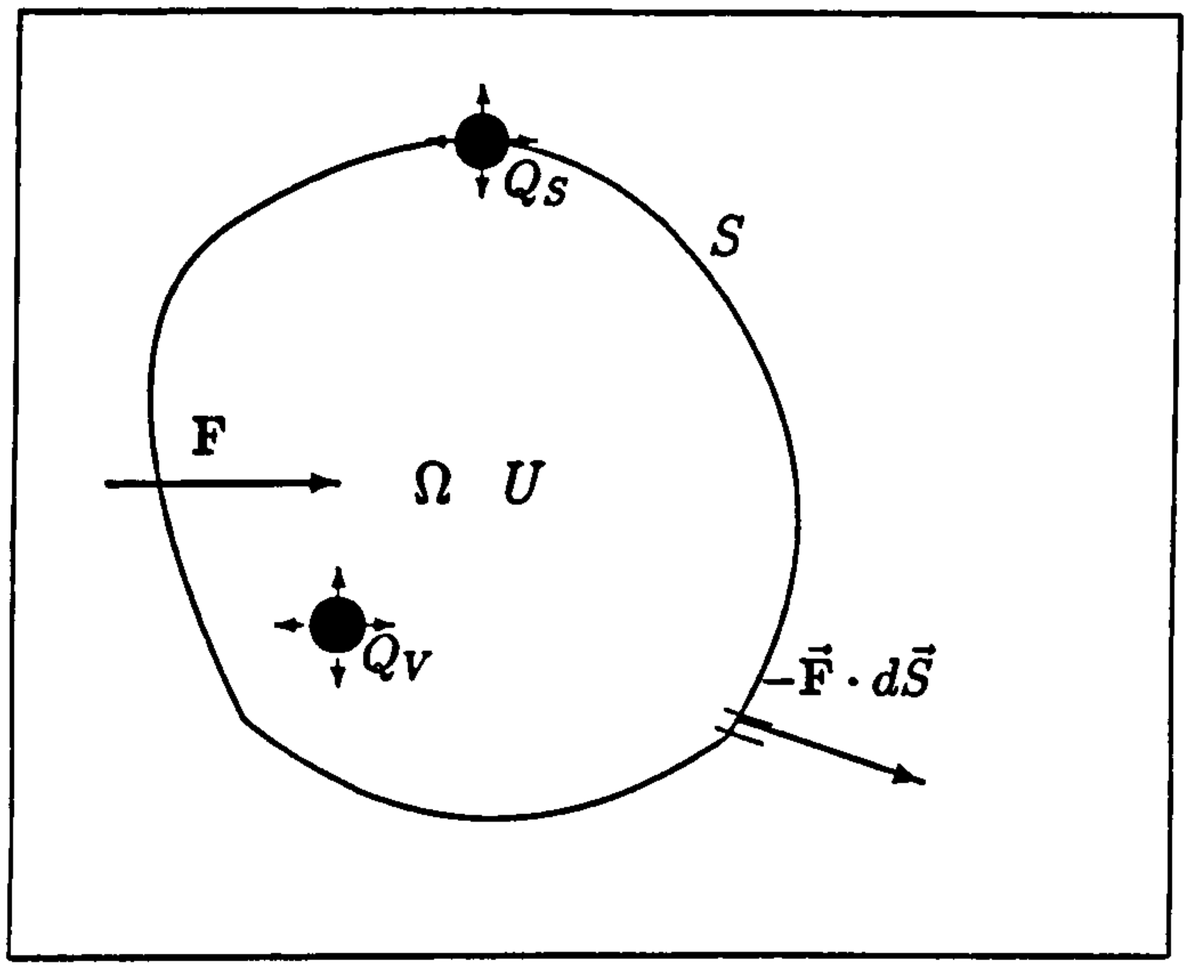


Figure A.1: Arbitrary volume, containing sources and surface fluxes

describes the conservation of mass, where ρ is the density of the fluid and $\vec{v}$ is the velocity vector.

$$\frac{\partial}{\partial t}\int_{\Omega}\rho d\Omega = -\oint_{S}\rho\vec{v}\cdot d\vec{S} \tag{A.2}$$

Equation A.3 defines the conservation of momentum, where $\vec{f_e}$ are external forces acting on the control volume, and p is the static pressure. The conservation of total energy E is defined in Equation A.4 as a summation of a convective energy term and sources which model the work done by the external forces $\vec{f_e}$ and the forces working on the surface of the volume.

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\vec{v}d\Omega + \oint_{S}\rho\vec{v}\left(\vec{v}\cdot d\vec{S}\right) = -\int_{\Omega}\rho\vec{f_e}d\Omega - \oint_{S}pd\vec{S} \tag{A.3}$$

$$\frac{\partial}{\partial t}\int_{\Omega}\rho E d\Omega + \oint_{S}\rho E\vec{v}\cdot d\vec{S} = -\int_{\Omega}\left(\rho\vec{f_e}\cdot\vec{v}\right)d\Omega - \oint_{S}p\vec{v}\cdot d\vec{S} \tag{A.4}$$

Equation A.4 can be simplified by introducing the stagnation enthalpy, as defined in Equation 3.7. Thus the energy conservation law is now expressed in the mode commonly recognised form as given in Equation A.5.

$$\frac{\partial}{\partial t}\int_{\Omega}\rho E d\Omega + \oint_{S}\rho H\vec{v}\cdot d\vec{S} = \int_{\Omega}\left(\rho\vec{f_e}\cdot\vec{v}\right)d\Omega \tag{A.5}$$

These equations can be written in vector form as

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{U}d\Omega + \oint_{S}\mathbf{F}\cdot d\vec{S} = \int_{\Omega}\mathbf{Q}d\Omega \tag{A.6}$$

where **U** is the conservative state vector $\left\{ \begin{array}{c} \rho \\ \rho\vec{v} \\ \rho E \end{array} \right\}$ as defined in Equation 3.2, as are the flux vector $\vec{F}$ and the source vector **Q**.

The result of the control volume analysis is the integral conservative form of the Euler equations. A differential form of the equations may also be found by studying an infinitesimal fluid particle of size dx by dy by dz. Expressions for the conserved variables on each of the faces as well as the change in the variables over the fluid particle may be found. Alternatively the integral form can be manipulated in order to obtain the differential form.

Because the volume Ω is fixed in space the limits of integration in Equation A.6 are constant. This allows the time derivative to be placed inside the integral and for Gauss' theorem to be applied to the surface integral. As a result the integral form of the Euler equations takes the form defined in Equation A.7.

$$\int_\Omega \frac{\partial \mathbf{U}}{\partial t} d\Omega + \int_\Omega \vec{\nabla} \cdot \mathbf{F} d\Omega = \int_\Omega \mathbf{Q} d\Omega$$

$$\rightarrow \int_\Omega \left[\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \mathbf{F} - \mathbf{Q} \right] d\Omega = 0 \qquad \text{(A.7)}$$

It is clear that for the integral to equal zero it must be zero everywhere in the domain due to the arbitrary nature of the volume. Thus Equation A.7 leads to the differential form of the conservative Euler equations -

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \mathbf{F} = \mathbf{Q} \qquad \text{(A.8)}$$

It should be noted that Gauss' theorem implies that the fluid properties are continuous. Real flows however may contain discontinuities such as shock waves and thus the integral form of the conservative Euler equations (Equation A.6) which do not make this assumption is deemed the more fundamentally correct.

A.2 Non-Conservative Form of Euler Equations

The equations derived in Section A.1 are known as the Conservative equations, after the variables in which they are expressed (U). If the same derivation method is followed for a volume travelling with the fluid, such that the mass of fluid within the volume remains constant but the surface area and volume vary, the Non-Conservative form of the Euler equations may be found. These equations are defined in terms of the primitive variables (V).

This form of equations can be reached through direct derivation or by manipulation of the Conservative form. The mass conservation law (Equation A.2) can be written in the differential form as

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \rho \vec{v} = 0 \tag{A.9}$$

Applying the product rule to the dot product term and with the use of the substantial (or total) derivative $\frac{D}{Dt}$ the mass conservation law may be rewritten as

$$\frac{D\rho}{Dt} + \rho \vec{\nabla} \cdot \vec{v} = 0 \tag{A.10}$$

The substantial derivative describes the rate of change as the fluid element moves through space; it sums the local time dependant changes (denoted by $\frac{\partial}{\partial t}$) and the changes that occur because the fluid particle moves around in space. The integral form of the mass conservation law in non-conservative form is defined as

$$\frac{D}{Dt} \int_{\Omega} \rho d\Omega = 0 \tag{A.11}$$

The momentum and energy conservation equations may also be written in non-conservative form as

$$\rho \frac{D\vec{v}}{Dt} = -\vec{\nabla} p + \rho \vec{f_e} \tag{A.12}$$

$$\rho \frac{D}{Dt} \left(e + \frac{\vec{v}^2}{2} \right) = -\vec{\nabla} \cdot (\vec{v} p) + \rho \vec{f_e} \cdot \vec{v} \tag{A.13}$$

As a result the primitive variables, V are defined as -

$$\mathbf{V} = \begin{Bmatrix} \rho \\ \vec{v} \\ p \end{Bmatrix} \tag{A.14}$$

The presence of x, y, and z derivatives on the right hand side of the non-conservative forms results in weak conservation; only single derivatives exist in the conservative forms, thus resulting in strong conservation. Across a discontinuity there exists only small or no changes to the flux variables $\vec{\mathbf{F}}$ whereas the primitive variables such as p are discontinuous. This continuous nature of the fluxes in the conservative form leads to greater stability and hence more common usage in computations of the Euler equations.

A.3 Quasi-linear form of Euler equations

In order to investigate the mathematical properties of the system of Euler equations it is necessary to write these equations in a quasi-linear format. This is because a system of quasi-linear partial differential equations of the first order will be hyperbolic if its homogenous part admits wave-like solutions. The Euler system of equations contains only first-order derivatives and if the external forces $\vec{f_e}$ are independent of the flow gradients the system of Euler equations is of first order in the variables **U**.

The quasi-linear form of Equation A.8 is written as

$$\frac{\partial \mathbf{U}}{\partial t} + \left(\frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{U}}\right) \cdot \vec{\nabla}\mathbf{U} = \mathbf{Q}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\mathbf{A}} \cdot \vec{\nabla}\mathbf{U} = \mathbf{Q} \tag{A.15}$$

or explicitly as

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{U}}{\partial y} + \mathbf{C}\frac{\partial \mathbf{U}}{\partial z} = \mathbf{Q} \tag{A.16}$$

where **A**, **B** and **C** are the three Jacobian matrices of the flux vector $\vec{\mathbf{F}}$. The flux components of the standard Euler equations have the property of being homogenous functions of degree 1 of the conservative variable vector **U** for fluids satisfying the relation

$$p = \rho \mathcal{F}(e) \tag{A.17}$$

This implies that

$$\vec{\mathbf{F}}(\lambda \mathbf{U}) = \lambda \vec{\mathbf{F}}(\mathbf{U}) \quad \text{for any } \lambda \tag{A.18}$$

and by differencing with respect to λ and setting $\lambda = 1$ one obtains the relation

$$\vec{\mathbf{F}}(\mathbf{U}) = \frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{U}}\mathbf{U} = \vec{\mathbf{A}}\mathbf{U} \tag{A.19}$$

If isentropic flow is assumed the Jacobian matrices $\vec{\mathbf{A}}$ may be written in terms of the primitive, or non-conservative, variables, $\mathbf{V}$. The Euler equations are thus expressed as

$$\frac{\partial \mathbf{V}}{\partial t} + \left(\vec{\tilde{\mathbf{A}}} \cdot \vec{\nabla}\right) \mathbf{V} = 0 \tag{A.20}$$

In order to gain further insight into the mathematical properties of the equations it is necessary to find the eigenvalues of the Jacobians. Using Cramer's rule, the eigenvalues are found from

$$|\vec{\mathbf{A}} - \lambda \mathbf{I}| = 0 \tag{A.21}$$

where $\mathbf{I}$ is the identity matrix and λ is defined as an eigenvalue of the matrix $\vec{\mathbf{A}}$.

It should be noted that for the standard Euler equations it is easier to obtain the eigenvalues of the system when they are written in non-conservative form, as a function of $\mathbf{V}$ due to the much simpler structure of the Jacobians.

A.4 Characteristic Form of Euler Equations

If all eigenvalues are distinct and real then it can be concluded that the system of equations are hyperbolic, and exhibit wave-like solutions. An extension of this concept is to think of the equations as a series of waves propagating properties throughout the time-space domain. Further to this, the waves can be described as lines or surfaces along which certain properties remain constant. The domain on one side of the line/surface represents the region already affected by the wave, while the region on the other side of the line/surface represents the domain which is as yet non affected. These are the zones of 'influence' and 'dependence' respectively. The lines/surfaces are known as 'characteristic lines', the slopes of which are given by the eigenvalues of the Jacobians.

By directionalising the three dimensional equations using an arbitrary direction $\vec{k}$, the properties in three dimensions may be examined. The Jacobian matrix corresponding to this direction can be written as

$$\mathbf{K} = \vec{\mathbf{A}} \cdot \vec{k} \tag{A.22}$$

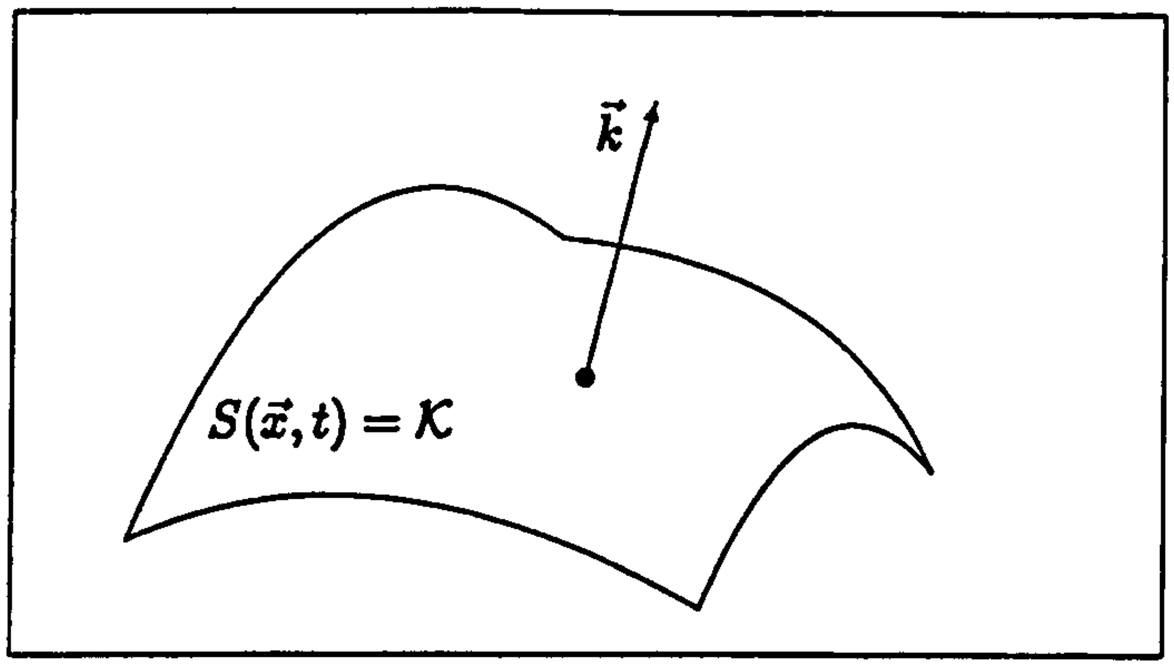


Figure A.2: Characteristic surface

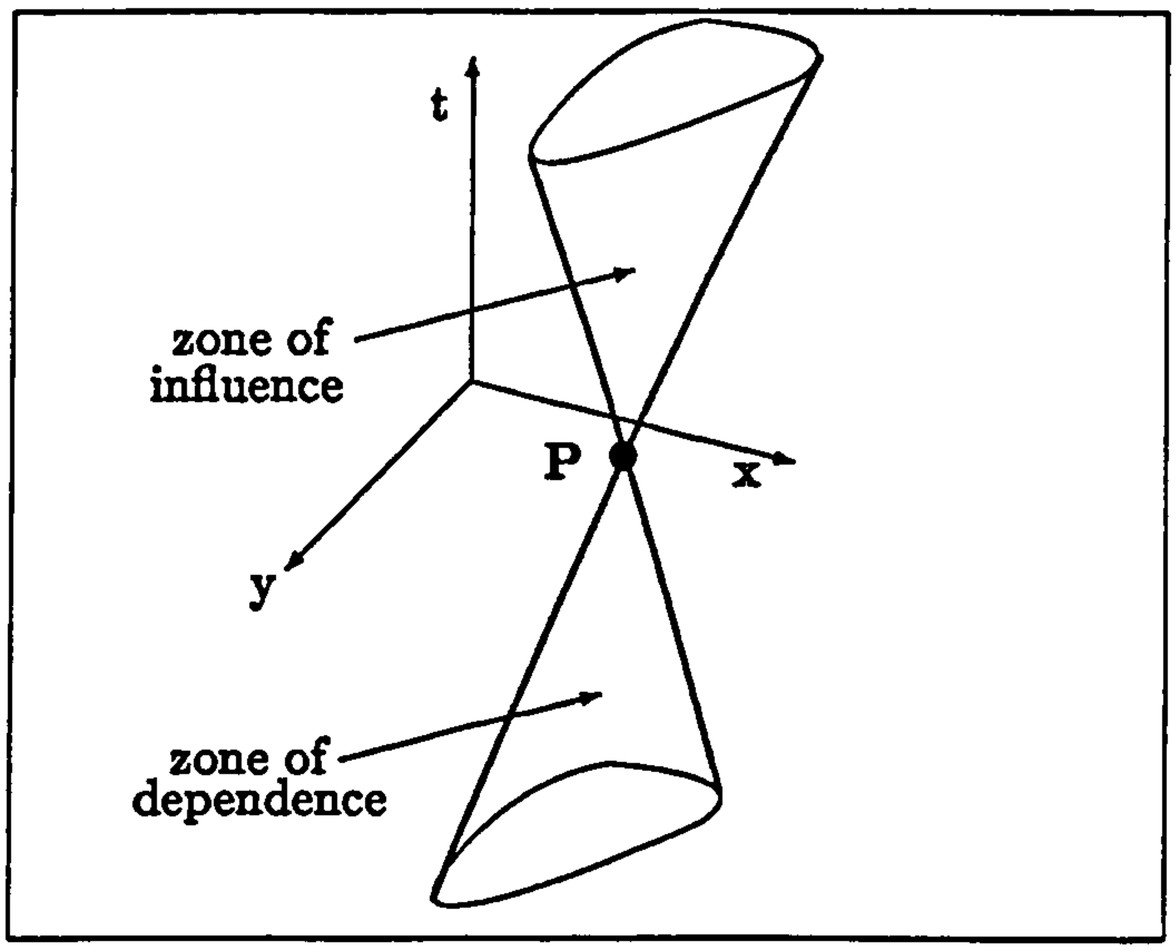


Figure A.3: Zones of dependence and influence

If $\vec{k}$ is the unit vector in any direction, the eigenvalues give the speed of propagation in the physical plane and the slopes of the characteristic surfaces $S(\vec{x},t) = constant$ in the $(\vec{x},t)$ plane, shown in Figure A.2. Since $\vec{k}$ is arbitrary an infinite number of characteristic surfaces can be found, the envelope of which produce the 'Mach' conoid defining the zones of dependence and influence at a point in the $(\vec{x},t)$ plane.

The existence of these characteristic surfaces allows the system of equations to be reformulated such that they only contain derivatives along the surfaces, since the behaviour of the system is dictated by the properties which are propagated with the surfaces. The advantage to this formulation is that the number of spatial dimensions in which the equations are written reduces by one. The

re-formulation results in a form containing only derivatives in directions lying on the surface, and can be constructed from a linear combination of the original equations. The transformation equation for λ_i, known as the compatibility relationship, can be written as

$$\tilde{l}^i \frac{\partial \mathbf{V}}{\partial t} + \tilde{l}^i \left(\vec{\tilde{\mathbf{A}}} \cdot \vec{\nabla} \right) \mathbf{V} = \tilde{l}^i \tilde{\mathbf{Q}} \tag{A.23}$$

where l^i, the arbitrary coefficients of the linear combination, are found to be the left eigenvectors of the matrix $\mathbf{K}$.

A.5 Transformation Between Forms

The Jacobian matrix of the transformation from the conservative to the non-conservative variables is defined as

$$\mathbf{M} = \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \tag{A.24}$$

and its evaluation requires the explicit formulation of the fluid constitutive relations. An important point is that the definition of the non-conservative Jacobians does not require an explicit definition of the fluid constitutive relations, and therefore has a larger validity range; that is they are not necessarily connected to a perfect gas assumption as is the case for the conservative Jacobians.

The relations between the conservative and the non-conservative Jacobians $\vec{\mathbf{A}}$ and $\vec{\tilde{\mathbf{A}}}$ can be expressed through a similar transformation with matrix $\mathbf{M}$. Applying Equation A.24 to Equation A.15 produces

$$\mathbf{M} \frac{\partial \mathbf{V}}{\partial t} + \vec{\mathbf{A}} \mathbf{M} \cdot \vec{\nabla} \mathbf{V} = \mathbf{Q} \tag{A.25}$$

Identifying with the non-conservative form, Equation A.20 gives the relation

$$\vec{\tilde{\mathbf{A}}} = \mathbf{M}^{-1} \vec{\mathbf{A}} \mathbf{M} \quad \text{or} \quad \vec{\mathbf{A}} = \mathbf{M} \vec{\tilde{\mathbf{A}}} \mathbf{M}^{-1} \tag{A.26}$$

It can be seen from Equation A.23 that a matrix $\mathbf{L}^{-1}$ can be constructed from the left eigenvectors $\tilde{l}^i$. That is, the i^{th} line of $\mathbf{L}^{-1}$ is the left eigenvector $\tilde{l}^i$. Thus, grouping all the eigenvalues together leads to

$$\mathbf{L}^{-1} \tilde{\mathbf{K}} = \Lambda \mathbf{L}^{-1} \tag{A.27}$$

Since the matrix $\tilde{\mathbf{K}}$ is not symmetric there exists a set of right eigenvectors $\tilde{r}^j$ associated with the eigenvalues λ^j. These column vectors are defined by

$$\tilde{\mathbf{K}}\tilde{r}^j = \lambda_j \tilde{r}^j \tag{A.28}$$

The matrix of right eigenvectors is the inverse of the matrix $\mathbf{L}^{-1}$ of the left eigenvectors.

With the introduction of the matrices $\mathbf{L}$ and $\mathbf{L}^{-1}$ one can write the compatibility equations in a compact form

$$\left(\mathbf{L}^{-1}\partial_t + \mathbf{L}^{-1}\vec{\tilde{\mathbf{A}}} \cdot \vec{\nabla}\right)\mathbf{V} = \mathbf{L}^{-1}\tilde{\mathbf{Q}} \tag{A.29}$$

The matrix $\mathbf{P}$ defined by

$$\mathbf{P}^{-1} = \mathbf{L}^{-1}\mathbf{M}^{-1} \quad \text{or} \quad \mathbf{P} = \mathbf{M}\mathbf{L} \tag{A.30}$$

plays the same role with respect to the conservative variables as the matrix $\mathbf{L}$ with the primitive variables.

The compatibility relations (Equation A.23 or Equation A.29) lead to the introduction of the characteristic variables $\mathbf{W}$. They are defined as a column vector by the relation valid for arbitrary variations δ

$$\begin{aligned} \delta\mathbf{W} &= \mathbf{L}^{-1}\delta\mathbf{V} \\ \delta\mathbf{V} &= \mathbf{L}\delta\mathbf{W} \end{aligned} \tag{A.31}$$

and

$$\begin{aligned} \delta\mathbf{W} &= \mathbf{P}^{-1}\delta\mathbf{U} \\ \delta\mathbf{U} &= \mathbf{P}\delta\mathbf{W} \end{aligned} \tag{A.32}$$

The relationship between the three sets of variables can be summarised as shown in Figure A.4

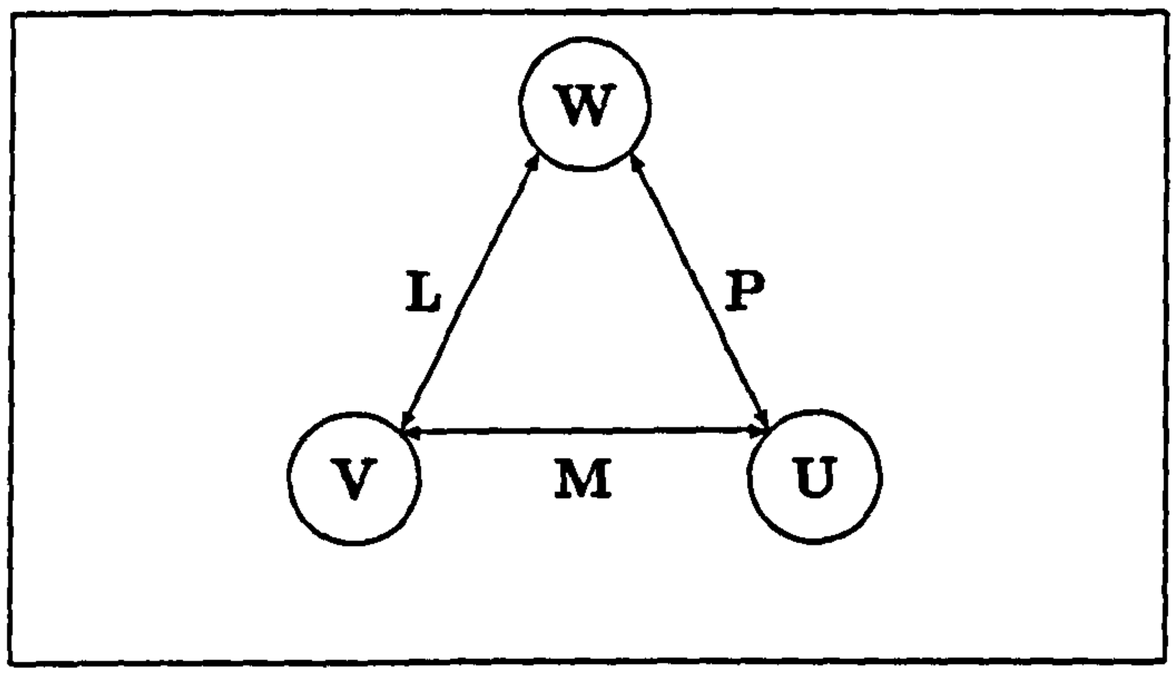


Figure A.4: Relation between the conservative, primitive and characteristic variables

B Derivation of Characteristics of Euler Equations

To gain directionality in the conservative form of the Euler equations the eigenvalues must be incorporated in the Jacobian via the transformation matrices between the characteristic and conservative forms. If $\mathbf{P} = \frac{\partial \mathbf{U}}{\partial \mathbf{W}}$, and $\mathbf{K}$ is taken to be the generalised Jacobian of the flux components for the conservative form of the Euler equations (i.e. $\mathbf{K} = n_x A + n_y B + n_z C$ where $A = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ etc.), then -

$$\begin{aligned}
\mathbf{W_t} + \mathbf{\Lambda W_x} &= 0 \\
\Rightarrow (\mathbf{P^{-1}U_t}) + \mathbf{\Lambda}(\mathbf{P^{-1}U_x}) &= 0 \\
\Rightarrow \mathbf{P P^{-1} U_t} + \mathbf{P \Lambda P^{-1} U_x} &= 0 \\
\Rightarrow \mathbf{U_t} + (\mathbf{P \Lambda P^{-1}})\mathbf{U_x} &= 0
\end{aligned}$$

$$\Rightarrow \mathbf{K} = \mathbf{P \Lambda P^{-1}} \tag{B.1}$$

It should be noted that the system of equations maintains hyperbolicity if the eigenvalues are real and the norms of the transformation matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ are uniformly bounded for arbitrary real n_x, n_y and n_z.

The matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ can be gained from the asymmetry of the $\mathbf{K}$ matrix; there exists a set of right eigenvectors and a set of left eigenvectors (r^j and l^j respectively) of $\mathbf{K}$ such that the rows of $\mathbf{P^{-1}}$ are equal to the left eigenvectors and the columns of $\mathbf{P}$ are equal to the right eigenvectors.

$$\begin{aligned}
i.e.\ (l_i)^j \mathbf{K}_{ik} &= \lambda_j (l_k)^j \ , \ summation\ in\ i \\
\text{and} \quad \mathbf{K}_{ik} (r_k)^j &= \lambda_j (r_i)^j \ , \ summation\ in\ k
\end{aligned} \tag{B.2}$$

From this an upwind formulation can be obtained with the Jacobians

$$\begin{aligned} \mathbf{K}^{+} &= \mathbf{P}\boldsymbol{\Lambda}^{+}\mathbf{P}^{-1} \\ \mathbf{K}^{-} &= \mathbf{P}\boldsymbol{\Lambda}^{-}\mathbf{P}^{-1} \\ \textit{with}\ \mathbf{K} &= \mathbf{K}^{+} + \mathbf{K}^{-} \end{aligned} \tag{B.3}$$

Hence one has

$$\mathbf{F} = \mathbf{K}^{+}\mathbf{U} + \mathbf{K}^{-}\mathbf{U} = \mathbf{F}^{+} + \mathbf{F}^{-} \tag{B.4}$$

B.1 Artificial Compressibility Formulation

The state vector **U** and the flux vector, $\vec{F}$ = **(f, g, h)** are taken as -

$$\mathbf{U} = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} \mathbf{c}^2 u \\ u^2 + p \\ uv \\ uw \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} \mathbf{c}^2 v \\ uv \\ v^2 + p \\ vw \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} \mathbf{c}^2 w \\ uw \\ vw \\ w^2 + p \end{bmatrix}$$

Definition of generalised Jacobian matrix, **K**;

$$\mathbf{K} = n_x A + n_y B + n_z C \tag{B.5}$$

where

$$A = \frac{\partial \mathbf{F}}{\partial \mathbf{U}} \quad B = \frac{\partial \mathbf{G}}{\partial \mathbf{U}} \quad C = \frac{\partial \mathbf{H}}{\partial \mathbf{U}} \tag{B.6}$$

Thus

$$\mathbf{A} = \begin{bmatrix} \frac{\partial \mathbf{F}_1}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{F}_1}{\partial \mathbf{U}_2} & \frac{\partial \mathbf{F}_1}{\partial \mathbf{U}_3} & \frac{\partial \mathbf{F}_1}{\partial \mathbf{U}_4} \\ \frac{\partial \mathbf{F}_2}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{F}_2}{\partial \mathbf{U}_2} & \frac{\partial \mathbf{F}_2}{\partial \mathbf{U}_3} & \frac{\partial \mathbf{F}_2}{\partial \mathbf{U}_4} \\ \frac{\partial \mathbf{F}_3}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{F}_3}{\partial \mathbf{U}_2} & \frac{\partial \mathbf{F}_3}{\partial \mathbf{U}_3} & \frac{\partial \mathbf{F}_3}{\partial \mathbf{U}_4} \\ \frac{\partial \mathbf{F}_4}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{F}_4}{\partial \mathbf{U}_2} & \frac{\partial \mathbf{F}_4}{\partial \mathbf{U}_3} & \frac{\partial \mathbf{F}_4}{\partial \mathbf{U}_4} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{c}^2 & 0 & 0 \\ 1 & 2u & 0 & 0 \\ 0 & v & u & 0 \\ 0 & w & 0 & u \end{bmatrix} \tag{B.7}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial \mathbf{G}_1}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{G}_1}{\partial \mathbf{U}_2} & \frac{\partial \mathbf{G}_1}{\partial \mathbf{U}_3} & \frac{\partial \mathbf{G}_1}{\partial \mathbf{U}_4} \\ \frac{\partial \mathbf{G}_2}{\partial \mathbf{U}_1} & \ddots & & \vdots \\ \frac{\partial \mathbf{G}_3}{\partial \mathbf{U}_1} & & \ddots & \vdots \\ \frac{\partial \mathbf{G}_4}{\partial \mathbf{U}_1} & \cdots & \cdots & \frac{\partial \mathbf{G}_4}{\partial \mathbf{U}_4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \mathbf{c}^2 & 0 \\ 0 & v & u & 0 \\ 1 & 0 & 2v & 0 \\ 0 & 0 & w & v \end{bmatrix} \tag{B.8}$$

$$\mathbf{C} = \begin{bmatrix} \frac{\partial \mathbf{H}_1}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{H}_1}{\partial \mathbf{U}_2} & \frac{\partial \mathbf{H}_1}{\partial \mathbf{U}_3} & \frac{\partial \mathbf{H}_1}{\partial \mathbf{U}_4} \\ \frac{\partial \mathbf{H}_2}{\partial \mathbf{U}_1} & \ddots & & \vdots \\ \frac{\partial \mathbf{H}_3}{\partial \mathbf{U}_1} & & \ddots & \vdots \\ \frac{\partial \mathbf{H}_4}{\partial \mathbf{U}_1} & \cdots & \cdots & \frac{\partial \mathbf{H}_4}{\partial \mathbf{U}_4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \mathbf{c}^2 \\ 0 & w & 0 & u \\ 0 & 0 & w & v \\ 1 & 0 & 0 & 2w \end{bmatrix} \tag{B.9}$$

Therefore -

$$\mathbf{K} = \begin{bmatrix} 0 & \mathbf{c}^2 n_x & \mathbf{c}^2 n_y & \mathbf{c}^2 n_z \\ n_x & \vec{v}\vec{n} + u n_x & u n_y & u n_z \\ n_y & v n_x & \vec{v}\vec{n} + v n_y & v n_z \\ n_z & w n_x & w n_y & \vec{v}\vec{n} + w n_z \end{bmatrix} \tag{B.10}$$

Eigenvalues of this generalised matrix can be found from the solution of $det|\mathbf{K} - \lambda\mathbf{I}| = 0$ or by solving for the eigenvalues of **A**, **B** and **C** separately, and then combining the results. The second of these methods has fewer terms and allows the eigenvalues of one dimensional and two dimensional flows to be selected; hence it is this method that is followed.

$$det|\mathbf{K} - \lambda\mathbf{I}| = 0$$

$$\Rightarrow \begin{vmatrix} -\lambda & \mathbf{c}^2 n_x & \mathbf{c}^2 n_y & \mathbf{c}^2 n_z \\ n_x & \vec{v}\vec{n} + u n_x - \lambda & u n_y & u n_z \\ n_y & v n_x & \vec{v}\vec{n} + v n_y - \lambda & v n_z \\ n_z & w n_x & w n_y & \vec{v}\vec{n} + w n_z - \lambda \end{vmatrix} = 0 \tag{B.11}$$

Separating into component matrices **A**, **B** and **C** and solving for their eigenvalues gives

$$det|\mathbf{A} - \lambda\mathbf{I}| = 0$$

$$\Rightarrow \begin{vmatrix} -\lambda_x & \mathbf{c}^2 & 0 & 0 \\ 1 & 2u - \lambda_x & 0 & 0 \\ 0 & v & u - \lambda_x & 0 \\ 0 & w & 0 & u - \lambda_x \end{vmatrix} = 0$$

$$\Rightarrow \frac{1}{\lambda_x^2}\begin{vmatrix} \begin{vmatrix} -\lambda_x & \mathbf{c}^2 \\ 1 & 2u-\lambda_x \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 1 & 0 \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 1 & 0 \end{vmatrix} \\ \begin{vmatrix} -\lambda_x & \mathbf{c}^2 \\ 1 & v \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & u-\lambda_x \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} -\lambda_x & \mathbf{c}^2 \\ 0 & w \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & u-\lambda_x \end{vmatrix} \end{vmatrix} = 0$$

$$\Rightarrow \begin{vmatrix} \frac{\lambda_x^2-2u\lambda_x+\mathbf{c}^2}{\lambda_x} & 0 & 0 \\ -\frac{v}{\lambda_x} & -\frac{(u-\lambda_x)}{\lambda_x} & 0 \\ -\frac{w}{\lambda_x} & 0 & -\frac{(u-\lambda_x)}{\lambda_x} \end{vmatrix} = 0$$

$$\Rightarrow \frac{\lambda_x^2-2u\lambda_x+\mathbf{c}^2}{\lambda_x^2}\left(\frac{(\lambda_x-u)(\lambda_x-u)}{\lambda_x^2}-0\right)=0$$

$$\Rightarrow \left(\lambda_x^2-2u\lambda_x+\mathbf{c}^2\right)(\lambda_x-u)(\lambda_x-u)=0$$

$$\lambda_{x1}=u \qquad \text{(2.12-a)}$$

$$\lambda_{x2}=u \qquad \text{(2.12-b)}$$

$$\lambda_{x3}=u+\sqrt{u^2+\mathbf{c}^2} \qquad \text{(2.12-c)}$$

$$\lambda_{x4}=u-\sqrt{u^2+\mathbf{c}^2} \qquad \text{(2.12-d)}$$

Similarly for the solutions of $det|\mathbf{B}-\lambda\mathbf{I}|=0$ and $det|\mathbf{C}-\lambda\mathbf{I}|=0$, the eigenvalues for the y and z directions are

$$\lambda_{y1}=v \qquad \text{(2.13-a)}$$

$$\lambda_{y2}=v \qquad \text{(2.13-b)}$$

$$\lambda_{y3}=v+\sqrt{v^2+\mathbf{c}^2} \qquad \text{(2.13-c)}$$

$$\lambda_{y4}=v-\sqrt{v^2+\mathbf{c}^2} \qquad \text{(2.13-d)}$$

$$\lambda_{z1}=w \qquad \text{(2.14-a)}$$

$$\lambda_{z2}=w \qquad \text{(2.14-b)}$$

$$\lambda_{z3}=w+\sqrt{w^2+\mathbf{c}^2} \qquad \text{(2.14-c)}$$

$$\lambda_{z4}=w-\sqrt{w^2+\mathbf{c}^2} \qquad \text{(2.14-d)}$$

These results combine to give -

$$\lambda_1 = un_x + vn_y + wn_z$$
$$= \vec{v}\vec{n} \qquad (2.15\text{-a})$$
$$\lambda_2 = un_x + vn_y + wn_z$$
$$= \vec{v}\vec{n} \qquad (2.15\text{-b})$$
$$\lambda_3 = un_x + vn_y + wn_z + \sqrt{(un_x)^2 + (vn_y)^2 + (wn_z)^2 + \mathrm{c}^2(n_x^2 + n_y^2 + n_z^2)}$$
$$= \vec{v}\vec{n} + a \qquad (2.15\text{-c})$$
$$\lambda_4 = un_x + vn_y + wn_z - \sqrt{(un_x)^2 + (vn_y)^2 + (wn_z)^2 + \mathrm{c}^2(n_x^2 + n_y^2 + n_z^2)}$$
$$= \vec{v}\vec{n} - a \qquad (2.15\text{-d})$$

$$\mathbf{\Lambda} = \begin{bmatrix} \vec{v}\vec{n} & -- & -- & -- \\ -- & \vec{v}\vec{n} & -- & -- \\ -- & -- & \vec{v}\vec{n} + a & -- \\ -- & -- & -- & \vec{v}\vec{n} - a \end{bmatrix} \qquad (2.16)$$

$$where\ a^2 = \vec{v}\vec{n}^2 + \mathrm{c}^2(n_x^2 + n_y^2 + n_z^2). \qquad (2.17)$$

As stated by Equation (B.2), the matrices $\mathbf{P}$ and $\mathbf{P}^{-1}$ can be gained from these eigenvalues combined with the matrix $\mathbf{K}$.

Using the right eigenvectors of $\mathbf{K}$ -

$$\mathbf{K}_{ik}(r_k)^j = \lambda_j(r_i)^j \ ,\ summation\ in\ k$$

$$j = 1,2\ , \quad \mathrm{c}^2 n_x r_2 + \mathrm{c}^2 n_y r_3 + \mathrm{c}^2 n_z r_4 = \vec{v}\vec{n} r_1$$
$$n_x r_1 + u(n_x r_2 + n_y r_3 + n_z r_4) = 0$$
$$n_y r_1 + v(n_x r_2 + n_y r_3 + n_z r_4) = 0$$
$$n_z r_1 + w(n_x r_2 + n_y r_3 + n_z r_4) = 0$$
$$\Rightarrow \vec{n} r_1 + \vec{v}(n_x r_2 + n_y r_3 + n_z r_4) = 0$$

$$\Rightarrow r_1 \equiv 0$$
$$and\ (n_x r_2 + n_y r_3 + n_z r_4) \equiv 0$$

$$Taking\ r_3 = 0,\ r_4 = -\frac{n_x}{n_z} r_2$$
$$Taking\ r_4 = 0,\ r_3 = -\frac{n_x}{n_y} r_2$$

There is of course the third option of taking $r_2 = 0$, but as will be shown later, the choice of which two eigenvector components are taken to be zero is irrelevant.

$$\begin{aligned} j = 3\ , \quad \mathbf{c}^2 n_x r_2 + \mathbf{c}^2 n_y r_3 + \mathbf{c}^2 n_z r_4 &= (\vec{v}\vec{n} + a) r_1 \\ n_x r_1 + u(n_x r_2 + n_y r_3 + n_z r_4) &= a r_2 \\ n_y r_1 + v(n_x r_2 + n_y r_3 + n_z r_4) &= a r_3 \\ n_z r_1 + w(n_x r_2 + n_y r_3 + n_z r_4) &= a r_4 \end{aligned}$$

if r_1 *is made arbitrary (the norm of* $\vec{n}$ *doesn't matter),*

$$\begin{aligned} \Rightarrow r_1 &= \mathbf{c}^2 a \\ \Rightarrow r_2 &= \mathbf{c}^2 n_x + u(\vec{v}\vec{n} + a) \\ \Rightarrow r_3 &= \mathbf{c}^2 n_y + v(\vec{v}\vec{n} + a) \\ \Rightarrow r_4 &= \mathbf{c}^2 n_z + w(\vec{v}\vec{n} + a) \end{aligned}$$

$$\begin{aligned} j = 4\ , \quad \mathbf{c}^2 n_x r_2 + \mathbf{c}^2 n_y r_3 + \mathbf{c}^2 n_z r_4 &= (\vec{v}\vec{n} - a) r_1 \\ n_x r_1 + u(n_x r_2 + n_y r_3 + n_z r_4) &= -a r_2 \\ n_y r_1 + v(n_x r_2 + n_y r_3 + n_z r_4) &= -a r_3 \\ n_z r_1 + w(n_x r_2 + n_y r_3 + n_z r_4) &= -a r_4 \end{aligned}$$

if r_1 *is made arbitrary again,*

$$\begin{aligned} \Rightarrow r_1 &= -\mathbf{c}^2 a \\ \Rightarrow r_2 &= \mathbf{c}^2 n_x + u(\vec{v}\vec{n} - a) \\ \Rightarrow r_3 &= \mathbf{c}^2 n_y + v(\vec{v}\vec{n} - a) \\ \Rightarrow r_4 &= \mathbf{c}^2 n_z + w(\vec{v}\vec{n} - a) \end{aligned}$$

Therefore,

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & \mathbf{c}^2 a & -\mathbf{c}^2 a \\ -n_z & -n_y & u(\vec{v}\vec{n}+a)+\mathbf{c}^2 n_x & u(\vec{v}\vec{n}-a)+\mathbf{c}^2 n_x \\ 0 & n_x & v(\vec{v}\vec{n}+a)+\mathbf{c}^2 n_y & v(\vec{v}\vec{n}-a)+\mathbf{c}^2 n_y \\ n_x & 0 & w(\vec{v}\vec{n}+a)+\mathbf{c}^2 n_z & w(\vec{v}\vec{n}-a)+\mathbf{c}^2 n_z \end{bmatrix} \tag{2.18}$$

From which its inverse can be found -

$$\mathbf{P^{-1}} = \begin{bmatrix} \frac{n_z \vec{v}\vec{n} - w\vec{n}^2}{n_x a^2} & -\frac{w\vec{v}\vec{n}+n_z c^2}{a^2} & -\frac{n_y(w\vec{v}\vec{n}+n_z c^2)}{n_x a^2} & \frac{(un_x+vn_y)\vec{v}\vec{n}+(n_x^2+n_y^2)c^2}{n_x a^2} \\ \frac{n_y \vec{v}\vec{n} - v\vec{n}^2}{n_x a^2} & -\frac{v\vec{v}\vec{n}+n_y c^2}{a^2} & \frac{(un_x+wn_z)\vec{v}\vec{n}+(n_x^2+n_z^2)c^2}{n_x a^2} & -\frac{n_z(v\vec{v}\vec{n}+n_y c^2)}{n_x a^2} \\ -\frac{\vec{v}\vec{n}-a}{2a^2 c^2} & \frac{n_x}{2a^2} & \frac{n_y}{2a^2} & \frac{n_z}{2a^2} \\ -\frac{\vec{v}\vec{n}+a}{2a^2 c^2} & \frac{n_x}{2a^2} & \frac{n_y}{2a^2} & \frac{n_z}{2a^2} \end{bmatrix} \tag{2.19}$$

It should be noted at this point that if the eigenvector components r_2 and r_4 were taken to be zero during the calculation of the first and second column of the matrix $\mathbf{P}$ then a $\frac{1}{n_y}$ term would appear in the $\mathbf{P^{-1}}$ matrix instead of $\frac{1}{n_x}$. The process is the same for r_2 and r_3 with n_z respectively. The choice of which two components are reduced to zero is irrelevant due to the multiplication of $\mathbf{P}$ and $\mathbf{P^{-1}}$ during the calculation of $\mathbf{K}$ and $\mathbf{F}$.

By equations (B.1), (B.4),

$$\begin{aligned} \mathbf{F^{\pm}} &= \mathbf{K^{\pm}U} \\ &= \mathbf{P\Lambda^{\pm}P^{-1}U} \end{aligned}$$

All of these matrices are now known. Thus $\mathbf{K^{\pm}}$ and $\mathbf{F^{\pm}}$ can be defined as shown in Equations 2.20 and 2.21 respectively.

$$
\mathbf{K}^{\pm} = \begin{bmatrix}
\frac{1}{2a}\begin{Bmatrix} -\lambda_3^{\pm}(\vec{v}\vec{n}-a) \\ +\lambda_4^{\pm}(\vec{v}\vec{n}+a) \end{Bmatrix} & \frac{c^2}{2a}n_x(\lambda_3^{\pm}-\lambda_4^{\pm}) & \frac{c^2}{2a}n_y(\lambda_3^{\pm}-\lambda_4^{\pm}) & \frac{c^2}{2a}n_z(\lambda_3^{\pm}-\lambda_4^{\pm}) \\
\frac{1}{a^2}\begin{Bmatrix} \lambda_1(n_x\vec{v}\vec{n}-u) \\ -\frac{1}{2c^2}\lambda_3(\vec{v}\vec{n}-a)\begin{pmatrix} u(\vec{v}\vec{n}+a) \\ +n_xc^2 \end{pmatrix} \\ -\frac{1}{2c^2}\lambda_4(\vec{v}\vec{n}+a)\begin{pmatrix} u(\vec{v}\vec{n}-a) \\ +n_xc^2 \end{pmatrix} \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} \lambda_1\begin{pmatrix} (vn_y+wn_z)\vec{v}\vec{n} \\ +(n_y^2+n_z^2)c^2 \end{pmatrix} \\ +\frac{n_x}{2}\lambda_3(u(\vec{v}\vec{n}+a)+n_xc^2) \\ +\frac{n_x}{2}\lambda_4(u(\vec{v}\vec{n}-a)+n_xc^2) \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} -\lambda_1n_y(u\vec{v}\vec{n}+n_xc^2) \\ +\frac{n_y}{2}\lambda_3(u(\vec{v}\vec{n}+a)+n_xc^2) \\ +\frac{n_y}{2}\lambda_4(u(\vec{v}\vec{n}-a)+n_xc^2) \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} -\lambda_1n_z(u\vec{v}\vec{n}+n_xc^2) \\ +\frac{n_z}{2}\lambda_3(u(\vec{v}\vec{n}+a)+n_xc^2) \\ +\frac{n_z}{2}\lambda_4(u(\vec{v}\vec{n}-a)+n_xc^2) \end{Bmatrix} \\
\frac{1}{a^2}\begin{Bmatrix} \lambda_1(n_y\vec{v}\vec{n}-v) \\ -\frac{1}{2c^2}\lambda_3(\vec{v}\vec{n}-a)\begin{pmatrix} v(\vec{v}\vec{n}+a) \\ +n_yc^2 \end{pmatrix} \\ -\frac{1}{2c^2}\lambda_4(\vec{v}\vec{n}+a)\begin{pmatrix} v(\vec{v}\vec{n}-a) \\ +n_yc^2 \end{pmatrix} \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} -\lambda_1n_x(v\vec{v}\vec{n}+n_yc^2) \\ +\frac{n_x}{2}\lambda_3(v(\vec{v}\vec{n}+a)+n_yc^2) \\ +\frac{n_x}{2}\lambda_4(v(\vec{v}\vec{n}-a)+n_yc^2) \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} \lambda_1\begin{pmatrix} (un_x+wn_z)\vec{v}\vec{n} \\ +(n_x^2+n_z^2)c^2 \end{pmatrix} \\ +\frac{n_y}{2}\lambda_3(v(\vec{v}\vec{n}+a)+n_yc^2) \\ +\frac{n_y}{2}\lambda_4(v(\vec{v}\vec{n}-a)+n_yc^2) \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} -\lambda_1n_z(v\vec{v}\vec{n}+n_yc^2) \\ +\frac{n_z}{2}\lambda_3(v(\vec{v}\vec{n}+a)+n_yc^2) \\ +\frac{n_z}{2}\lambda_4(v(\vec{v}\vec{n}-a)+n_yc^2) \end{Bmatrix} \\
\frac{1}{a^2}\begin{Bmatrix} \lambda_1(n_z\vec{v}\vec{n}-w) \\ -\frac{1}{2c^2}\lambda_3(\vec{v}\vec{n}-a)\begin{pmatrix} w(\vec{v}\vec{n}+a) \\ +n_zc^2 \end{pmatrix} \\ -\frac{1}{2c^2}\lambda_4(\vec{v}\vec{n}+a)\begin{pmatrix} w(\vec{v}\vec{n}-a) \\ +n_zc^2 \end{pmatrix} \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} -\lambda_1n_x(w\vec{v}\vec{n}+n_zc^2) \\ +\frac{n_x}{2}\lambda_3(w(\vec{v}\vec{n}+a)+n_zc^2) \\ +\frac{n_x}{2}\lambda_4(w(\vec{v}\vec{n}-a)+n_zc^2) \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} -\lambda_1n_y(w\vec{v}\vec{n}+n_zc^2) \\ +\frac{n_y}{2}\lambda_3(w(\vec{v}\vec{n}+a)+n_zc^2) \\ +\frac{n_y}{2}\lambda_4(w(\vec{v}\vec{n}-a)+n_zc^2) \end{Bmatrix} & \frac{1}{a^2}\begin{Bmatrix} \lambda_1\begin{pmatrix} (un_x+vn_y)\vec{v}\vec{n} \\ +(n_x^2+n_y^2)c^2 \end{pmatrix} \\ +\frac{n_z}{2}\lambda_3(w(\vec{v}\vec{n}+a)+n_zc^2) \\ +\frac{n_z}{2}\lambda_4(w(\vec{v}\vec{n}-a)+n_zc^2) \end{Bmatrix}
\end{bmatrix}
\tag{2.20}
$$

$$
\mathbf{F}^{\pm} = \begin{bmatrix}
\frac{p}{2a}\left\{\lambda_4^{\pm}(\vec{v}\vec{n}+a)-\lambda_3^{\pm}(\vec{v}\vec{n}-a)\right\}+\frac{c^2\vec{v}\vec{n}}{2a}\left\{\lambda_3^{\pm}-\lambda_4^{\pm}\right\} \\
\\
\frac{p}{a^2}\left\{\lambda_1^{\pm}(n_x\vec{v}\vec{n}-u\vec{n}^2)\right. \\
-\frac{\lambda_3^{\pm}}{2c^2}(\vec{v}\vec{n}-a)(u(\vec{v}\vec{n}+a)+n_x\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2c^2}(\vec{v}\vec{n}+a)(u(\vec{v}\vec{n}-a)+n_x\mathbf{c}^2)\right\} \\
+\frac{u\lambda_1^{\pm}}{a^2}\left\{(vn_y+wn_z)\vec{v}\vec{n}+(n_y^2+n_z^2)c^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{a^2}\left\{u\vec{v}\vec{n}+n_xc^2\right\} \\
-\frac{wn_x\lambda_1^{\pm}}{a^2}\left\{u\vec{v}\vec{n}+n_xc^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a^2}\left\{\lambda_3^{\pm}(u(\vec{v}\vec{n}+a)+n_xc^2)+\lambda_4^{\pm}(u(\vec{v}\vec{n}-a)+n_x\mathbf{c}^2)\right\} \\
\\
\frac{p}{a^2}\left\{\lambda_1^{\pm}(n_y\vec{v}\vec{n}-v\vec{n}^2)\right. \\
-\frac{\lambda_3^{\pm}}{2c^2}(\vec{v}\vec{n}-a)(v(\vec{v}\vec{n}+a)+n_y\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2c^2}(\vec{v}\vec{n}+a)(v(\vec{v}\vec{n}-a)+n_y\mathbf{c}^2)\right\} \\
-\frac{un_x\lambda_1^{\pm}}{a^2}\left\{v\vec{v}\vec{n}+n_y\mathbf{c}^2\right\} \\
+\frac{v\lambda_1^{\pm}}{a^2}\left\{(un_x+wn_z)\vec{v}\vec{n}+(n_x^2+n_z^2)c^2\right\} \\
-\frac{wn_x\lambda_1^{\pm}}{a^2}\left\{v\vec{v}\vec{n}+n_y\mathbf{c}^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a^2}\left\{\lambda_3^{\pm}(v(\vec{v}\vec{n}+a)+n_yc^2)+\lambda_4^{\pm}(v(\vec{v}\vec{n}-a)+n_y\mathbf{c}^2)\right\} \\
\\
\frac{p}{a^2}\left\{\lambda_1^{\pm}(n_z\vec{v}\vec{n}-w\vec{n}^2)\right. \\
-\frac{\lambda_3^{\pm}}{2c^2}(\vec{v}\vec{n}-a)(w(\vec{v}\vec{n}+a)+n_z\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2c^2}(\vec{v}\vec{n}+a)(w(\vec{v}\vec{n}-a)+n_z\mathbf{c}^2)\right\} \\
-\frac{un_z\lambda_1^{\pm}}{a^2}\left\{w\vec{v}\vec{n}+n_z\mathbf{c}^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{a^2}\left\{w\vec{v}\vec{n}+n_z\mathbf{c}^2\right\} \\
+\frac{w\lambda_1^{\pm}}{a^2}\left\{(un_x+vn_y)\vec{v}\vec{n}+(n_x^2+n_y^2)\mathbf{c}^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a^2}\left\{\lambda_3^{\pm}(w(\vec{v}\vec{n}+a)+n_z\mathbf{c}^2)+\lambda_4^{\pm}(w(\vec{v}\vec{n}-a)+n_z\mathbf{c}^2)\right\}
\end{bmatrix} \tag{2.21}
$$

B.2 Moving Mesh Formulation

When moving meshes are introduced the flux vectors alter to accommodate the Global Conservation Law by the introduction of the contravarient velocities. As a result the state and flux vector are defined as

$$\mathbf{U} = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix} \mathbf{f} = \begin{bmatrix} \mathbf{c}^2(u-u_g) \\ u(u-u_g)+p \\ v(u-u_g) \\ w(u-u_g) \end{bmatrix} \mathbf{g} = \begin{bmatrix} \mathbf{c}^2(v-v_g) \\ u(v-v_g) \\ v(v-v_g)+p \\ w(v-v_g) \end{bmatrix}$$

$$\mathbf{h} = \begin{bmatrix} \mathbf{c}^2(w-w_g) \\ u(w-w_g) \\ v(w-w_g) \\ w(w-w_g)+p \end{bmatrix}$$

The method of calculating the eigenvectors and the various matrices is the same as defined in Appendix B.1. The eigenvector, Λ_{MM}, is defined in Equation 2.22. Matrices $\mathbf{P}_{MM}$ and $\mathbf{P}^{-1}_{MM}$ are defined in Equations 2.26 and 2.27 respectively. Finally the directionalised Jacobian, $\mathbf{K}^{\pm}_{MM}$, and the directionalised flux vector, $\mathbf{F}^{\pm}_{MM}$, are defined by Equations 2.28 and 2.29 respectively.

$$\Lambda_{MM} = \begin{bmatrix} \vec{v_g}\vec{n} & -- & -- & -- \\ -- & \vec{v_g}\vec{n} & -- & -- \\ -- & -- & \vec{v_m}\vec{n}+\mathbf{a}_m & -- \\ -- & -- & -- & \vec{v_m}\vec{n}-\mathbf{a}_m \end{bmatrix} \tag{2.22}$$

$$\text{where} \quad \mathbf{a}^2_m = \vec{v_m}\vec{n} + \mathbf{c}^2(n_x^2+n_y^2+n_z^2) \tag{2.23}$$

Because of the use of two distinct velocity terms, $\vec{v}$ and $\vec{v_g}$ defining the flow speed and the mesh speed respectively, there are a number of extra velocity terms used -

$$\vec{v_{cv}} = \vec{v} - \vec{v_g} \tag{2.24}$$

$$\vec{v_m} = \vec{v} - \frac{1}{2}\vec{v_g} \tag{2.25}$$

$$\mathbf{P}=\begin{bmatrix} 0 & 0 & \mathbf{c}^2\left(\mathbf{a}_m+\frac{1}{2}\overrightarrow{v_g}\vec{n}\right) & -\mathbf{c}^2\left(\mathbf{a}_m-\frac{1}{2}\overrightarrow{v_g}\vec{n}\right) \\ -n_z & -n_y & u(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+\mathbf{c}^2 n_x & u(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+\mathbf{c}^2 n_x \\ 0 & n_x & v(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+\mathbf{c}^2 n_y & v(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+\mathbf{c}^2 n_y \\ n_x & 0 & w(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+\mathbf{c}^2 n_z & w(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+\mathbf{c}^2 n_z \end{bmatrix} \tag{2.26}$$

$$\mathbf{P}^{-1}=\begin{bmatrix} \frac{n_x\vec{v}\vec{n}-w}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{w\overrightarrow{v_{cv}}\vec{n}+n_x\mathbf{c}^2}{\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{n_y(w\overrightarrow{v_{cv}}\vec{n}+n_x\mathbf{c}^2)}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{(un_x+vn_y)\overrightarrow{v_{cv}}\vec{n}+(n_x^2+n_y^2)\mathbf{c}^2}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \\ \frac{n_y\vec{v}\vec{n}-v}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{v\overrightarrow{v_{cv}}\vec{n}+n_y\mathbf{c}^2}{\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{(un_z+wn_x)\overrightarrow{v_{cv}}\vec{n}+(n_z^2+n_x^2)\mathbf{c}^2}{n_z\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{n_x(v\overrightarrow{v_{cv}}\vec{n}+n_y\mathbf{c}^2)}{n_z\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \\ \frac{\vec{v}\vec{n}\left(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m\right)+\mathbf{c}^2}{2\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)\mathbf{a}_m\mathbf{c}^2} & \frac{n_x\left(\mathbf{a}_m-\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_y\left(\mathbf{a}_m-\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_z\left(\mathbf{a}_m-\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \\ -\frac{\vec{v}\vec{n}\left(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m\right)+\mathbf{c}^2}{2\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)\mathbf{a}_m\mathbf{c}^2} & \frac{n_x\left(\mathbf{a}_m+\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_y\left(\mathbf{a}_m+\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_z\left(\mathbf{a}_m+\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \end{bmatrix} \tag{2.27}$$

$$
\mathbf{K}^{\pm} = \left[\begin{array}{cccc}
\frac{1}{2a_m\phi_D}\left\{\begin{array}{c} -\lambda_3^{\pm}\phi_B(\vec{v}\vec{n}\phi_B - a^2) \\ +\lambda_4^{\pm}\phi_C(\vec{v}\vec{n}\phi_C + a^2)\end{array}\right\} & \frac{c^2 n_x}{2a_m\phi_D}(\lambda_3^{\pm}\phi_B\phi_C - \lambda_4^{\pm}\phi_B\phi_C) & \frac{c^2 n_y}{2a_m\phi_D}(\lambda_3^{\pm}\phi_B\phi_C - \lambda_4^{\pm}\phi_B\phi_C) & \frac{c^2 n_z}{2a_m\phi_D}(\lambda_3^{\pm}\phi_B\phi_C - \lambda_4^{\pm}\phi_B\phi_C) \\
\frac{1}{\phi_D}\left\{\begin{array}{c} \lambda_1(n_x\vec{v}\vec{n} - u) \\ -\frac{\lambda_3(\vec{v}\vec{n}\phi_B - a^2)}{2a_m c^2}\begin{pmatrix} u(\vec{v_m}\vec{n} + a_m) \\ +n_x c^2\end{pmatrix} \\ -\frac{\lambda_4(\vec{v}\vec{n}\phi_C + a^2)}{2a_m c^2}\begin{pmatrix} u(\vec{v_m}\vec{n} - a_m) \\ +n_x c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} \lambda_1\begin{pmatrix}(vn_y + wn_z)\vec{v_{cv}}\vec{n} \\ +(n_y^2 + n_z^2)c^2\end{pmatrix} \\ +\frac{n_x\lambda_3\phi_C}{2a_m}\begin{pmatrix} u(\vec{v_m}\vec{n} + a_m) \\ +n_x c^2\end{pmatrix} \\ +\frac{n_x\lambda_4\phi_B}{2a_m}\begin{pmatrix} u(\vec{v_m}\vec{n} - a_m) \\ +n_x c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} -\lambda_1 n_y(u\vec{v_{cv}}\vec{n} + n_x c^2) \\ +\frac{n_y\lambda_3\phi_C}{2a_m}\begin{pmatrix}(\vec{v_m}\vec{n} + a_m) \\ +n_x c^2\end{pmatrix} \\ +\frac{n_y\lambda_4\phi_B}{2a_m}\begin{pmatrix} u(\vec{v_m}\vec{n} - a_m) \\ +n_x c^2\end{pmatrix}\end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_z(u\vec{v_{cv}}\vec{n} + n_x c^2) \\ +\frac{n_z\lambda_3\phi_C}{2a_m}\begin{pmatrix} u(\vec{v_m}\vec{n} + a_m) \\ +n_x c^2\end{pmatrix} \\ +\frac{n_z\lambda_4\phi_B}{2a_m}\begin{pmatrix} u(\vec{v_m}\vec{n} - a_m) \\ +n_x c^2\end{pmatrix}\end{array}\right\} \\
\frac{1}{\phi_D}\left\{\begin{array}{c} \lambda_1(n_y\vec{v}\vec{n} - v) \\ -\frac{\lambda_3(\vec{v}\vec{n}\phi_B - a^2)}{2a_m c^2}\begin{pmatrix} v(\vec{v_m}\vec{n} + a_m) \\ +n_y c^2\end{pmatrix} \\ -\frac{\lambda_4(\vec{v}\vec{n}\phi_C + a^2)}{2a_m c^2}\begin{pmatrix} v(\vec{v_m}\vec{n} - a_m) \\ +n_y c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} -\lambda_1 n_x(v\vec{v_{cv}}\vec{n} + n_y c^2) \\ +\frac{n_x\lambda_3\phi_C}{2a_m}\begin{pmatrix} v(\vec{v_m}\vec{n} + a_m) \\ +n_y c^2\end{pmatrix} \\ +\frac{n_x\lambda_4\phi_B}{2a_m}\begin{pmatrix} v(\vec{v_m}\vec{n} - a_m) \\ +n_y c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} \lambda_1\begin{pmatrix}(un_x + wn_z)\vec{v_{cv}}\vec{n} \\ +(n_x^2 + n_z^2)c^2\end{pmatrix} \\ +\frac{n_y\lambda_3\phi_C}{2a_m}\begin{pmatrix} v(\vec{v_m}\vec{n} + a_m) \\ +n_y c^2\end{pmatrix} \\ +\frac{n_y\lambda_4\phi_B}{2a_m}\begin{pmatrix} v(\vec{v_m}\vec{n} - a_m) \\ +n_y c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} -\lambda_1 n_z(v\vec{v_{cv}}\vec{n} + n_y c^2) \\ +\frac{n_z\lambda_3\phi_C}{2a_m}\begin{pmatrix} v(\vec{v_m}\vec{n} + a_m) \\ +n_y c^2\end{pmatrix} \\ +\frac{n_z\lambda_4\phi_B}{2a_m}\begin{pmatrix} v(\vec{v_m}\vec{n} - a_m) \\ +n_y c^2\end{pmatrix}\end{array}\right\} \\
\frac{1}{\phi_D}\left\{\begin{array}{c} \lambda_1(n_z\vec{v}\vec{n} - w) \\ -\frac{\lambda_3(\vec{v}\vec{n}\phi_B - a)}{2a_m c^2}\begin{pmatrix} w(\vec{v_m}\vec{n} + a_m) \\ +n_z c^2\end{pmatrix} \\ -\frac{\lambda_4(\vec{v}\vec{n}\phi_C + a)}{2a_m c^2}\begin{pmatrix} w(\vec{v_m}\vec{n} - a_m) \\ +n_z c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} -\lambda_1 n_x(w\vec{v_{cv}}\vec{n} + n_z c^2) \\ +\frac{n_x\lambda_3\phi_C}{2a_m}\begin{pmatrix} w(\vec{v_m}\vec{n} + a_m) \\ +n_z c^2\end{pmatrix} \\ +\frac{n_x\lambda_4\phi_B}{2a_m}\begin{pmatrix} w(\vec{v_m}\vec{n} - a_m) \\ +n_z c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} -\lambda_1 n_y(w\vec{v_{cv}}\vec{n} + n_z c^2) \\ +\frac{n_y\lambda_3\phi_C}{2a_m}\begin{pmatrix} w(\vec{v_m}\vec{n} + a_m) \\ +n_z c^2\end{pmatrix} \\ +\frac{n_y\lambda_4\phi_B}{2a_m}\begin{pmatrix} w(\vec{v_m}\vec{n} - a_m) \\ +n_z c^2\end{pmatrix}\end{array}\right\} & \frac{1}{\phi_D}\left\{\begin{array}{c} \lambda_1\begin{pmatrix}(un_x + vn_y)\vec{v_{cv}}\vec{n} \\ +(n_x^2 + n_y^2)c^2\end{pmatrix} \\ +\frac{n_z\lambda_3\phi_C}{2a_m}\begin{pmatrix} w(\vec{v_m}\vec{n} + a_m) \\ +n_z c^2\end{pmatrix} \\ +\frac{n_z\lambda_4\phi_B}{2a_m}\begin{pmatrix} w(\vec{v_m}\vec{n} - a_m) \\ +n_z c^2\end{pmatrix}\end{array}\right\}
\end{array}\right] \tag{2.28}
$$

where $\phi_D = a^2 - \vec{v_g}\vec{n}\vec{v}\vec{n}$, $\phi_B = a_m + \frac{1}{2}\vec{v_g}\vec{n}$ and $\phi_C = a_m - \frac{1}{2}\vec{v_g}\vec{n}$

$$
\mathbf{F}^{\pm} = \left[
\begin{array}{c}
\frac{p}{2a_m\phi_D}\left\{\lambda_4^{\pm}(\vec{v}\vec{n}\phi_C + a^2) - \lambda_3^{\pm}(\vec{v}\vec{n}\phi_B - a^2)\right\} + \frac{c^2\vec{v}\vec{n}}{2a_m\phi_D}\left\{\lambda_3^{\pm}\phi_B\phi_C - \lambda_4^{\pm}\phi_B\phi_C\right\} \\
\\
\frac{p}{\phi_D}\left\{\lambda_1^{\pm}(n_x\vec{v}\vec{n} - u)\right. \\
-\frac{\lambda_3^{\pm}}{2a_m c^2}(\vec{v}\vec{n}\phi_B - a)(u(\overrightarrow{v_m}\vec{n} + a_m) + n_x c^2) \\
\left.-\frac{\lambda_4^{\pm}}{2a_m c^2}(\vec{v}\vec{n}\phi_C + a)(u(\overrightarrow{v_m}\vec{n} - a_m) + n_x c^2)\right\} \\
+\frac{u\lambda_1^{\pm}}{\phi_D}\left\{(vn_y + wn_z)\overrightarrow{v_{cv}}\vec{n} + (n_y^2 + n_z^2)c^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{\phi_D}\left\{u\overrightarrow{v_{cv}}\vec{n} + n_x c^2\right\} \\
-\frac{wn_x\lambda_1^{\pm}}{\phi_D}\left\{u\overrightarrow{v_{cv}}\vec{n} + n_x c^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a_m\phi_D}\left\{\lambda_3^{\pm}\phi_C(u(\overrightarrow{v_m}\vec{n} + a_m) + n_x c^2) + \lambda_4^{\pm}\phi_B(u(\overrightarrow{v_m}\vec{n} - a_m) + n_x c^2)\right\} \\
\\
\frac{p}{\phi_D}\left\{\lambda_1^{\pm}(n_y\vec{v}\vec{n} - v)\right. \\
-\frac{\lambda_3^{\pm}}{2a_m c^2}(\vec{v}\vec{n}\phi_B - a)(v(\overrightarrow{v_m}\vec{n} + a_m) + n_y c^2) \\
\left.-\frac{\lambda_4^{\pm}}{2a_m c^2}(\vec{v}\vec{n}\phi_C + a)(v(\overrightarrow{v_m}\vec{n} - a_m) + n_y c^2)\right\} \\
-\frac{un_x\lambda_1^{\pm}}{\phi_D}\left\{v\overrightarrow{v_{cv}}\vec{n} + n_y c^2\right\} \\
+\frac{v\lambda_1^{\pm}}{\phi_D}\left\{(un_x + wn_z)\overrightarrow{v_{cv}}\vec{n} + (n_x^2 + n_z^2)c^2\right\} \\
-\frac{wn_z\lambda_1^{\pm}}{\phi_D}\left\{v\overrightarrow{v_{cv}}\vec{n} + n_y c^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a_m\phi_D}\left\{\lambda_3^{\pm}\phi_C(v(\overrightarrow{v_m}\vec{n} + a_m) + n_y c^2) + \lambda_4^{\pm}\phi_B(v(\overrightarrow{v_m}\vec{n} - a_m) + n_y c^2)\right\} \\
\\
\frac{p}{\phi_D}\left\{\lambda_1^{\pm}(n_z\vec{v}\vec{n} - w)\right. \\
-\frac{\lambda_3^{\pm}}{2a_m c^2}(\vec{v}\vec{n}\phi_B - a)(w(\overrightarrow{v_m}\vec{n} + a_m) + n_z c^2) \\
\left.-\frac{\lambda_4^{\pm}}{2a_m c^2}(\vec{v}\vec{n}\phi_C + a)(w(\overrightarrow{v_m}\vec{n} - a_m) + n_z c^2)\right\} \\
-\frac{un_x\lambda_1^{\pm}}{\phi_D}\left\{w\overrightarrow{v_{cv}}\vec{n} + n_z c^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{\phi_D}\left\{w\overrightarrow{v_{cv}}\vec{n} + n_z c^2\right\} \\
+\frac{w\lambda_1^{\pm}}{\phi_D}\left\{(un_x + vn_y)\overrightarrow{v_{cv}}\vec{n} + (n_x^2 + n_y^2)c^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a_m\phi_D}\left\{\lambda_3^{\pm}\phi_C(w(\overrightarrow{v_m}\vec{n} + a_m) + n_z c^2) + \lambda_4^{\pm}\phi_B(w(\overrightarrow{v_m}\vec{n} - a_m) + n_z c^2)\right\}
\end{array}
\right] \tag{2.29}
$$

C Least Squares Theory

The least squares criterion is suitable for situations where there is vastly more data available than parameters, hence causing an exact match to be highly uncommon. Least squares is often regarded as smoothing the data, or noise reduction.

If M measurements are taken of some quantity, x, then we do not see x but only the measurements x_i with errors of measurement ε_i.

$$x_i = x + \varepsilon_i \qquad\qquad i = 1, 2, 3 \ldots M \tag{3.1}$$

If the residuals ε_i are regarded as noise, and x regarded as the *true* value, then the principle of least squares states that the best estimate $\hat{x}$ of x is that value which minimises the sum of the squares of the deviations of the data from their estimate.

$$f(\hat{x}) = \sum_{i=1}^{M} \varepsilon_i^2 = \sum_{i=1}^{M} (x_i - \hat{x})^2 = minimum \tag{3.2}$$

The minimal value of this function, $f(\hat{x})$, must occur when

$$\frac{\partial f(\hat{x})}{\partial \hat{x}} = 0. \tag{3.3}$$

Thus a set of linear equations can be created for each variable, which can be solved by matrix manipulation.

If there are a linearly independent set of functions $f_j(x)$ that are to be fitted to a set of data, then the following can be assumed -

$$y(x) = a_0 f_0(x) + a_1 f_1(x) + a_2 f_2(x) \ldots + a_N f_N(x) \tag{3.4}$$

and the least squares fit is given by

$$\sum_{i=1}^{M}\left[\sum_{j=1}^{N} a_j f_j(x_i) - y_i\right]^2 = min \tag{3.5}$$

Differentiating this with respect to the parameters a_k (which occur linearly) leads to the system of equations -

$$\sum_{i=1}^{M}\sum_{j=1}^{N} a_j f_j(x_i) f_k(x_i) = \sum_{i=1}^{M} y_i f_k(x_i) \qquad k = 1, 2 \ldots N. \tag{3.6}$$

If the following notation is used

$$S_{j,k} = \sum_{i=k}^{M} f_j(x_i) f_k(x_i)$$

$$T_k = \sum_{i=1}^{M} y_i f_k(x_i),$$

then Equation 3.6 can be written as

$$\sum_{j=1}^{N} a_j S_{j,k} = T_k. \tag{3.7}$$

The calculation of all terms and the inversion of $S_{j,k}$ thus provides the variables a_j.

D Multigridding

Multigridding aids very fast convergence to a steady state via the use of several meshes at the same time. Normally two or three successively finer meshes are used in parallel[45, 51, 98, 99].

Auxiliary grids are introduced by halving the grid spacing in the local ξ, η, ζ directions, so from one hexahedra two, four, then eight children are created. Because of the presence of the binary tree data structure, and the form of use of memory linked lists and stack the creation of the graded meshes is simple. Recognition of the underlying surface definition on solid walls allows accurate subdivision of control volumes adjacent to the surface.

D.1 Theory

As presented in Chapter 4, the numerical system of equations defining the finite volume discretisation of the Euler equations can be written in a matrix format. In multidimensional applications of finite difference methods the matrix relating the variables to their functions (in this case the conservative state vector **U** and their fluxes **F**) is large and sparse, with the non zero pattern having a regular structure. As noted in Chapter 4, these circumstances favour the use of iterative methods for their solution.

D.1.1 Von Neumann error analysis

By Von Neumann error analysis, the error at any given data point in the domain, i, and given iteration, n, is given by a Fourier series as:

$$\begin{aligned} \epsilon_i^n &= u^\infty - u_i^n \\ &= \sum_{j=-N}^{N} C_j^n e^{Iij\frac{\pi}{N}} \end{aligned} \tag{4.1}$$

the exponential term $j\frac{\pi}{N}$ is often represented as a phase angle,

$$\theta \equiv j\frac{\pi}{N}, \quad \theta \in (-\pi, \pi) \tag{4.2}$$

The amplification factor, which measures the growth or decay of the error during an iteration, is defined as $G = \frac{C^{n+1}}{C^n}$. That is, it is the ratio of the amplitudes of successive harmonics. For a scheme to be stable this value $G \leq 1.0$, and is a function of time step, frequency and mesh size.

For the first order upwinding scheme presented in Section 4.2 the stability conditions have to be considered carefully. This is because the Jacobians of $f^{\pm}$ are not equal to $K^{\pm}$, *and* their eigenvalues are different, hence the linear CFL condition of Equation 4.3 is not the true stability limit. If a local linearisation is performed upon a given state, U^n, then Equation 4.18 becomes

$$\mathbf{U}_i^{n+1} - \mathbf{U}_i^n = -\frac{\partial t}{\partial x}\mathbf{F}_u^{+n}(\mathbf{U}_i - \mathbf{U}_{i-1})^n - \frac{\partial t}{\partial x}\mathbf{F}_u^{-n}(\mathbf{U}_{i+1} - \mathbf{U}_i)^n \tag{4.3}$$

where $\mathbf{F}_u^{+n}$ and $\mathbf{F}_u^{-n}$ are the linearised values taken by the Jacobians of $\mathbf{F}^+$ and $\mathbf{F}^-$ for $U = U^n$. The amplification matrix G becomes

$$G - 1 = -2\frac{\partial t}{\partial x}(\mathbf{F}_u^+ - \mathbf{F}_u^-)\sin^2\frac{\theta}{2} - I\frac{\partial t}{\partial x}(\mathbf{F}_u^+ + \mathbf{F}_u^-)\sin\theta \tag{4.4}$$

If the matrix $|\mathbf{F}_u| = \mathbf{F}_u^+ - \mathbf{F}_u^-$ is introduced (which is different from $|\mathbf{K}|$), then

$$G = 1 - 2\frac{\partial t}{\partial x}|\mathbf{F}_u|\sin^2\frac{\theta}{2} - I\frac{\partial t}{\partial x}\mathbf{K}\sin\theta \tag{4.5}$$

The difficulty in defining the stability limit arises because $|\mathbf{F}_u|$ and $\mathbf{A}$ have, generally, a different set of eigenvalues. An estimation of the eigenvalues of G can be obtained by first transforming G with the similarity matrix P. Since G and $\bar{G}$ have the same eigenvalues one can as easily analyse the eigenvalues of $\bar{G}$:

$$\begin{aligned} \bar{G} &= P^{-1}GP \\ &= 1 - 2\frac{\partial t}{\partial x}|E|\sin^2\frac{\theta}{2} - I\frac{\partial t}{\partial x}\Lambda\sin\theta \end{aligned} \tag{4.6}$$

where $|E|$ is the transformed matrix of $|\mathbf{F}_u|$ and Λ the diagonal matrix of the eigenvalues of $\mathbf{A}$.

$$|E| = P^{-1}|\mathbf{F}_u|P = P^{-1}\mathbf{F}_u^+P - P^{-1}\mathbf{F}_u^-P \tag{4.7}$$

Numerical estimation of the eigenvalues of $\bar{G}$ show that they reach their maximum for the high frequency limit $\theta = \pi$. Hence the maximum eigenvalue of $\bar{G}$ is given by

$$\lambda(G)_{max} = 1 - 2\frac{\partial t}{\partial x}\lambda(|E|)_{max}. \tag{4.8}$$

D.1.2 Stability and Efficiency

As the amplification factor decreases below 1.0 the convergence rate of the scheme improves. Equation 4.1 shows that the asymptotic behaviour of the error is dominated by the eigenvalues of the amplification matrix close to one (in absolute value). The lowest eigenvalue of G is obtained via Equation 4.8 for the low frequency component $\theta = \frac{\pi}{N}$. With further study it can be shown that as h, the mesh characteristic cell size, decreases then $\theta(= \frac{j\pi}{N})$ decreases and the rate of convergence deteriorates. This reduction of rate of convergence is found to occur with all basic iterative methods and for most kinds of boundary conditions [105].

The rate of convergence of basic iterative methods can be improved with multigrid methods. The basic observation from Equation 4.5 shows that $|G|$ decreases as j increases. This means that although long wavelength error modes ($j \approx 1$) decay slowly, short wavelength modes are reduced rapidly. The essential multigrid principle is to approximate the smooth (long wavelength) error modes on coarser grids. The higher frequency errors are reduced with a small number of iterations with a basic iterative method on the fine grid.

The ability to efficiently reduce the high frequency (rough) errors is defined by the largest eigenvalue of G, for the range of frequencies that are resolved by the finest mesh. The set of rough wave numbers, θ_r is defined by

$$\theta_r = \left\{\theta = j\frac{\pi}{N}, |j| \geq CN, j \in (-N, N)\right\} \tag{4.9}$$

where $0 \leq C \leq 1$ is a fixed constant independent of N. For the standard case of doubling the grid spacing to obtain the coarse mesh, $C = 0.5$.

D.1.3 Two grid and full multigrid models

There are two broad categories of multigrid, namely Correction Schemes (CS) and Full Approximation Schemes (FAS). The former operates with corrections to the solution on fine grids which are eventually added to the fine grid while FAS operates on the solution (rather than its correction) on all grid levels. CS is suitable for linear systems while FAS is better suited to coupled systems of strongly non-linear algebraic equations. Thus FAS is a better approach to use for the highly coupled and complex sets of governing equations of fluid flow [106].

A coarse grid, D_c, is defined by doubling the mesh size of the domain, D_f:

$$D_f = \{x \in \Re : x = x_i = ih_f, i \in (1, 2n), h_f = 1/2n\} \quad (4.10)$$

$$D_c = \{x \in \Re : x = x_i = ih_c, i \in (1, n), h_c = 1/n\} \quad (4.11)$$

The vertices of D_c also belong to D_f. The original grid is called the fine grid. There does not have to be any particular relationship between the space discretisation on the fine and coarse grids, but local mode analysis shows that the coarse grid spacing should be about twice that of the next fine grid [106]. The discretisation of the governing equations upon this coarser grid results in an approximation of the fine grid algebraic system. The object of the coarse grid equations is so that once the iterations stall on the fine grid due to the presence of smooth error components a coarse grid formulation of the system may be used for the efficient elimination of these errors.

The exact solution of the discretisated equations upon the fine grid satisfy

$$\mathbf{A}_f \mathbf{U}_f = 0 \quad \text{assuming no source terms} \quad (4.12)$$

After several iterations (n_f) on the whole set of equations an approximate solution, $\mathbf{U}_f^n$, is obtained which satisfies Equation 4.12 to a residual R_f;

$$\mathbf{A}_f \mathbf{U}_f = \mathbf{A}_f^{n_f} \mathbf{U}_f^{n_f} + R_f \quad (4.13)$$

If

$$\mathbf{A}_f = \mathbf{A}_f^{n_f} + \Delta \mathbf{A}_f$$

$$\mathbf{U}_f = \mathbf{U}_f^{n_f} + \Delta \mathbf{U}_f$$

then Equation 4.13 reduces to

$$\mathbf{A}_f^{n_f} \Delta \mathbf{U}_f = R_f - \Delta \mathbf{A}_f \mathbf{U}_f^{n_f} - \Delta \mathbf{A}_f \Delta \mathbf{U}_f \quad (4.14)$$

For non-linear equation systems the correction ΔU_f depends not only upon the value of the residual R_f but also upon the changes within A which is a function of the advancing solution $\mathbf{U}_f^{n_f}$. The correction scheme (CS) can be used for non-linear problems to solve Equation 4.14 for $\Delta \mathbf{U}_f$, thereby neglecting the last two terms on the right hand side. However, in order to keep track of all non-linear dependencies and intervariable grid quantities, Equation 4.13 is used as the basis for multigrid solution in the FAS scheme. Fine grid quantities are interpolated (restricted) to the coarse grid (see Figure D.1). These restricted quantities are denoted by a tilde ($\tilde{}$) superscript. An approximation to the combination of Equations 4.12 and 4.13 on the coarse grid leads to the equation

$$\mathbf{A}_c\mathbf{U}_c = \tilde{\mathbf{A}}_c\tilde{\mathbf{U}}_c + \tilde{R}_c \tag{4.15}$$

$$\rightarrow \mathbf{A}_c^{n_c}\mathbf{U}_c^{n_c} = \left(\tilde{\mathbf{A}}_c\tilde{\mathbf{U}}_c + \tilde{R}_c\right) + R_c \tag{4.16}$$

It should be noted that the first two terms of the right hand side of this equation will remain constant during any iterations at this coarse grid level. After a few iterations on the coarse grid (n_c) the coarse grid corrections can be calculated then interpolated (prolongated) back onto the fine grid, where upon the fine grid solution can be adjusted and the process repeated until the residuals have dropped to a sufficient level. Due to interpolation some high frequency errors may be introduced during the prolongation process, but these are quickly removed during the iterative solution on the fine mesh. The coarse grid corrections are defined as

$$\Delta \mathbf{U}_c = \mathbf{U}_c^{n_c} + \tilde{\mathbf{U}}_c \tag{4.17}$$

and the fine grid adjustments are

$$\mathbf{U}_f^{n_f} = \mathbf{U}_f^{n_f} + \Delta \mathbf{U}_f \tag{4.18}$$

This is, strictly speaking, a two grid method. However, Equation 4.16 may in itself be solved using a coarser grid and so on. Therefore the extension of the solution procedure from two-grid to an arbitrary multigrid scheme is straightforward.

D.1.4 Restrictor and Prolongator Operators

The transfer of values defined on the fine grid to the next coarser one is called restriction, denoted by the operator I_c^f;

$$\tilde{\mathbf{U}}_c = I_c^f \mathbf{U}_f \tag{4.19}$$

The restrictor operator should be an average of all fine grid computation points surrounding the coarse grid one, rather than a simple transference in which the value of the coarse grid point is taken as the value at the coincident fine grid point. A linearly weighted restrictor can be used for flow properties, the only requirement being that for consistency the sum of the weighting factors should be equal to one. Since the residuals R_f and $\tilde{R}_c$ are not functions of space and

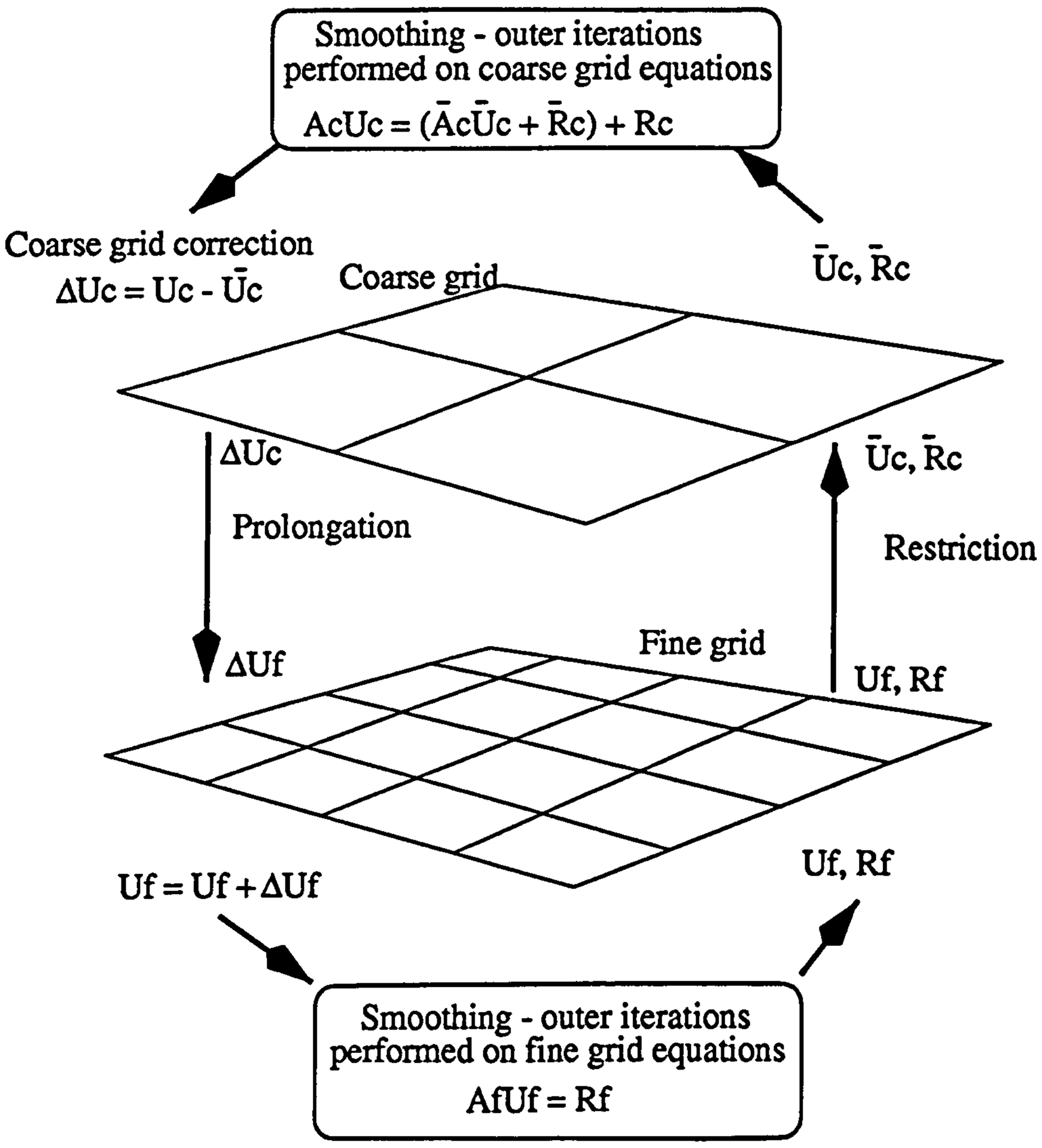


Figure D.1: Two grid multigrid coupling

time co-ordinates only, but also dependant upon the mesh size, interpolation is inappropriate. The residual on the coarse grid can be simply obtained by summation of the residuals of the fine cells compromising the coarse cell.

The prolongator operator I_f^c transfers values from the coarse to the fine grid.

$$\mathbf{U}'_f = I_f^c \mathbf{U}'_c \tag{4.20}$$

The prolongator operator may represent a simple transfer of values to the fine grid assuming that the field values are constant, or it may represent a transfer of some weighted averages. For an effective multigrid the order of interpolation should be no less than the order of the parent differential equations. The linear interpolation is widely accepted as the optimum:

$$\mathbf{U}_f = \mathbf{U}_c + \nabla_c \mathbf{U}_c \cdot (R_f - R_c) \tag{4.21}$$

D.1.5 Multigrid cycle

The coarse grid equations used in the multigrid process are only approximate (until convergence), so there is no requirement to solve them exactly. By contrast, it is important to sufficiently reduce the long wavelength errors so that they do not dominate the convergence rate after interpolation back onto the fine grid. The order in which the grids of different resolution are visited defines the multigrid cycle, the aim of which is to maximise the potential of each grid in reducing errors. Figure D.2 presents the schedule for the grids in the order in which they are used for a 'V' and 'W' multigrid cycle. The 'V' cycle consists of a succession of smoothing and transfer to the next coarser grid until the coarsest grid is reached. This is followed by a sequence of smoothing and transference of information back to the finest grid. In the 'W' cycle intermediate 'V' cycles are performed on the coarser grids. The computational work of a multigrid cycle is dominated by the number of operations required for the two grid computation from the finest grid to the next level. The 'W' cycle is in general 50% more computationally expensive that the 'V' cycle, but is more robust and should be preferred, particularly for sensitive problems.

D.1.6 Convergence

The choice of the number of outer iterations to be performed on each grid is not always obvious. There are two common approaches:

The fixed scheme The cycling parameters are chosen *a priori* and remain fixed throughout the course of the algorithm. Their choice is made on the basis of analysis or past experience

The accommodative scheme The parameters are determined during the run to account for variations in the pattern of convergence.

The outer iteration sequence of the accommodative algorithms are controlled by the ratio of the current residual (R_l^n) to the previous one (R_l^{n-1}) at the l-level grid, where the n defines the outer iteration counter on that grid;

$$\frac{R_l^n}{R_l^{n-1}} < \Omega \tag{4.22}$$

If Equation 4.22 holds, then the scheme is deemed efficient and is continued until $R_l^n < \mu_l$. If the equation is not satisfied then the solution and residuals are restricted onto the next coarser level. The convergence criterion for that grid

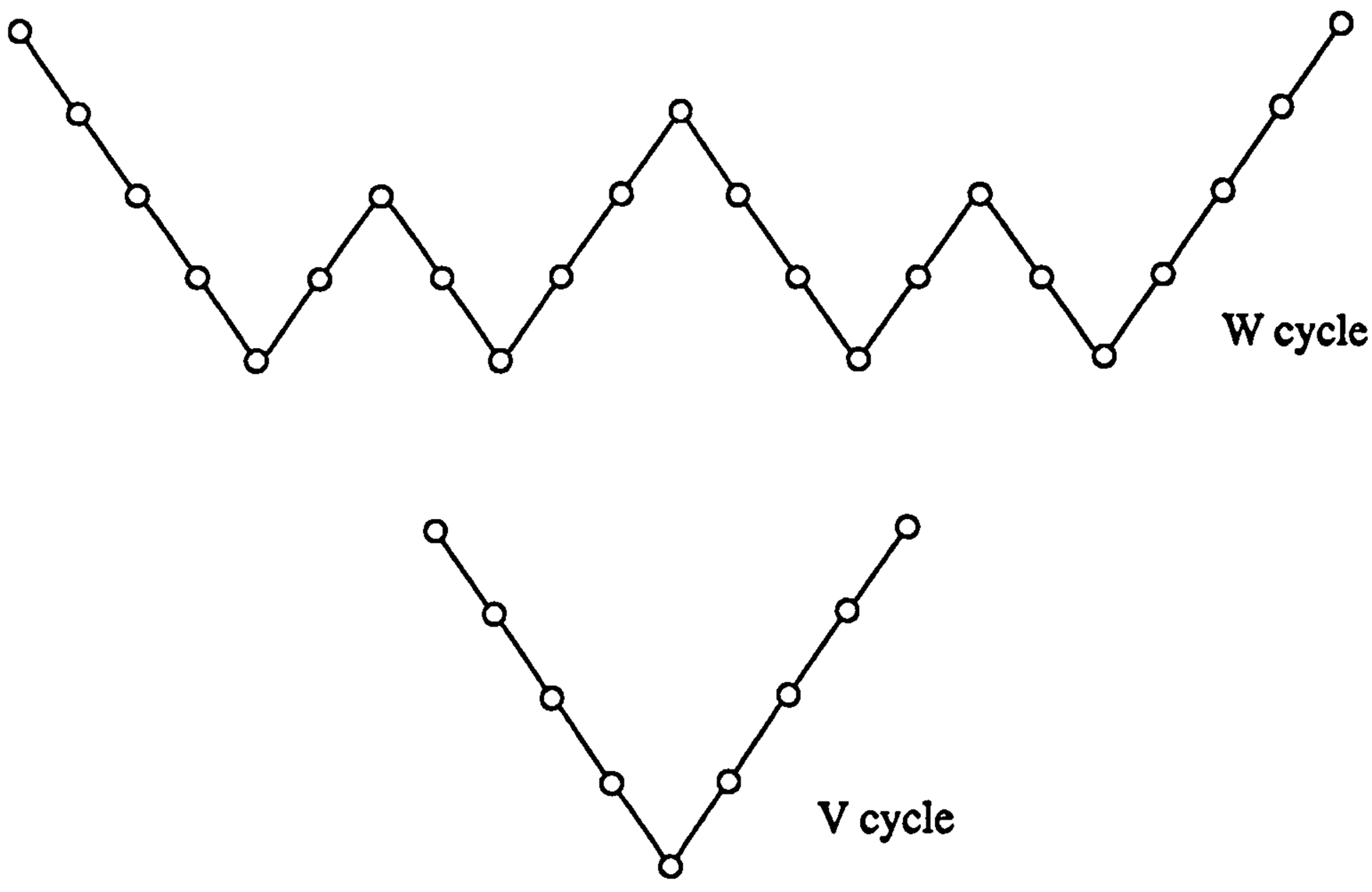


Figure D.2: The two basic types of multigrid cycle possible

(μ_{l-1}), is set on the basis of the last global residual on the l-level grid:

$$\mu_{l-1} < \mathcal{C}|R_l^n| \tag{4.23}$$

The values of the convergence rate criterion, Ω, and the convergence ratio, $\mathcal{C}$, that provide the most efficient scheme, vary but are generally in the region of 0.3 to 0.6. Since the convergence is not always monotone and smooth (see Figures 4.7 and 4.17) accommodative algorithms are found not to perform better in a general case than the fixed ones. Convergence analysis for multigrid methods is difficult to perform, and still has aspects that are obscure to fully understand and define accurately. However a wealth of computational evidence testifies to the general effectiveness of multigrid methods [106].

Hence the scheme that has been implemented in the flow solver detailed in Chapter 4 uses a Full Approximation Scheme, as shown in Figure D.1, upon a two grid architecture. A fixed number of iterations on each of the meshes will be used until the residual on the finest mesh is below the overall limit. In order to aid the overall efficiency of the algorithm a gradual 'wind-up' from the coarsest grid to the finest will be used in a 'V' style cycle.

D.2 Implementation

Due to time restrictions only a two grid method has been implemented, the overall control of which is detailed in Algorithm D.1. This implementation will however demonstrate the effectiveness of the data structure for multigrid solutions as well as the efficiency benefits possible from multigridding.

Algorithm : D.1 (Multigrid strategy)

- SOLVE ON MESH 1 UNTIL SET LEVEL, μ_1
- SUB-DIVIDE DOMAIN TO CREATE THE NEXT FINER GRID (MESH 2)
- PROLONGATE ONTO MESH 2
- SOLVE ON MESH 2 FOR N_2 ITERATIONS

- Do

    1. Restrict to mesh 1
    2. Solve on mesh 1 for N_1 iterations
    3. Prolongate onto mesh 2
    4. Solve on mesh 2 for N_2 iterations

- while $R_2 > \mu_2$

The prolongator operator uses weighted averages of the state vector from the coarse mesh. This is achieved by Algorithm D.2.

Algorithm : D.2 (Prolongator operator)

- On the fine mesh
- For each of the connected control volumes
    1. Collect the state vector of the 'parent' node on the coarse mesh
- Divide $\sum$ of collected state vectors by number of connected CVs
- Fine mesh state vector is averaged state vector

Because several of the neighbouring control volumes will have the same 'parent' on the coarse mesh, the state vector will be weighted accordingly

The restrictor operator utilises a volume weighted averaging process to transfer the coarse grid properties onto the fine mesh, as detailed in Algorithm D.3

Algorithm : D.3 (Restrictor operator)

- FOR ALL 'CHILDREN' ON THE FINE MESH
  1. COLLECT CHILD CV STATE VECTOR MULTIPLIED BY VOLUME
- DIVIDE $\sum$ OF COLLECTED STATE VECTORS BY VOLUME OF 'PARENT' CV
- COARSE MESH STATE VECTOR IS VOLUME AVERAGED STATE VECTOR
- COARSE MESH STATE VECTOR RESIDUAL IS SUM OF FINE MESH STATE VECTOR RESIDUALS

D.3 Results

The multigrid algorithm was tested by the simple case study of a square channel, initially at rest, but with an external free stream of $5ms^{-1}$. The channels dimensions were 1m by 1m cross section, and 3m long, as detailed in Figure D.3. The coarse mesh (the initial mesh inserted in the flow solver) was $3 \times 3 \times 15$ nodes in size, giving a total number of 135 nodes. The fine mesh, created via the CV splitting algorithm of Section 6.3, was thus $6 \times 6 \times 30$, creating a total of 1080 nodes. Figure D.4 shows the initial control volumes in the coarse mesh, clearly defining the entire domain, and the children of these control volumes on the fine mesh that have the same global ID label. The other 945 children have been discarded for this view.

The solution was deemed to be converged when the maximum X momentum residual dropped below 1×10^{-8}.

Figure D.5 presents the convergence times for four cases; the coarse and fine meshes alone, without any multigridding, and two multigrid cases. The difference between the multigrid solutions was the regularity of prolongation and restriction. Both models initiated the first prolongation when the residual fell below 1×10^{-3}, but the first case changed meshes every 10 iterations while the second prolongated/restricted every 100 iterations.

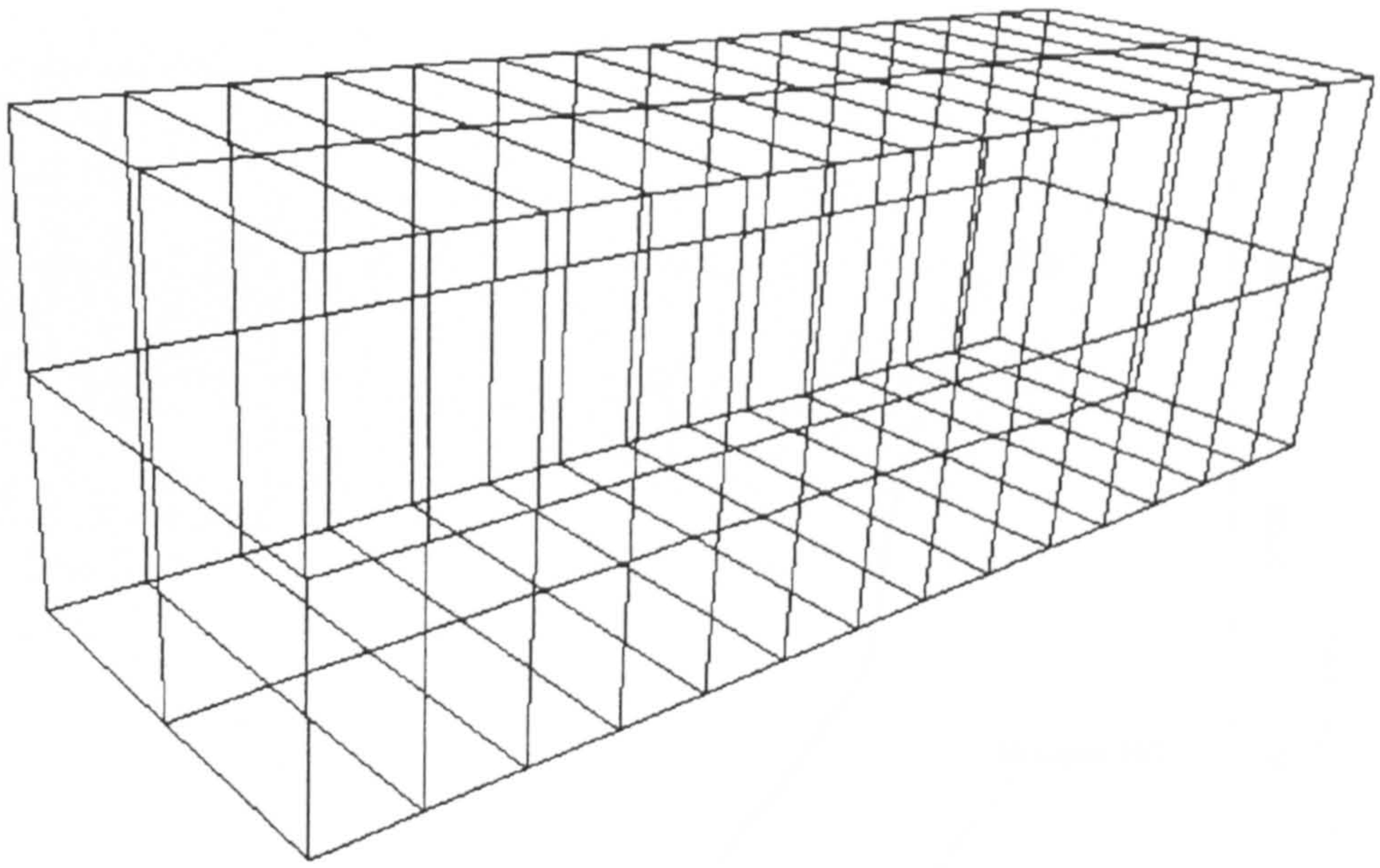


Figure D.3: Wire frame of domain and node - node connectivity on initial coarse mesh

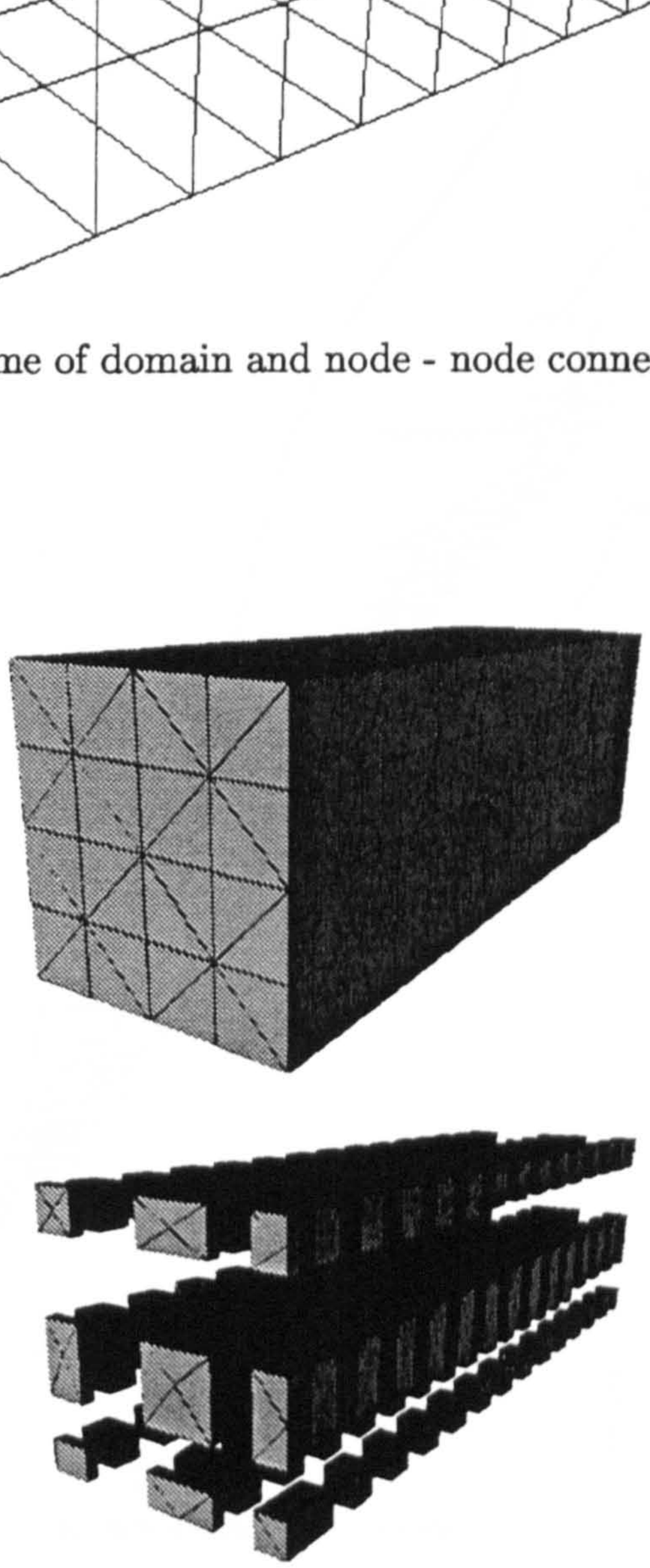


Figure D.4: Coarse and fine meshes, showing the original control volumes and the children bearing the same global ID

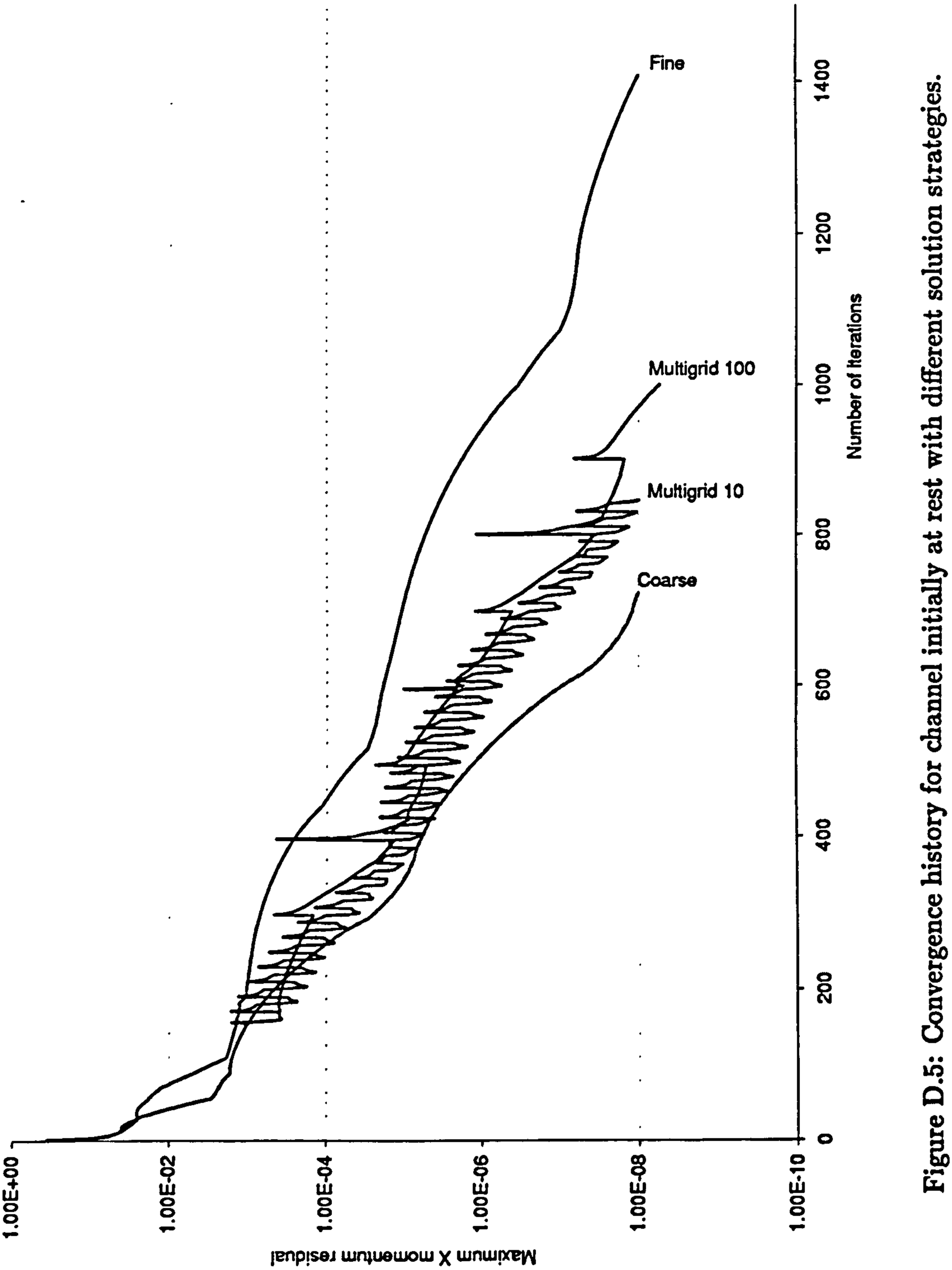


Figure D.5: Convergence history for channel initially at rest with different solution strategies.

Mesh	CPU coarse mesh (s)	CPU fine mesh (s)	Avg. CPU time per iteration (s)
Coarse mesh	0.3	–	0.3
Multigrid	1.2	2.1	1.65
Fine mesh	–	2.1	2.1

Table D.1: CPU time usage per iteration for different solution strategies

Mesh	Flow nodes (KBytes)	Construction nodes (KBytes)	Faces (KBytes)	Total (KBytes)
Coarse mesh	15.9	66.9	26.5	109.4
Multigrid	147.5	375.5	177.9	700.9
Fine mesh	131.6	375.5	177.9	685.0

Table D.2: Memory requirements for different meshes

Table D.1 details the CPU time per iteration for the various stages of the solution and Table D.2 details the memory requirements of the various solution schemes.

D.4 Discussion

The case study for testing the multigrid algorithm was selected due to it simplicity, so that the only influencing factor upon solution convergence would be the use of multiple grids.

The clear indication from Figure D.5 is that the use of multigridding decreases the number of iterations required by approximately 35%. It can be observed that the scale up in the number of iterations required from the coarse mesh to the fine mesh was a factor of 2 which was to be expected when the number of control volumes in the stream-wise direction was doubled. When using the multigrid strategy however the increase in the number of iterations has reduced to 1.3.

The drop in required CPU time is also noticeable. It can be observed from Table D.1 that the time to solve on the coarse mesh with the multigrid strategy is four times that required on the coarse mesh alone. This is due to the presence of four small faces between adjacent control volumes instead of just one. This is a

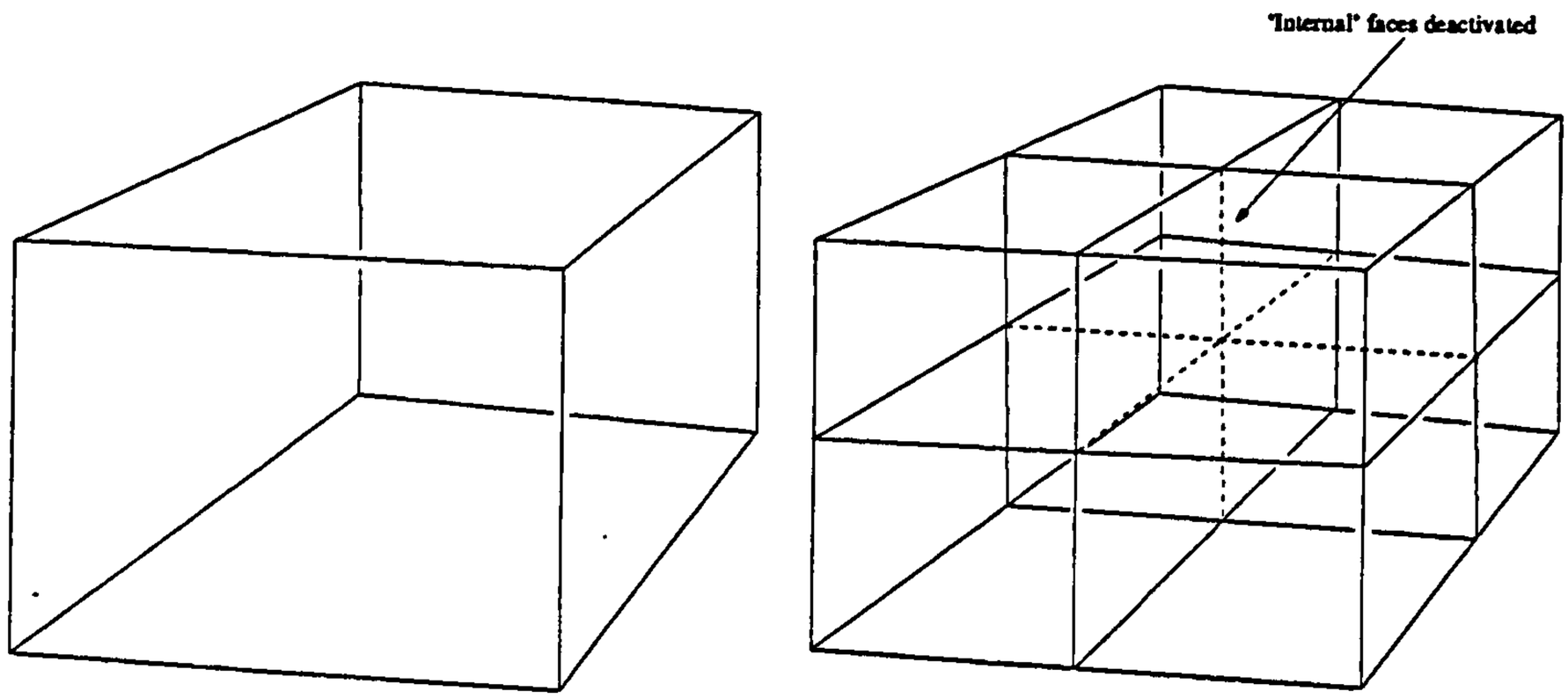


Figure D.6: Coarse mesh control volume definition with fine mesh faces

result of the initial face being split during the adaption process. The initial face is not reconstructed during restriction due to the manner of the algorithm. Figure D.6 details this, also showing the faces between children of the same parent that are switched 'off' during restriction, which are not used during any calculations on the coarse mesh. Because the main spatial discretisation algorithm is face based this results in the variation in CPU times as listed in Table D.1. There is a factor of four difference between the coarse mesh alone and solving on the coarse mesh with the multigrid due to there being four times as many faces, and the further increase in time required to solve on the fine mesh is due to the child-child faces being re-activated.

The average time per iteration (assuming an equal number of iterations on the fine and coarse meshes) is however still less than that required by the fine mesh alone. As a result the computational efficiency rises still further from a comparison of iterations alone. Table D.3 lists the total time required for each of the four solutions. Therefore the total increase in efficiency from using the multigrid strategy, for the same level of solution accuracy, is in the region of 50% to 45%. The memory requirements of the multigrid are slightly higher than that of the fine mesh, requiring the storage of the 'parent' nodes as well as the fine mesh entities. It should be noted however that the memory requirements are noticeably less than the sum of the memory requirements of the coarse and fine mesh; this is due to the non-requirement for a separate face and construction node structure. Referring to Section 2.5.1 and Equation 2.6-a, the increase in memory requirements can be approximated to be -

Mesh	Number of iterations	CPU per iteration (s)	Total CPU time (s)	CPU per node (s)
Coarse mesh	723	0.3	216.9	1.61
Multigrid (swap every 10 iterations)	846	1.65	1395.9	1.29
Multigrid (swap every 100 iterations)	967	1.65	1595.6	1.48
Fine mesh	1408	2.1	2956.8	2.74

Table D.3: Required total CPU time for different solution strategies

$$\begin{aligned} \text{EXTRA MEMORY} &= 3\frac{1}{8}N^3FLT \qquad (4.24) \\ &+ 2\cdot 3(\frac{1}{8}N-2)(\frac{1}{8}N-1)^2)PTR \\ &\simeq 7\cdot(\frac{1}{8})^3\ \% \\ &\simeq 1.4\% \qquad (4.25) \end{aligned}$$

The extra memory over and above this 1.4% , as presented in Table D.2 (approximately 1%), is due to extra flags required to signify the status of the geometrical entities.

The final points that can be drawn from Figure D.5 are the general and relative performance of the two multigrid solutions. As the coarse solution is prolongated onto the fine mesh a large spike is evident, which can be attributed to high frequency errors occurring in the coarse solution and prolongation process. These errors are rapidly removed by the fine mesh solution phase. As a result of this process the residual trace for the multigrid solution changing meshes every 10 iterations has what appears, unfoundedly, to be an unstable solution. It is worth commenting however that a more complex flow field could cause such a solution strategy to diverge, hence restricting/prolongating the solution only every 100 iterations may be a more robust strategy.

There are a number of aspects to the topic of multigridding that have been left un-developed due to time restrictions, as well as the overall aim of the work being merely to investigate the ease with which the framework lends itself to multigridding. These aspects include;

- the use of more than 2 grids, and the relative trade off between increased memory requirements and decreased solution time,
- the variation in the number of cycles between prolongation and restriction,
- the variation in the wait before the initial prolongation,
- the use of different multigrid cycles, as schematically shown in Figure D.2

D.5 Summary

Multigridding techniques have been investigated and a simple two grid model incorporated within the flow solver framework. This has proven to be very successful, requiring a minimum amount of change to the code and negligible increase in memory requirements, yet producing a very large improvement in efficiency of the code. Suggestions have been made as how to develop the technique further within the current flow solver framework, the most important of which is a parametric study of the rate of convergence variables.