Error Analysis in Finite Volume CFD

Franjo Juretic

Thesis submitted for the

Degree of Doctor of Philosophy of the University of London

and

Diploma of Imperial College

Department of Mechanical Engineering

Imperial College London

December, 2004

Abstract

This study was aimed towards improving the accuracy of Computational Fluid Dynamics (CFD) by developing methods for reliable estimation of the discretisation error and its reduction.

A new method for error estimation of the discretisation error for the second-order accurate Finite Volume Method is presented, called the Face Residual Error Estimator (FREE), which estimates the discretisation error on the cell faces. The estimator is tested on a set of cases in comparison with analytical solutions, ranging from convection-dominated to diffusion-dominated ones. Testing is also performed on a set of cases of engineering interest and on polygonal meshes.

In order to automatically produce a solution of pre-determined accuracy, an automatic error-controlled adaptive-mesh-refinement procedure is set up. It uses local mesh refinement to control the local error magnitude by refining hexahedral cells parallel to the face with large discretisation error. The procedure is tested on four cases with analytical solutions and on several laminar and turbulent-flow cases of engineering interest. It was found able to produce accurate solutions with savings in computational resources.

In order to explore the possibilities of different mesh structures, a mesh generator producing polyhedral meshes based on the Delaunay technique is developed. An adaptive-mesh-generation technique for polyhedral meshes is also developed and is based on re-meshing parts of the mesh which are selected for refinement. The mesh-adaptation technique is tested on a case with an analytical solution. A comparison of accuracy achieved on quadrilateral, triangular and polygonal meshes is also given, which shows that quadrilateral meshes perform best followed by polygonal meshes.

Acknowledgements

I would like to express my gratitude to my supervisor Prof A. D. Gosman for his interest and continuous guidance during this study.

I would like to use the opportunity to thank my colleagues from Prof Gosman's CFD group, especially Dr. Hrvoje Jasak, Mr. Henry Weller and Mr Mattijs Janssens for developing the FOAM C++ simulation code which made the implementation of the ideas easier. Their support and suggestions were invaluable.

This study and the text of this thesis has benefited a lot from the numerous suggestions and comments by Dr. Hrvoje Jasak.

It would be unfair not to thank Mrs Nicky Scott-Knight and Mrs Susan Clegg for arranging administrative matters.

Finally, the financial support provided by the Computational Dynamics Ltd. is gratefully acknowledged.

Contents

1 Introduction **25**

1.1 Background 25

1.2 Present Contributions 28

1.3 Thesis Outline 28

2 Governing Equations of Continuum Mechanics **31**

2.1 Navier-Stokes Equations 31

2.2 Constitutive Relations for Newtonian Fluids 31

2.2.1 Turbulence Modelling 32

2.3 General Form of a Transport Equation 36

2.4 Summary 36

3 Finite Volume Discretisation **39**

3.1 Introduction 39

3.2 Measures of Mesh Quality 41

3.3 Discretisation of Spatial Terms 43

3.3.1 Convection Term 47

3.3.2 Diffusion Term 50

3.3.3 Source Terms 53

3.4 Temporal Discretisation 54

3.5 Boundary Conditions 56

3.5.1 Boundary Conditions for the General Transport Equation . . . 56

3.5.2 Boundary Conditions for the Navier-Stokes Equations 58

3.6 Discretisation Errors on different types of meshes 59

3.6.1 Convection Term 61

3.6.2 Diffusion Term 65

3.7 Solution of Linear Equation Systems 67

3.8 Solution Algorithm for the Navier-Stokes System 69
3.8.1 Pressure Equation 69
3.8.2 Algorithms for Pressure-Velocity Coupling 72
3.9 Summary and Conclusions 73

4 Error Estimation **75**
4.1 Introduction 75
4.2 Literature Survey 76
4.2.1 Methods Used in FEM Analysis 76
4.2.2 Methods Used in FV Analysis 78
4.3 Error Transport Through a Face 83
4.4 Face Residual Error Estimator 87
4.4.1 Analysis of the Normalisation Practice 88
4.5 Examples 91
4.5.1 Planar Jet 91
4.5.2 Creeping Stagnation Flow 96
4.5.3 Convection Transport of Heat with a Distributed Heat Source 99
4.6 Summary and Conclusions 103

5 Mesh Adaptation **105**
5.1 Introduction 105
5.2 Literature Survey 105
5.3 Adaptation Procedure 109
5.3.1 Selection of Cells and Mesh Refinement 109
5.3.2 Solution Mapping Between Meshes 115
5.4 Examples 116
5.4.1 Planar Jet 116
5.4.2 Stokes Stagnation Flow 120
5.4.3 Convection Transport of Heat with a Distributed Heat Source 127
5.4.4 Convection and diffusion of a Temperature Profile without a Heat Source 129
5.5 Summary and Conclusions 135

6 Further Case Studies **137**
6.1 Introduction 137

6.2 Flow Over a Cavity . . . 137
6.3 S-shaped Pipe Bend . . . 143
6.4 Tube Bundle . . . 153
6.5 Wall-Mounted Cube . . . 162
6.6 Summary and Conclusions . . . 177

7 Adaptive-Polyhedral-Mesh Generation 179
7.1 Introduction . . . 179
7.2 Literature survey . . . 180
7.3 Voronoi Polygons and Delaunay Triangulation . . . 185
7.3.1 Algorithm for calculation of the Dirichlet Tessellation . . . 187
7.4 Computational mesh . . . 190
7.5 Polyhedral-Mesh Generation . . . 191
7.5.1 A comparison of accuracy on Quadrilateral, Polygonal and Triangular Meshes . . . 197
7.6 Mesh adaptation on Polyhedral Meshes . . . 206
7.6.1 Examples . . . 207
7.7 Summary and Conclusions . . . 212

8 Conclusions and Future Work 213
8.1 Introduction . . . 213
8.2 Error Estimation . . . 214
8.3 Mesh Adaptation . . . 214
8.4 Mesh Generation . . . 216
8.5 Future Work . . . 216

List of Figures

3.1 Computational cell 40
3.2 Mesh non-orthogonality 42
3.3 Mesh skewness 42
3.4 Variation of ϕ near the face 49
3.5 Non-orthogonality treatment 51
3.6 Boundary cell 56
3.7 Square mesh 59
3.8 Triangular mesh 60
3.9 Hexagonal mesh 60
3.10 Split-hexahedron mesh 60

4.1 Inconsistency of the interpolated values on the face 84
4.2 Distance between points on non-orthogonal mesh 90
4.3 Solution domain and boundary conditions for the jet case 92
4.4 Starting mesh for the jet case (10 x 4 cells) 93
4.5 Velocity field for the jet case [m/s] (80 x 32 mesh) 94
4.6 Pressure isobars for the jet case [m^2/s^2] (80 x 32 mesh) 94
4.7 Velocity-error field for the jet case [m/s] (80 x 32 mesh) 95
4.8 Variation of errors with uniform mesh refinement for the jet case ($|\mathbf{U}_{norm}| = 2.474 m/s$) 96
4.9 Solution domain and boundary conditions for the stagnation flow . . 97
4.10 Velocity and pressure for the stagnation flow (40 x 40 mesh) 98
4.11 Velocity-error fields for the stagnation flow [m/s] (40 x 40 mesh) . . 98
4.12 Variation of errors with uniform mesh refinement for the stagnation flow ($|\mathbf{U}_{norm}| = 1.107 m/s$) 99
4.13 Solution domain and boundary conditions for the convection transport case 100

4.14 Temperature and source fields for the convection transport case (40 x 40 mesh) . . . 101
4.15 Error fields for the convection transport case [^oC] (40 x 40 mesh) . . 101
4.16 Uniform mesh . . . 102
4.17 Variation of errors with uniform mesh refinement for the convection transport case ($T_{norm} = 1^o$C) . . . 102

5.1 A split-hexahedron cell with left face split in one direction shared with two cells. The top face is cross-split and shared with four cells . 110
5.2 Directional splitting of cells . . . 111
5.3 Refinement of split-hexahedron cells . . . 111
5.4 Node distances at a split face . . . 112
5.5 Additional splitting of cells . . . 113
5.6 Consistency over a split face in 2D (dotted lines represent the selected refinement) . . . 114
5.7 Consistency over a cross-split face in 3D (dotted lines represent the selected refinement) . . . 114
5.8 Treatment of incompatible cell splitting directions in 3D (dotted lines represent the selected refinement) . . . 115
5.9 Mesh after 6 cycles of refinement for the jet case (209 cells) . . . 116
5.10 Variation of velocity errors with adaptive refinement for the jet case ($|\mathbf{U}_{norm}| = 2.474m/s$) . . . 117
5.11 Velocity errors after 6 cycles of refinement for the jet case [m/s] (209 cells) . . . 118
5.12 Estimated errors on the faces of the final mesh with 209 cells (given as percentage of the maximum-estimated error on that mesh) . . . 119
5.13 Meshes for the creeping stagnation flow . . . 121
5.14 Velocity errors after 4 cycles of refinement for the creeping stagnation flow [m/s] . . . 122
5.15 Velocity-error scaling with adaptive refinement for the creeping stagnation flow ($|\mathbf{U}_{norm}| = 1.107m/s$) . . . 123
5.16 Estimated errors on the faces of the final mesh with 280 cells (given as percentage of the maximum-estimated error on that mesh) . . . 124
5.17 Meshes for the creeping stagnation flow (second calculation) . . . 125

5.18 Variation of velocity errors with adaptive mesh refinement for the stagnation flow (Second calculation) ($|\mathbf{U}_{norm}| = 1.107m/s$) 126
5.19 Velocity-error fields after 2 cycles of adaptive refinement for the stagnation flow (Second calculation) [m/s] 126
5.20 Mesh after 3 cycles of refinement for the convection transport case . . 127
5.21 Fields after 3 cycles of refinement 128
5.22 Variation of temperature errors with adaptive refinement for the convection transport case ($T_{norm} = 1^oC$) 129
5.23 Temperature field for the internal layer case [^{o}C] 129
5.24 Solution domain and boundary conditions for the convection and diffusion of heat 131
5.25 Variation of temperature errors with adaptive refinement for the convection and diffusion of heat ($T_{norm} = 1^oC$) 132
5.26 Temperature errors after 5 cycles of refinement for the convection and diffusion of heat [^{o}C] (1706 cells) 133
5.27 Temperature and its gradient for the convection and diffusion of heat 133
5.28 Mesh and errors for the calculation driven by the exact face errors . . 134

6.1 Geometry and boundary conditions for the flow over a cavity 138
6.2 Starting mesh for the flow over a cavity (36 cells) 138
6.3 Velocity field for the flow over a cavity on the final adapted mesh with 3257 cells (normalised by U_{avg}) 139
6.4 Pressure coefficient field for the flow over a cavity on the final adapted mesh with 3257 cells 140
6.5 Mesh after 8 cycles of refinement for the flow over a cavity (3257 cells) 140
6.6 Errors for the flow over a cavity on the final adapted mesh with 3257 cells (given as percentage of U_{avg}) 141
6.7 Variation of velocity errors with adaptive mesh refinement for the flow over a cavity (errors given as percentage of U_{avg}) 141
6.8 Estimated-velocity error on the faces of the final mesh with 3257 cells (given as percentage of the maximum estimated error on that mesh) . 142
6.9 Case setup for the S-bend case 144
6.10 Starting mesh for the S-bend case (270 cells) 145

6.11 Velocity field in for the S-bend case obtained on the final adapted mesh with 390787 cells (normalised by U_{avg}) 146
6.12 Pressure coefficient in the symmetry plane for the S-bend case obtained on the final adapted mesh with 390787 cells 147
6.13 Section at $X = 2D$ (390787 cells) 148
6.14 Variation of velocity errors with adaptive mesh refinement for the S-bend case (given as percentage of U_{max}) 149
6.15 Mesh after 9 cycles of refinement for the S-bend case (390787 cells) . 150
6.16 Velocity error in the symmetry plane after 9 cycles of refinement for the S-bend case (390787 cells) (given as percentage of U_{max}) 151
6.17 Geometry and boundary conditions for the tube bundle case 153
6.18 Starting mesh for the tube bundle case (640 cells) 154
6.19 Velocity field for the tube bundle case (17505 cells)(given as $\frac{\mathbf{U}}{U_{avg}}$) . . 155
6.20 Pressure coefficient for the tube bundle case (17505 cells) 156
6.21 q field for the tube bundle case (17505 cells)(given as $\frac{q}{U_{avg}}$) 156
6.22 ζ field for the tube bundle case (17505 cells)(given as $\frac{\zeta D}{Uavg^2}$) 156
6.23 Mesh after 7 cycles of refinement for the tube bundle case (17505 cells) 158
6.24 Variation of errors with adaptive mesh refinement for the tube bundle case (errors given as percentage of U_{max}, q_{max} and ζ_{max} respectively) . 159
6.25 Exact and estimated-error fields after 7 cycles of refinement (17505 cells)(errors are given as percentage of U_{max}, q_{max} and ζ_{max} respectively). Exact errors are calculated as the difference from the benchmark solution. Estimated errors are plotted as a weighted average of face errors. 160
6.26 A comparison of profiles for the tube bundle case 161
6.27 Geometry and boundary conditions for the wall mounted cube case . 162
6.28 Starting mesh for the wall-mounted cube case (3444 cells) 163
6.29 Distribution of flow variables in the symmetry plane for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells) 164
6.30 Distribution of flow variables in the plane $y/H = 0.5$ for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells) 166
6.31 Distribution of flow variables in the plane $x/H = 0.5$ for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells) 167

6.32 Streamlines in the plane $x/H = 2$ for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells) 168
6.33 Mesh after 7 cycles of refinement for the wall-mounted cube case (1161490 cells) 169
6.34 Variation of estimated errors with adaptive mesh refinement for the wall-mounted cube case (errors are given as percentage of U_{avg}, k_{max} at inlet and ϵ_{max} at inlet) 170
6.35 Remaining estimated errors after 7 cycles of refinement (errors are given as percentage of U_{avg}, k_{max} at inlet and ϵ_{max} at inlet, respectively) Estimated errors are plotted as a weighted average of face errors. 171
6.36 Regions of highest gradients after 7 cycles of refinement 172
6.37 Profiles of flow variables taken at $\frac{x}{H} = 0$, $\frac{z}{H} = -0.5$ (above the corner at which the leading edges meet) 173
6.38 C_p on the bottom wall 175
6.39 Velocity and turbulent energy k profiles 176

7.1 Delaunay Triangulation (solid lines) and Voronoi Polygons (dashed lines) for a set of points P_n (dots) 186
7.2 Delaunay vs other triangulations 186
7.3 Initial hull for the Delaunay triangulation 188
7.4 Dirichlet Tessellation and polyhedral mesh 191
7.5 Generation of an internal face of the polyhedral mesh (section in the plane of the face) 193
7.6 Tetrahedra sharing a face. The edge of the polyhedral mesh (thick lines) is perpendicular to the shared triangular face (red). There exist a polygonal face for every edge of the triangular face. 194
7.7 Generation of an internal face including the intersection of a boundary edge (section in the plane of the face) 195
7.8 Generation of boundary faces (coloured) including intersections with boundary edges (thick lines) and a corner point 195
7.9 An example of a 3D mesh for a cube 196
7.10 Meshes used for the jet case 198
7.11 Exact-velocity errors for the jet case [m/s] 199

7.12 Variation of the exact-velocity error and solution times on different types of meshes for the jet case. 200
7.13 Quadrilateral, polygonal and triangular meshes used for comparison . 202
7.14 Variation of the exact-velocity error on different types of meshes for the cavity case. Errors are given as percentage of average inlet velocity U_{avg}. 203
7.15 Magnitude of the exact-velocity error (given as percentage of average inlet velocity U_{avg}) 203
7.16 The exact-pressure error (given as percentage of $p_{max} = \frac{1.64}{2}\,\rho\,U_{avg}^2$) . . 204
7.17 Scaling of the pressure drop for different types of meshes 205
7.18 Adaptation of polyhedral meshes 206
7.19 Starting polygonal mesh for the convection and diffusion of heat . . . 208
7.20 Mesh after 9 cycles for the convection and diffusion of heat (5925 cells) 209
7.21 Polygonal mesh from nearly degenerate Delaunay Triangulation . . . 209
7.22 Errors after 9 cycles of refinement (5925 cells) 210
7.23 Uniform mesh and the exact error (6769 cells) 210
7.24 Variations of errors with adaptive refinement for the convection and diffusion of heat ($T_{norm} = 1^oC$) 211

List of Tables

5.1 Estimated errors on the faces of the final mesh with 209 cells (given as percentage of the maximum estimated error on that mesh) 119

5.2 Estimated errors on the faces of the final mesh with 280 cells (given as percentage of the maximum-estimated error on that mesh) 124

6.1 Distribution of face errors for the cavity case 142

6.2 Pressure-drop coefficients for the flow over a cavity 143

6.3 Pressure-drop coefficients C_p, force coefficients C_F and average vorticity coefficient C_ω for the S-bend case 152

6.4 Maximum of ϵ, velocity gradient, k field gradient and pressure gradient fields 168

6.5 Magnitude of the force acting on the cube and the lengths of vortices downstream and upstream of the cube 174

6.6 Distribution of estimated velocity error on the faces 177

7.1 Data structure for the triangulation 187

7.2 Relations between objects forming Delaunay Triangulation and Dirichlet Tessellation 192

7.3 Number of cells for different types of meshes (Jet case) 197

7.4 Number of cells for different types of meshes 201

7.5 A comparison of pressure drop for different types of meshes 204

7.6 Measures of mesh quality 209

Nomenclature

Latin Characters

$\mathbf{a}$ – general vector property

a_N – matrix coefficient corresponding to the neighbour N

a_P – central coefficient

C – constant dependent on the scheme for temporal discretisation

C_F – force coefficient

C_p – pressure coefficient

Co – Courant number

$\mathbf{d}$ – vector between P and N

$\mathbf{d}_n$ – vector between the cell centre and the boundary face

E – exact error, required error tolerance

e – total specific energy, solution error, truncation error

e_f – error on the face

F – mass flux through the face

F_{conv} – convection transport coefficient

F_{diff} – diffusion transport coefficient

F_{norm} – normalisation factor for the residual

f – face, point in the centre of the face

f_i – point of interpolation on the face

f_x – interpolation factor

g_b – boundary condition on the fixed gradient boundary

$\mathbf{g}$ – acceleration of the gravity force

$\mathbf{G}$ – matrix for Least-Squares Fit

$\vec{H}$ – transport part, Hessian matrix

h – mesh size

$\mathbf{I}$ – unit tensor

$\mathbf{k}$ – non-orthogonal part of the face area vector

k – turbulent kinetic energy

L – functional, set of edges

$\mathbf{m}$ – skewness correction vector, second moment

$\mathbf{M}$ – geometric moment of inertia, momentum

N – point in the centre of the neighbouring cell, number of cells

P – pressure, point in the centre of the cell, set of points

P – atmospheric pressure

$\mathbf{x}_{dist}$ – position difference vector

p – kinematic pressure, order of accuracy

q – q in the $q - \zeta$ turbulence model

Q_P – source for the system of linear equations

Q_V – body forces

$\mathbf{Q}_S$ – surface forces

Re – Reynolds number

r – ratio

Res_f – face residual

Res_f^P – face residual from the cell owner

Res_F^N – face residual from the cell neighbour

Res_P – cell residual

$\mathbf{S}$ – outward-pointing face area vector

$\mathbf{S}_f$ – face area vector

$\mathbf{s}$ – parametric curve

S_ϕ – source term

S_e – error source term

Sp – linear part of the source term

Su – constant part of the source term

S_{CV} – area of a control volume

T – temperature, time-scale

t – time

$\mathbf{U}$ – velocity

$\mathbf{U}_b$ – velocity of the arbitrary volume's face

V – volume

V_M – material volume

V_{CV} – control volume

V_i – Voronoi Polygon

V_P – volume of the cell

x – x component

$\mathbf{x}$ – position vector

y – y component

z – z component

Greek Characters

α – under-relaxation factor

α_N – non-orthogonality angle

Γ_ϕ – diffusivity

γ – blending factor

$\mathbf{\Delta}$ – orthogonal part of the face area vector

ϵ – dissipation rate of turbulent kinetic energy

ζ – effectivity index, ζ-field in the $q - \zeta$ turbulence model

κ – von Kármán constant

λ – heat conduction coefficient

μ – dynamic viscosity

ν – kinematic viscosity

ν_T – turbulent kinematic viscosity

ρ – density

σ – turbulent Prandtl number

σ – stress tensor

Φ – exact solution

ϕ – general scalar property

ψ – measure of mesh skewness

Superscripts

$\mathbf{q}^T$ – transpose

$\overline{q}$ – mean

q' – fluctuation around the mean value, shadow points

q^n – new time-level

q^{new} – new value

q^o – old time-level

q^{old} – old value

$\hat{\mathbf{q}}$ – unit vector

$\widetilde{q}$ – normalised, recovered value

Subscripts

q_f – value on the face

q_b – value on the boundary face

q_P – value in the cell owner

q_N – value in the cell neighbour, value in new point

q_t – turbulent

q_{real} – real flow

q_{govEqn} – exact solution of governing equations

q_{FV} – solution obtained by using the Finite Volume Method

$q_{iteration}$ – solution obtained by using an iterative procedure

$\underline{q}$ – interpolated value, approximated value

Abbreviations

ADT – Alternating Digital Tree

AFT – Advancing Front Method

Bi-CGSTAB – Bi-Conjugate Gradient Stabilised

CAD – Computer-Aided Design

CD – Central Differencing

CFD – Computational Fluid Dynamics

CG – Conjugate Gradient

CV – Control Volume

DNS – Direct Numerical Simulation

FD – Finite Difference Method

FEM – Finite Element Method

FV – Finite Volume

FVM – Finite Volume Method

FREE – Face Residual Error Estimator

ICCG – Incomplete Cholesky Conjugate Gradient

LES – Large Eddy Simulation

LSF – Least Squares Fit

LU – Lower-Upper

NVA – Normalised Variable Approach

NS – Navier-Stokes equations

PISO – Pressure-Implicit with Splitting of Operators

RANS – Reynolds Averaged Navier-Stokes

RE – Richardson Extrapolation

SIMPLE – Semi-Implicit Method for Pressure-Linked Equations

TDMA – Thomas algorithm

UD – Upwind Differencing

2D – Two-dimensional space

3D – Three-dimensional space

Chapter 1

Introduction

1.1 Background

Computational Fluid Dynamics (CFD) provides solutions to fluid flow problems by solving the governing equations on a computer. CFD has undergone rapid development in the last two decades, and problems which can be solved with it range from simple laminar flows to very complicated multi-phase flows, including heat exchange. Many of the existing CFD codes are coupled with CAD systems to make the design process easier and less expensive. As CFD is becoming an engineering tool its accuracy is gaining more and more importance, introducing the need for a reliable method for assessing and controlling accuracy.

The governing equations are of partial differential form, coupled in most cases. Closed-form solutions cannot be found except for some simple problems, which are not of much practical interest. The numerical methods used for CFD provide solutions by dividing the domain into smaller sub-domains, and assuming a certain variation of the dependent fields over each sub-domain. This, together with the conditions specified at the boundary of the original domain, generates a system of N algebraic equations with N unknowns for each dependent variable, N representing the number of sub-domains, which can be solved using a computer. The process of converting a differential equation into a system of algebraic equations is called discretisation. This process may introduce errors which can have a great influence on the quality of the results obtained.

There are many different discretisation practices. The most widely used ones are [48]: Finite Difference Method (FD), Finite Element Method (FEM) and Finite

Volume Method (FVM). The most popular discretisation practice in the CFD community is the Finite Volume (FV) Method and it is also adopted in this study. Finite Volume discretisation is performed using the integral formulation of the conservation laws, which are then discretised in physical space by assuming linear variation of the dependent variables over each discrete volume, called a control volume (CV). The nature of the FV discretisation practice allows the use of an arbitrary mesh, where control volumes can be of arbitrary topology consisting of general polyhedral volumes. The FV discretisation practice is conservative in its nature, and the quantities like momentum, mass, energy, *etc.* remain conserved during the calculation process.

CFD solutions may contain errors which can be divided into four main groups [48]:

- **Modelling errors** are defined as the difference between the real flow Φ_{real} and the exact solution of the governing equations Φ_{govEqn}, thus:

$$E_{modelling} = \Phi_{real} - \Phi_{govEqn}. \tag{1.1}$$

  Complex problems like turbulence, combustion and multi-phase flows are very difficult to describe exactly and require approximations in the process of deriving the governing equations. Initial and boundary conditions may also introduce errors if they are not known exactly. The geometry description can also introduce errors if it is not exact.

- **Discretisation errors** are defined as the difference between the exact solution of the differential equations Φ_{govEqn} and the exact solution of their discrete FV approximations ϕ_{FV}:

$$E_{discretisation} = \Phi_{govEqn} - \phi_{FV}. \tag{1.2}$$

  Discretisation errors originate from the discretisation approximations, the mesh density and quality, and the size of the time-step in transient calculations. A discretisation procedure, which can be partly characterised by its approximation order, can be exact in the parts of the flow where the variable distribution corresponds exactly to the discretisation assumptions and very inaccurate in the regions where the variables vary in different ways.

- **Iteration errors** are a group of errors which arise if the governing equations are solved using an iterative procedure. They are defined as the difference between the exact solution of the FV equations and the solution obtained by using an iterative procedure $\phi_{iteration}$, thus:

$$E_{iteration} = \phi_{FV} - \phi_{iteration}. \tag{1.3}$$

  These errors can be reduced to the level of computer truncation error for any given problem at the expense of time needed to complete the calculation.

- **Programming and User errors** are a result of the incorrect implementation or use of the CFD methodology in a computer code.

Error estimators are tools for estimation of the discretisation error in the numerical solution by using properties of the discretisation practice and the governing equations. They provide information about the discretisation-error distribution and its magnitude in some norm, and therefore measure the quality of the results in this respect.

The required discretisation accuracy is known before the analysis is performed. It depends on the objective of the calculations and on the accuracy of the differential equations which are used to describe the physics. Error estimators can be used as indicators of where and how to modify the mesh to achieve solutions of the required discretisation accuracy. This can be achieved by locally refining the mesh where the error is large and coarsening the mesh in the regions where the error is small, in order to maintain the error at the required level and equidistribute it over the computational domain. An adaptive procedure, used for achieving the required accuracy, should be composed of a number of cycles, each cycle consisting of solving equations using a current mesh and the discretisation practice, followed by error estimation and finally modification of the mesh.

The quality of a computational mesh is an important factor in minimising discretisation error [48]. The quality is influenced by spatial resolution, skewness and non-orthogonality of the mesh and also by the type of cells used.

The aim of the present study is to develop:

1. An accurate method for estimation of the discretisation error in the FV solution, which is applicable to different types of differential equations and for problems ranging from convection to diffusion-dominated ones.

2. A method capable of reducing the discretisation error below the required level without any user intervention, and with the smallest computational load by producing the optimum mesh for the given problem.

1.2 Present Contributions

This study has contributed to the field of Computational Fluid Dynamics in the following respects:

- A Face Residual Error Estimator (FREE) which estimates the error on the faces of the computational cells is developed. It is a result of an analysis of the discretisation error and its transport through cell faces. The estimator measures the error in the field under consideration by comparing the values extrapolated onto the face from the neighbouring nodes.

- A fully automatic mesh-adaptation procedure is set up and applied to several steady-state flows, both laminar and turbulent. The mesh is adapted by refining the cells sharing a face with large error by splitting them parallel to the face.

- A mesh generator for polyhedral meshes for the Finite Volume Method based on the Delaunay method is developed.

- A mesh-adaptation procedure for polyhedral meshes is developed. It is performed by enriching the Delaunay structure with vertices where more resolution is needed, resulting in local refinement.

- An analysis of discretisation errors on meshes consisting of squares, triangles, hexagons and split-hexahedra is presented. A comparison of the relative accuracy on quadrilateral, polygonal and triangular meshes is performed on laminar flow cases.

1.3 Thesis Outline

In Chapter 2, a summary of the governing equations of continuum mechanics can be found along with the Newtonian constitutive relations. A transport equation as a model equation is introduced. A brief description of turbulence modelling is given.

Chapter 3 presents the Finite Volume Method used in this study. It is a second-order method for arbitrary unstructured meshes. Discretisation of spatial terms in the transport equation is described term by term along with the errors which may arise. Temporal discretisation and the errors which may result from it are briefly discussed. An analysis of the discretisation error on different shapes of computational cells is performed. A solution algorithm for Navier-Stokes equations is presented at the end of the chapter.

Developments in the field of $a - posteriori$ error estimation made during this study are presented in Chapter 4. A literature survey of the existing methods is given first. A new method for error estimation is proposed. The performance of the proposed error estimator is tested on a set of cases with analytical solutions, including convection and diffusion-dominated ones.

In Chapter 5 a mesh-refinement procedure is proposed. A literature survey of mesh-adaptation methods is presented first, followed by the proposed mesh-refinement procedure based on directional cell-by-cell refinement of hexahedral cells. The performance of the refinement procedure is examined on a set of test cases for which analytical solutions are available.

In Chapter 6, the mesh-refinement procedure proposed in Chapter 5 is further tested on four cases of engineering interest, involving laminar and turbulent flows.

Chapter 7 presents an algorithm for polyhedral-mesh generation developed during this study. A survey of mesh-generation methods is given at the beginning of the chapter. It is followed by an algorithm assembled for calculating polyhedral meshes from the Delaunay Triangulation and the Voronoi Polygons which is described step by step. A mesh-adaptation technique for polyhedral meshes is also presented. A comparison of relative accuracy which can be achieved on triangular, quadrilateral and polygonal meshes is performed on cases introduced in earlier chapters. An example of the adaptive-mesh generation is also given.

Finally, a summary of the Thesis with some conclusions and suggestions for future work are given in Chapter 8.

Chapter 2

Governing Equations of Continuum Mechanics

2.1 Navier-Stokes Equations

The governing equations of Fluid Mechanics, the so-called Navier-Stokes equations, are a set of partial differential equations which read [112, 113]:

- Continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0, \tag{2.1}$$

- Momentum equation:

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) = \rho \mathbf{g} + \nabla \cdot \sigma, \tag{2.2}$$

  where $\mathbf{g}$ is the gravity acceleration and σ is a surface stress tensor.

- Energy equation:

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e \mathbf{U}) = \rho \mathbf{g}.\mathbf{U} + \nabla \cdot (\sigma \cdot \mathbf{U}) - \nabla \cdot \mathbf{q} + \rho Q. \tag{2.3}$$

  Here, $\mathbf{q}$ is the heat flux through the control-volume surface and Q is the heat source within the CV.

2.2 Constitutive Relations for Newtonian Fluids

The fluids treated in this study are assumed to obey the following constitutive relations [113]:

- Newton's law of viscosity

$$\sigma = -\left(P + \frac{2}{3}\,\mu\nabla\cdot\mathbf{U}\right)\mathbf{I} + \mu\left[\nabla\mathbf{U} + (\nabla\mathbf{U})^T\right] \tag{2.4}$$

  where P is the pressure, μ is dynamic viscosity and $\mathbf{I}$ is the unit tensor.

- The equation of state for the ideal gas

$$P = \rho RT, \tag{2.5}$$

  where R is a universal gas constant.

- Fourier law of heat conduction

$$\mathbf{q} = -\lambda\,\nabla T, \tag{2.6}$$

  λ being a heat conduction coefficient.

When the above relations are inserted into Eqs. (2.2) and (2.3) a closed system of equations is obtained, as follows [113]:

- Continuity equation:

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{U}) = 0, \tag{2.7}$$

- Momentum equation:

$$\begin{aligned}\frac{\partial\rho\mathbf{U}}{\partial t} + \nabla\cdot(\rho\mathbf{U}\mathbf{U}) = \rho\mathbf{g} - \nabla\left(P + \frac{2}{3}\,\mu\nabla\cdot\mathbf{U}\right)\\ +\nabla\cdot\left[\mu\left(\nabla\mathbf{U} + (\nabla\mathbf{U})^T\right)\right],\end{aligned} \tag{2.8}$$

- Energy equation:

$$\begin{aligned}\frac{\partial\rho e}{\partial t} + \nabla\cdot(\rho e\mathbf{U}) = \rho\mathbf{g}\cdot\mathbf{U} - \nabla\cdot(P\,\mathbf{U}) - \nabla\cdot\left(\frac{2}{3}\,\mu(\nabla\cdot\mathbf{U})\,\mathbf{U}\right)\\ +\nabla\cdot\left[\mu\left(\nabla\mathbf{U} + (\nabla\mathbf{U})^T\right)\cdot\mathbf{U}\right] + \nabla\cdot(\lambda\nabla T) + \rho Q,\end{aligned} \tag{2.9}$$

2.2.1 Turbulence Modelling

Turbulent flows occur in most engineering applications. There are many methods developed for prediction of such flows, which differ in the level of detail the flow is resolved [124].

Direct Numerical Simulation (DNS) is the most detailed approach to turbulence modelling and it solves the governing equations over the whole range of turbulent

scales. This approach requires high spatial and temporal resolution, demanding large computational resources and long simulation times, making DNS unsuitable for most engineering applications. Some examples can be found in [19, 86, 124, 131].

Large Eddy Simulation (LES) (Smagorinsky [138], Haworth and Jansen [59], a review by Piomelli [122]) is an approach where the large scale eddies are resolved and the small eddies are modelled. Therefore, this approach requires a spatial filter separating the large scales from the small ones. As the small eddies are usually much weaker and more isotropic than the large ones it makes sense to model them and to fully resolve the large ones, as they are the main transporters of the conserved properties. When the mesh size tends to zero, such that it can resolve the smallest eddies, LES tends to DNS.

Reynolds-averaged method (RANS), originally proposed by Osborne Reynolds, is a statistical approach to turbulence modelling. The rationale behind this approach is that the instantaneous quantity $\phi(\mathbf{x}, t)$ in a certain point in the domain can be written as the sum of an averaged value and a fluctuation about that value, thus:

$$\phi(\mathbf{x}, t) = \overline{\phi}(\mathbf{x}, t) + \phi'(\mathbf{x}, t), \tag{2.10}$$

where $\phi'(\mathbf{x}, t)$ denotes turbulent fluctuations and $\overline{\phi}(\mathbf{x}, t)$ is the averaged value. There are three main techniques for calculating the averaged value namely time averaging, space averaging and ensemble averaging Hinze [61].

Depending on whether the flow is incompressible or compressible, averaging can be unweighted namely Reynolds averaging, or density weighted named Favre averaging (*eg.* Favre [46], Cebeci and Smith [34]).

When the above averaging is applied to the momentum and the continuity equations for incompressible isothermal flow without body forces, Eqs. (2.1) and (2.8), there results:

$$\nabla \cdot \overline{\mathbf{U}} = 0, \tag{2.11}$$

$$\frac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}}\,\overline{\mathbf{U}} + \overline{\mathbf{U}'\,\mathbf{U}'}) - \nabla \cdot \left(\nu \nabla \overline{\mathbf{U}}\right) = -\nabla \overline{p}. \tag{2.12}$$

where the term $\overline{\mathbf{U}'\,\mathbf{U}'}$, called kinematic Reynolds stress tensor, is the only term containing $\mathbf{U}'$. In order to link the kinematic Reynolds stress with the mean-flow variables, modelling approximations have to be introduced and they are usually called *turbulence models*.

The turbulence models used in this study are based on the Boussinesq approximation [27], which assumes that turbulent stresses are linked to the averaged-flow variables as follows:

$$\overline{\mathbf{U}'\mathbf{U}'} = -\nu_t \left(\nabla\overline{\mathbf{U}} + (\nabla\overline{\mathbf{U}})^T\right) + \frac{2}{3}\,k\,\mathbf{I}, \tag{2.13}$$

where k stands for the kinetic energy of turbulence defined as:

$$k = \frac{1}{2}\overline{\mathbf{U}' \cdot \mathbf{U}'}. \tag{2.14}$$

The turbulent eddy viscosity can be calculated in many ways but the most popular one is a "two-equation" approach where the ν_t is defined as:

$$\nu_t = C_\mu \frac{k^2}{\epsilon}, \tag{2.15}$$

where ϵ is the kinematic turbulence dissipation rate, defined as [157]:

$$\epsilon = \nu\overline{\nabla\mathbf{U}' \,:\, \nabla\mathbf{U}'} \tag{2.16}$$

The variables k and ϵ are calculated as the solution of their own transport equations. The equation for the kinematic turbulent dissipation ϵ has the following form [124]:

$$\nabla \cdot (\overline{\mathbf{U}}\epsilon) - \nabla \cdot ((\frac{\nu_T}{\sigma_\epsilon} + \nu)\nabla\epsilon) = C_1 \frac{P\epsilon}{k} - C_2 \frac{\epsilon^2}{k}, \tag{2.17}$$

and the equation for the turbulent kinetic energy k reads [124]:

$$\nabla \cdot (\overline{\mathbf{U}}k) - \nabla \cdot ((\frac{\nu_T}{\sigma_k} + \nu)\nabla k) = P - \epsilon. \tag{2.18}$$

The production term P in the above equation has the following form:

$$P = 2\nu_T \frac{(\nabla\overline{\mathbf{U}} + \nabla\overline{\mathbf{U}}^T)}{2} : \frac{(\nabla\overline{\mathbf{U}} + \nabla\overline{\mathbf{U}}^T)}{2}. \tag{2.19}$$

The values of the coefficients are: $C_\nu = 0.09$, $C_1 = 1.44$, $C_2 = 1.92$, $\sigma_k = 1.0$ and $\sigma_\epsilon = 1.3$.

In the vicinity of the impermeable no-slip walls physics of the turbulence is dominated by the presence of the wall. The most general treatment for resolving the flow near the wall is by solving the transport equations in the near wall region, *eg.* Launder and Sharma [84]. However, as the large variations of flow variables exist in the near-wall region, the computational mesh has to be very fine there. A model

developed to alleviate these problems is "$q-\zeta$" [54] where q and ζ vary linearly next to the wall. q and ζ are defined as [54]:

$$q = \sqrt{k} \text{ and} \tag{2.20}$$

$$\zeta = \frac{\epsilon}{2q}. \tag{2.21}$$

The equations for ζ and q read [54]:

$$\nabla \cdot (\overline{\mathbf{U}}\zeta) - \nabla \cdot ((\frac{\nu_T}{\sigma_\zeta} + \nu)\nabla\zeta) = (2C_1 - 1)\, G\, \frac{\zeta}{q} - (2C_2 - 1)\, f_2\, \frac{\zeta^2}{q} + E, \tag{2.22}$$

$$\nabla \cdot (\overline{\mathbf{U}}q) - \nabla \cdot ((\frac{\nu_T}{\sigma_q} + \nu)\nabla q) = G - \zeta. \tag{2.23}$$

The f_2 function is defined as follows:

$$f_2 = 1 - 0.3e^{-(Re_\tau)^2}. \tag{2.24}$$

The source terms E and G in the above equations are calculated from the velocity field as follows:

$$E = \frac{\nu\nu_T}{q}(\nabla\nabla\overline{\mathbf{U}}) : (\nabla\nabla\overline{\mathbf{U}}), \tag{2.25}$$

$$G = \frac{\nu_T}{q}(\frac{\nabla\overline{\mathbf{U}} + (\nabla\overline{\mathbf{U}})^T}{2}) : (\frac{\nabla\overline{\mathbf{U}} + (\nabla\overline{\mathbf{U}})^T}{2}). \tag{2.26}$$

The turbulent viscosity ν_T is calculated from q and ζ using:

$$\nu_T = C_\nu f_\nu \frac{q^3}{2\zeta}, \tag{2.27}$$

where the f_ν is a damping function:

$$f_\nu = e^{\left(\frac{-6}{(1+\frac{Re_\tau}{50})^2}\right)} \left(1 + 3e^{-\frac{Re_\tau}{10}}\right), \tag{2.28}$$

and the turbulence Reynolds number Re_τ is defined as:

$$Re_\tau = \frac{q^3}{2\nu\zeta}. \tag{2.29}$$

The values of the constants are: $C_\nu = 0.09$, $C_1 = 1.44$, $C_2 = 1.92$, $\sigma_\zeta = 1.3$ and $\sigma_q = 1$.

For most engineering problems it is often too expensive to resolve the boundary layer next to the wall using low-Re turbulence models. An alternative is to model it by using wall-functions (*eg.* Launder and Spalding [85], Ferziger and Perić [47]), which mimic the behaviour of the turbulent boundary layer near the wall by

assuming that it behaves as the turbulent boundary layer near a flat plate. The wall-functions rely on the existence of a logarithmic region in the velocity profile of a turbulent boundary layer. The wall-function has the following form [47]:

$$U^+ = \frac{\overline{U_t}}{U_\tau} = \frac{1}{\kappa} \ln Y^+ + B, \tag{2.30}$$

where $\overline{U_t}$ is the mean velocity parallel to the wall, U_τ defined as $U_\tau = \sqrt{\tau_w/\rho}$ is the shear velocity, τ_w is the shear stress at the wall, κ is the von Kármán constant ($\kappa = 0.41$), B is an empirical constant which depends on the thickness of the viscous sublayer ($B \approx 5.2$ in a flat plate boundary layer) and Y^+ is the dimensionless distance from the wall defined as:

$$Y^+ = C_\nu^{0.25} \frac{d\sqrt{k}}{\nu} \tag{2.31}$$

Here, d is the distance from the wall. The wall-function is valid when the near-wall node is within the logarithmic region, *i.e.* $Y^+ > 15$. This imposes a limitation on mesh resolution in the near-wall region characterised by high gradients of all fields which need fine mesh resolution to achieve accurate solutions of the governing equations. On the other hand, if the mesh becomes too fine ($Y^+ < 15$), the wall-function described above becomes invalid. If this is present over a large portion of wall boundaries it may result in serious modelling errors.

2.3 General Form of a Transport Equation

All equations described above can be written in the form of a general transport equation, given below and used throughout this study to present the FV discretisation practices and error analysis.

$$\underbrace{\int_{V_{CV}} \frac{\partial \rho\phi}{\partial t}\, dV}_{\text{temporal derivative}} + \underbrace{\int_{V_{CV}} \nabla \cdot (\rho \mathbf{U}\phi)\, dV}_{\text{convection term}} - \underbrace{\int_{V_{CV}} \nabla \cdot (\rho \Gamma_\phi \nabla\phi)\, dV}_{\text{diffusion term}} = \underbrace{\int_{V_{CV}} S_\phi(\phi)\, dV.}_{\text{source term}} \tag{2.32}$$

Here ϕ is a tensorial property considered continuous in space, Γ_ϕ is the diffusion coefficient and $S_\phi(\phi)$ is the source term.

2.4 Summary

In this chapter the laws of the continuum mechanics have been presented. An introduction into turbulence modelling, used in this study, is also given. Low-Re

turbulence models solve the turbulence equations in the near-wall region where they require fine mesh resolution to resolve sharp gradients of solution variables, but they do not impose a limit on mesh resolution there. High-Re turbulence models model the flow near wall boundaries by using wall-functions which reduce the number of cells required, but they impose a limit on mesh resolution there which may prevent the user from getting a mesh-independent solution of the problem under consideration. The general transport equation which will be used for explaining FV discretisation and error analysis is presented at the end of the chapter.

Chapter 3

Finite Volume Discretisation

3.1 Introduction

The Finite Volume discretisation used in this study will be described in this chapter by using the general transport equation Eqn. (2.32), introduced in the previous chapter, as the model. The FV discretisation of this equation will be performed term by term and the resulting discretisation errors which can arise will be identified. The boundary conditions and their influence on the accuracy will also be discussed. A solution algorithm for solving the Navier-Stokes equations will be presented at the end of the chapter.

An important property required of a FV discretisation practice is that the flow solution is sought at a certain number of nodes in space and time; and if the number of nodes tends to infinity then the solution should tend to the exact solution of the governing equations. This will happen if the FV method satisfies the following requirements [48]:

- **Consistency.** The discretisation error in the numerical solution must tend to zero as the grid spacing tends to zero. The discretisation can produce the exact solution if the truncation error, defined as the difference between the governing equation and its discrete approximation, tends to zero when the grid spacing tends to zero. The truncation error can be expressed as a power of the grid size and/or time step where the power of the most important term represents the order of the approximation. The order must be positive and, if possible, equal or higher than the order of the differential equation [48].

- **Stability.** The discretisation is considered stable if it does not magnify nu-

merical errors during the calculation process. Stability is dependent on the discretisation practice, but it is also dependent on the employed solution algorithm.

- **Conservation.** The FV equations are obtained from the integral form of the governing equations and therefore for conserved properties they should obey the same conservation laws as their parent differential equations. This must apply for every CV and for the whole domain.

- **Boundedness.** Some physical properties (*eg.* density, turbulence energy, mass fraction) lie between certain bounds. For example, none of these quantities can be negative; and mass fraction is bounded by one from above. All convection schemes of order higher than one can produce unphysical values if the computational mesh is too coarse. Only some first-order schemes guarantee boundedness [48]. Unphysical values of the bounded properties may also cause failure of the solution algorithms (*i.e.* negative values of ϵ give negative turbulent viscosity which causes instability in the momentum equation).

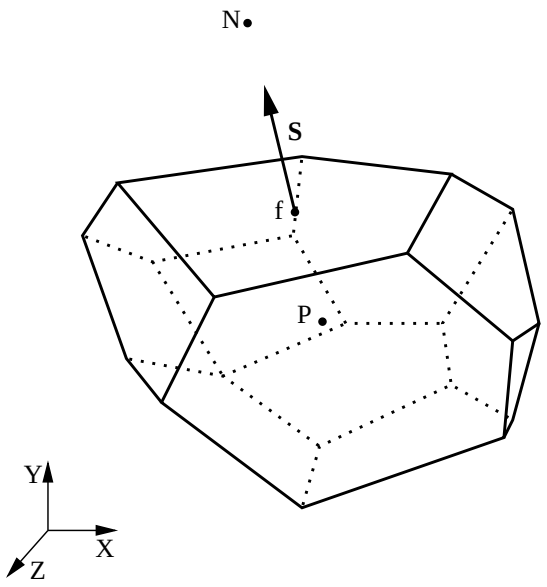


Figure 3.1: Computational cell

The Finite Volume discretisation process consists of two consecutive steps. The first step is the discretisation of the physical domain into contiguous computational cells, which can in general move in space with time and are convex polyhedra which do not overlap. All cells together form a computational mesh which is a discrete

representation of the computational domain. For each cell the values of the fields are stored at a node P, Fig. 3.1, located in the centroid of the cell $\mathbf{x}_P$, defined as:

$$\int_{V_P} (\mathbf{x} - \mathbf{x}_P) dV = 0. \tag{3.1}$$

Every cell shares an internal face with a neighbouring cell whose centroid is denoted with N in Fig. 3.1. Faces which are not shared by two cells are boundary faces.

The values of the fields defined on the faces (*i.e.* face flux, surface-normal gradients) are stored in the node located in the centroid of the face $\mathbf{x}_f$, whose position is given by:

$$\int_f d\mathbf{S} \, (\mathbf{x} - \mathbf{x}_f) = 0 \tag{3.2}$$

The second step of the FV discretisation process is the approximation of the governing equations over the typical cell, here done in a second-order fashion by assuming a linear variation of the property ϕ within each CV and during each time-step. This can be expressed via the Taylor Series expansion:

$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P + O(|(\mathbf{x} - \mathbf{x}_P)|^2), \tag{3.3}$$

$$\phi(t + \Delta t) = \phi^t + \Delta t \left(\frac{\partial \phi}{\partial t} \right)^t + O(\Delta t^2), \tag{3.4}$$

where the subscript P relates to the node in which the solution is sought and the superscript t denotes the current time step. $O(|(\mathbf{x} - \mathbf{x}_P)|^2)$ and $O(\Delta T^2)$ are the truncated terms in the full series, having the following form:

$$O(|(\mathbf{x} - \mathbf{x}_P)|^2) = \sum_{i=2}^{\infty} \frac{1}{i!} (\mathbf{x} - \mathbf{x}_P)^i \underbrace{:::}_{i} (\underbrace{\nabla\nabla..\nabla}_{i}\phi)_P, \tag{3.5}$$

$$O(\Delta t^2) = \sum_{i=2}^{\infty} \frac{1}{i!} \Delta t^n \frac{\partial^i \phi}{\partial t^i}, \tag{3.6}$$

where $\underbrace{:::}_{i}$ is a scalar product of ith rank tensors. $(\mathbf{x} - \mathbf{x}_P)^i$ is a ith tensor product of a vector with itself resulting in an ith rank tensor. The leading terms of the truncation errors are proportional to $(\mathbf{x} - \mathbf{x}_P)^2$ and Δt^2, so the approximations are of second-order accuracy.

3.2 Measures of Mesh Quality

The distribution of the nodes and the quality of the mesh influence the accuracy of results. The properties which determine mesh quality and their measures are

presented here, and they will be used when discussing different discretisation errors later in this chapter. These properties are defined for mesh faces and their definition is the same irrespective of the mesh type. Finally, the properties are:

- **Non-orthogonality** is measured by the angle α_N between the vector $\mathbf{d}$ and the face area vector $\mathbf{S}$, see Fig. 3.2. The vector $\mathbf{d}$ is defined as follows:

$$\mathbf{d} = \mathbf{x}_N - \mathbf{x}_P. \tag{3.7}$$

  The angle should be as small as possible. The reasons for this will be given later in this chapter.

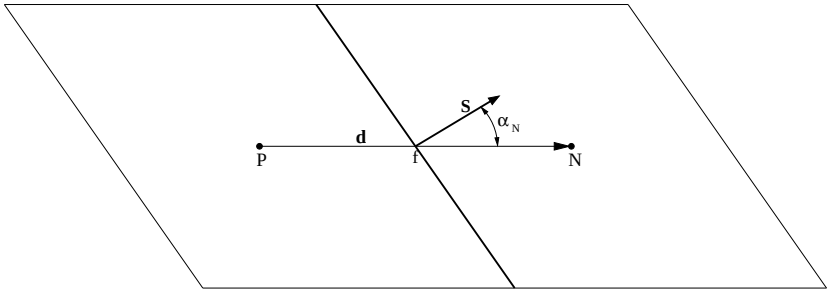


Figure 3.2: Mesh non-orthogonality

- **Mesh skewness.** When the vector $\mathbf{d}$ does not intersect a face in its centre the mesh is defined as skewed, Fig. 3.3. The degree of skewness can be measured

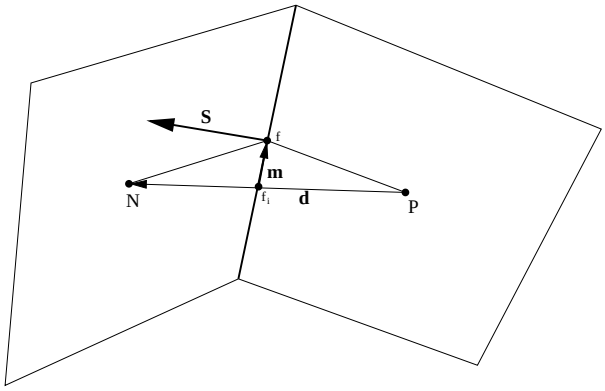


Figure 3.3: Mesh skewness

by:

$$\psi = \frac{|\mathbf{m}|}{|\mathbf{d}|}. \tag{3.8}$$

Here $\mathbf{m}$ is defined as:

$$\mathbf{m} = \mathbf{x}_f - \mathbf{x}_{f_i}, \tag{3.9}$$

where $\mathbf{x}_{f_i}$ denotes a point at which the vector $\mathbf{d}$ intersects the face, see Fig. 3.3. Skewness affects the accuracy of the interpolation from the nodes onto the faces as will be shown in the remainder of the chapter.

- **Uniformity.** A mesh is uniform when $\mathbf{d}$ intersects the face midway between the nodes P and N, Fig. 3.3. Uniformity can be measured by:

$$f_x = \frac{|\mathbf{x}_{f_i} - \mathbf{x}_N|}{|\mathbf{d}|}, \tag{3.10}$$

thus $f_x = 0.5$ on uniform meshes. The influence of uniformity on accuracy will be discussed later in the chapter.

3.3 Discretisation of Spatial Terms

The FV approximation of the spatial terms in Eqn. (2.32) will be given in this section. Approximations of volume integrals, surface integrals and interpolation techniques which are needed for the FV discretisation of the spatial terms in Eqn. (2.32) will be given first.

Volume integrals of ϕ can be approximated by integrating Eqn. (3.3) over the cell and using Eqn. (3.1) [69]:

$$\int_{V_P} \phi(\mathbf{x})\, dV = \phi_P V_P + O(|(\mathbf{x} - \mathbf{x}_P)|^2) \tag{3.11}$$

where V_P stands for the volume of the cell and ϕ_P is the value of ϕ in the centroid. Surface integrals can be evaluated in the similar fashion, thus [69]:

$$\int_{S_f} \phi(\mathbf{x})\, dS = \phi_f S_f + O(|(\mathbf{x} - \mathbf{x}_f)|^2), \tag{3.12}$$

$$\int_{S_f} d\mathbf{S} \cdot \mathbf{a}(\mathbf{x}) = \mathbf{S}_f \cdot \mathbf{a}_f + O(|(\mathbf{x} - \mathbf{x}_f)|^2), \tag{3.13}$$

where ϕ_f and $\mathbf{a}_f$ are the values of tensorial property ϕ and vector property $\mathbf{a}$ in the centroid of the face defined in Eqn. (3.2). Values in the face centroids can be interpolated or extrapolated from nodal values and are denoted by $\underline{\phi_f}$ and $\underline{\mathbf{a}_f}$.

A second-order interpolation practice from the nodes onto internal faces can be written as follows [20, 91]:

$$\underline{\phi_f} = \underline{\phi_{f_i}} + \mathbf{m} \cdot \underline{(\nabla\phi)_{f_i}}, \tag{3.14}$$

where $\underline{\phi_{f_i}}$ and $\underline{(\nabla\phi)_{f_i}}$ are the interpolated values of ϕ and $\nabla\phi$ at the point where the vector $\mathbf{d}$ intersects the face, as shown in Fig. 3.3. $\underline{\phi_{f_i}}$ and $\underline{(\nabla\phi)_{f_i}}$ can be evaluated by using linear interpolation:

$$\underline{\phi_{f_i}} = f_x\phi_P + (1 - f_x)\phi_N, \tag{3.15}$$

$$\underline{(\nabla\phi)_{f_i}} = f_x(\nabla\phi)_P + (1 - f_x)(\nabla\phi)_N. \tag{3.16}$$

The linear interpolation factor f_x is defined as follows, Fig. 3.3:

$$f_x = \frac{|\mathbf{x}_{f_i} - \mathbf{x}_N|}{|\mathbf{d}|}. \tag{3.17}$$

The truncation error for the interpolation practice defined in Eqn. (3.14) can be estimated by using the following Taylor expansions:

$$\phi_P = \phi_{f_i} + (\mathbf{x}_P - \mathbf{x}_{f_i}) \cdot (\nabla\phi)_{f_i} + \frac{1}{2}(\mathbf{x}_P - \mathbf{x}_{f_i})^2 : (\nabla\nabla\phi)_{f_i}, \tag{3.18}$$

$$\phi_N = \phi_{f_i} + (\mathbf{x}_N - \mathbf{x}_{f_i}) \cdot (\nabla\phi)_{f_i} + \frac{1}{2}(\mathbf{x}_N - \mathbf{x}_{f_i})^2 : (\nabla\nabla\phi)_{f_i}, \tag{3.19}$$

$$\phi_f = \phi_{f_i} + \mathbf{m} \cdot (\nabla\phi)_{f_i} + \frac{1}{2}\mathbf{m}^2 : (\nabla\nabla\phi)_{f_i}. \tag{3.20}$$

By substituting ϕ_P and ϕ_N in Eqn. (3.15) with Eqn. (3.18) and Eqn. (3.19), respectively, the truncation error for linear interpolation from Eqn. (3.15) can be obtained as [48]:

$$\begin{aligned} e_l &= \phi_{f_i} - \underline{\phi_{f_i}} \\ &= -\frac{1}{2}|\mathbf{x}_P - \mathbf{x}_{f_i}||\mathbf{x}_N - \mathbf{x}_{f_i}|\,(\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) \\ &= -\frac{1}{2}f_x(1 - f_x)|\mathbf{d}|^2\,(\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) \end{aligned} \tag{3.21}$$

$\hat{\mathbf{d}}$ being an unit vector in the direction of $\mathbf{d}$, Fig. 3.3.

From Eqn. (3.21) it follows that the truncation error for the gradient interpolated using Eqn. (3.16) has the form:

$$(\nabla e)_l = (\nabla\phi)_{f_i} - \underline{(\nabla\phi)_{f_i}} = -\frac{1}{2}f_x(1 - f_x)|\mathbf{d}|^2\,(\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_{f_i}) \tag{3.22}$$

Taking the difference between the Eqn. (3.20) and Eqn. (3.14), the truncation error for the linear interpolation scheme, which is of second-order on every mesh, can be

obtained in the following form:

$$\begin{aligned} e_{interpolation} &= -\frac{1}{2}|\mathbf{x}_P - \mathbf{x}_{f_i}||\mathbf{x}_N - \mathbf{x}_{f_i}| \left((\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) + \mathbf{m}\cdot(\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_{f_i})\right) \\ &+ \frac{1}{2}|\mathbf{m}|^2\,\hat{\mathbf{m}}^2 : (\nabla\nabla\phi)_{f_i} \\ &= -\frac{1}{2}f_x(1-f_x)|\mathbf{d}|^2 \left((\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) + \mathbf{m}\cdot(\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_{f_i})\right) \\ &+ \frac{1}{2}|\mathbf{m}|^2\,\hat{\mathbf{m}}^2 : (\nabla\nabla\phi)_{f_i} \\ &= -\frac{1}{2}|\mathbf{d}|^2 \left(f_x(1-f_x)(\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) + \psi|\mathbf{d}|\,\hat{\mathbf{m}}\cdot(\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_{f_i})\right) \\ &+ \frac{\psi^2}{2}|\mathbf{d}|^2(\hat{\mathbf{m}}^2 : (\nabla\nabla\phi)_{f_i}). \end{aligned} \tag{3.23}$$

Here, $\hat{\mathbf{d}}$ and $\hat{\mathbf{m}}$ are unit vectors in the directions of $\mathbf{d}$ and $\mathbf{m}$, respectively. This error reduces with the square of the distance between the neighbouring nodes and is minimal when the mesh is not skewed ($\psi = 0$).

The gradient and divergence terms can be approximated by using the Gauss-divergence theorem [69]:

- **Divergence** of the vector property $\mathbf{a}$ can be approximated as follows:

$$\begin{aligned} \int_V \nabla\cdot\mathbf{a}\,dV &= \oint_{S_{CV}} d\mathbf{S}\cdot\mathbf{a} \\ &= \sum_f \int_{S_f} d\mathbf{S}\cdot\mathbf{a} \\ &= \sum_f \mathbf{S}_f\cdot\mathbf{a}_f. \end{aligned} \tag{3.24}$$

  Here, V represents the volume of the CV and S_{CV} its surface area. $d\mathbf{S}$ is the surface area vector pointing outwards and $\mathbf{a}_f$ is evaluated using the interpolation practice defined in Eqn. (3.14) for the vector property $\mathbf{a}$.

  The truncation error for the divergence term consists of the error in the interpolation of $\mathbf{a}_f$, thus:

$$\begin{aligned} e_{div} &= \sum_f \mathbf{S}_f\cdot(\mathbf{a}_f - \underline{\mathbf{a}_f}) \\ &= \sum_f \mathbf{S}_f\cdot e_{interpolation} \\ &= \sum_f -\frac{1}{2}|\mathbf{d}|^2\mathbf{S}_f\cdot\left(f_x(1-f_x)(\hat{\mathbf{d}}^2 : (\nabla\nabla\mathbf{a})_{f_i}) + \psi|\mathbf{d}|\,\hat{\mathbf{m}}\cdot(\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\mathbf{a})_{f_i})\right) \\ &+ \sum_f \frac{\psi^2}{2}|\mathbf{d}|^2\mathbf{S}_f\cdot(\hat{\mathbf{m}}^2 : (\nabla\nabla\mathbf{a})_{f_i}). \end{aligned} \tag{3.25}$$

This error reduces with the square of $\mathbf{d}$ and is smallest in case when $\psi = 0$. The error is also dependent on the shape of the CV. This will be discussed in Section 3.6.

- **Gradient term.** Discretisation of the gradient term can be performed either using the Gauss-divergence theorem or the Least Squares Fit (LSF). Discretisation using the Gauss-divergence theorem can be written as follows [69]:

$$\begin{aligned}\int_{V_P} \nabla\phi \, dV &= \oint_{S_C V} d\mathbf{S}\, \phi \\ &= \sum_f \int_{S_f} d\mathbf{S}\, \phi \\ &= \sum_f d\mathbf{S}_f \, \phi_f, \end{aligned} \tag{3.26}$$

where ϕ_f is evaluated using the interpolation practice defined in Eqn. (3.14). The truncation error for this term is:

$$\begin{aligned}\mathbf{e}_{grad} &= \sum_f \mathbf{S}_f(\phi_f - \underline{\phi_f}) \\ &= \sum_f \mathbf{S}_f \, e_{interpolation} \\ &= \sum_f -\frac{1}{2}|\mathbf{d}|^2 \, \mathbf{S}_f \left(f_x(1-f_x)(\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) + \psi|\mathbf{d}| \, \hat{\mathbf{m}} \cdot (\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_{f_i}) \right) \\ &+ \sum_f \frac{\psi^2}{2}|\mathbf{d}|^2 \, \mathbf{S}_f(\hat{\mathbf{m}}^2 : (\nabla\nabla\phi)_{f_i}). \end{aligned} \tag{3.27}$$

Everything said about errors for the divergence term can be applied here.

The LSF is based on minimising the following functional [94]:

$$L = \sum_f \frac{1}{|\mathbf{d}|^2} \left(\phi_N - (\phi_P + \mathbf{d} \cdot (\nabla\phi)_P)\right)^2, \tag{3.28}$$

with respect to the gradient $(\nabla\phi)_P$. When

$$\frac{dL}{d(\nabla\phi)_P} = 0 \tag{3.29}$$

it results in a system of equations from which the cell gradient can be calculated, such as:

$$\mathbf{G} \, \nabla\phi_P = \sum_f \frac{\mathbf{d}(\phi_N - \phi_P)}{|\mathbf{d}|^2}, \tag{3.30}$$

where matrix G is defined as:

$$\mathbf{G} = \sum_f \frac{\mathbf{dd}}{|\mathbf{d}|^2}. \tag{3.31}$$

The foregoing results will now be used to obtain the FV form of the individual terms in Eqn. (2.32).

3.3.1 Convection Term

The discretisation of the convection term is performed using Eqn. (3.24) in the following fashion:

$$\begin{aligned}\int_{V_P} \nabla \cdot (\rho \mathbf{U}\phi)\, dV &= \sum_f \mathbf{S} \cdot (\rho \mathbf{U}\phi)_f \\ &= \sum_f \mathbf{S} \cdot (\rho \mathbf{U})_f\, \phi_f \\ &= \sum_f F\, \phi_f, \end{aligned} \tag{3.32}$$

where the volume integral is first transformed into a sum over the faces and then approximated. Here F represents the mass flux through the face:

$$F = \mathbf{S} \cdot (\rho \mathbf{U})_f. \tag{3.33}$$

These fluxes have to satisfy continuity for every CV. They can be estimated using the interpolated values of $\mathbf{U}$ and ρ onto the face. This interpolation may introduce an error into the mass flux which can then be written:

$$F = (\mathbf{S} \cdot \underline{(\rho \mathbf{U})_f}) + \mathbf{e}_{flux}. \tag{3.34}$$

The procedure for obtaining conservative fluxes and errors which can arise will be described in Section 3.8.

The next issue is how to obtain the value of ϕ on the face. Many different interpolation techniques can be used to obtain ϕ_f but some do not ensure boundedness.

- **Linear Interpolation (Central Differencing)** (CD) is a natural second-order interpolation practice for obtaining the value of ϕ on the face. This practice has already been described in Eqn. (3.14) and is:

$$\phi_f = (f_x \phi_P + (1 - f_x)\phi_N + \mathbf{m} \cdot \underline{(\nabla \phi)_{f_i}}) + e_{conv}. \tag{3.35}$$

  The truncation error, defined in Eqn. (3.23), is:

$$\begin{aligned} e_{CD} = -\frac{1}{2}|\mathbf{d}|^2 \Big(f_x(1 - f_x)(\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) + \psi |\mathbf{d}|\, \hat{\mathbf{m}} \cdot (\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_{f_i}) \Big) \\ + \frac{\psi^2}{2}|\mathbf{d}|^2 (\hat{\mathbf{m}}^2 : (\nabla\nabla\phi)_{f_i}). \end{aligned} \tag{3.36}$$

In [62, 115, 120, 149] it is shown that with CD the convective contribution to the coefficients of downstream nodes is always negative, which may give rise to non-physical oscillations which violate boundedness and degrade the quality of the results. On convection-diffusion problems this undesired behaviour may occur if the value of face Peclet Number is greater than two. This undesirable property may not become apparent if the gradients in the solution are not large, but if boundedness of the solution is essential (as in solutions of the *eg.* k and ϵ in turbulence equations) some other scheme may have to be used.

- **Upwind Differencing** (UD) was introduced to overcome the problem of oscillatory solutions, and it makes the convection term unconditionally positive. Boundedness of the convection term is achieved by assuming that the value on the face is determined by the upstream node, thus:

$$\phi_{f_i} = \begin{cases} \underline{\phi_{f_i}} = \phi_P & \text{for } F > 0. \\ \underline{\phi_{f_i}} = \phi_N & \text{for } F < 0. \end{cases} \tag{3.37}$$

This discretisation practice ensures the boundedness of the solution by making the matrix coefficients unconditionally positive.

The truncation error for the UD scheme can be obtained by using the following Taylor series expansions:

$$\phi_f = \begin{cases} \phi_P + (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla\phi)_P + \frac{1}{2}(\mathbf{x}_f - \mathbf{x}_P)^2 : (\nabla\nabla\phi)_P & \text{for } F > 0. \\ \phi_N + (\mathbf{x}_f - \mathbf{x}_N) \cdot (\nabla\phi)_N + \frac{1}{2}(\mathbf{x}_f - \mathbf{x}_N)^2 : (\nabla\nabla\phi)_N & \text{for } F < 0. \end{cases} \tag{3.38}$$

The truncation error can be found as a difference between Eqn. (3.38) and Eqn. (3.37), thus:

$$e_{UD} = \begin{cases} (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla\phi)_P + \frac{1}{2}(\mathbf{x}_f - \mathbf{x}_P)^2 : (\nabla\nabla\phi)_P & \text{for } F > 0. \\ (\mathbf{x}_f - \mathbf{x}_N) \cdot (\nabla\phi)_N + \frac{1}{2}(\mathbf{x}_f - \mathbf{x}_N)^2 : (\nabla\nabla\phi)_N & \text{for } F < 0. \end{cases} \tag{3.39}$$

The leading error term in the above equation is a function of $(\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla\phi)_P$, resembling a form of the diffusion term, and is therefore called numerical diffusion [115]. This discretisation practice is of first-order accuracy and it requires high spatial resolution to achieve accurate solutions [115].

- **Gamma differencing scheme** (Gamma) described in [69, 76] is a bounded scheme formed by blending CD with UD in the regions where CD would not

produce a bounded solution, thus:

$$\phi_f = \gamma(\phi)\,(\underline{\phi_f})_{CD} + (1-\gamma(\phi))\,(\underline{\phi_f})_{UD} + e_{Gamma}. \tag{3.40}$$

Here $\gamma(\phi)$ is a blending factor, $0 \leq \gamma(\phi) \leq 1$, dependent on the nature of the ϕ distribution around the face. The procedure used for evaluating $\gamma(\phi)$ is based on Normalised Variable Approach (NVA) of Leonard [88] and Gaskell *et al.* [50]. A normalised variable is defined as [88]:

$$\widetilde{\phi}_P = \frac{\phi_P - \phi_U}{\phi_D - \phi_U}, \tag{3.41}$$

where ϕ_P, ϕ_U and ϕ_D are the values in the node P, upwind node U and downstream node D, as depicted in Fig. 3.4. The solution is bounded if the

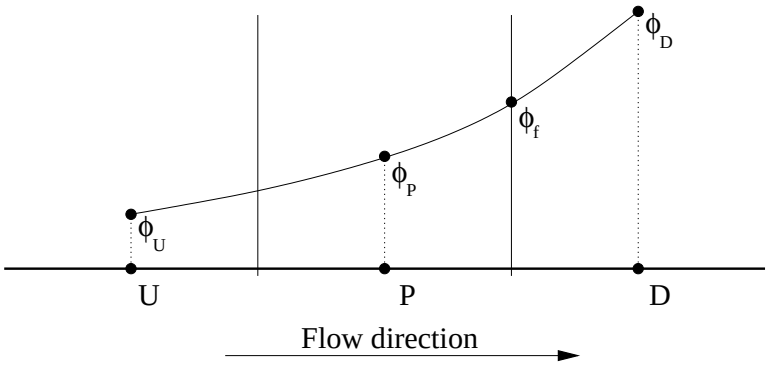


Figure 3.4: Variation of ϕ near the face

following conditions are satisfied:

$$\phi_U \leq \phi_P \leq \phi_D, \tag{3.42}$$

or

$$\phi_U \geq \phi_P \geq \phi_D, \tag{3.43}$$

from where it follows that $\widetilde{\phi}_P$ should obey:

$$0 \leq \widetilde{\phi}_P \leq 1. \tag{3.44}$$

Jasak [69] has modified Eqn. (3.41) to be applicable to arbitrary meshes, thus:

$$\widetilde{\phi}_P = 1 - \frac{(\nabla\phi)_f \cdot \mathbf{d}}{2(\nabla\phi)_P \cdot \mathbf{d}} \tag{3.45}$$

where the node P must be an upwind node to the face f and $\mathbf{d} = \mathbf{x}_D - \mathbf{x}_P$ connects the D and P nodes, Fig. 3.4. Depending on $\widetilde{\phi}_P$, the blending factor

$\gamma(\phi)$ is determined as follows [69]:

$$\begin{aligned}
\widetilde{\phi}_P > 1 &\Rightarrow \gamma(\phi) = 0,\\
\beta_m \leq \widetilde{\phi}_P \leq 1 &\Rightarrow \gamma(\phi) = 1,\\
0 \leq \widetilde{\phi}_P < \beta_m &\Rightarrow \gamma(\phi) = \frac{\widetilde{\phi}_P}{\beta_m},\\
\widetilde{\phi}_P < 0 &\Rightarrow \gamma(\phi) = 0,
\end{aligned}$$

where β_m is a constant of the scheme which was introduced to ensure linear transition between CD and UD when $0 \leq \widetilde{\phi}_P < \beta_m$ to improve convergence for steady-state problems. The range of β_m recommended by Jasak [69] is $0.1 \leq \beta_m \leq 0.5$.

The truncation error for this interpolation practice can be written as follows:

$$e_{Gamma} = \gamma(\phi) e_{CD} + (1 - \gamma(\phi)) e_{UD}, \tag{3.46}$$

where e_{CD} and e_{UD} are defined in Eqs. (3.36) and (3.39). The scheme is of second-order accuracy when $\gamma(\phi) = 1$ but it becomes a first-order scheme when $\gamma(\phi) < 1$.

The total error resulting from the discretisation of the convection term has contributions from the interpolation of mass flux and the interpolation of ϕ, so:

$$\begin{aligned}
\int_{V_P} \nabla \cdot (\rho \mathbf{U} \phi)\, dV &= \sum_f (F + e_{flux})(\underline{\phi_f} + e_{int})\\
&= \sum_f F \underline{\phi_f} + e_{conv},
\end{aligned} \tag{3.47}$$

where e_{conv} has the following form:

$$\begin{aligned}
e_{conv} &= \sum_f (F\, e_{int} + e_{flux}\, \underline{\phi_f} + e_{int}\, e_{flux})\\
&\approx C_{flux}\, e_{int} + C_{\phi_f}\, e_{flux} + e_{int}\, e_{flux}.
\end{aligned} \tag{3.48}$$

The order of the approximation is therefore equal to the lowest order approximation used in the process. If the procedures for interpolation of ϕ and F are of second-order then the approximation of the convection term is also of the second-order.

3.3.2 Diffusion Term

The discrete approximation of the diffusion term is obtained by using Eqn. (3.24) and taking the gradients on the faces to be constant due to the assumed linear

variation of the property ϕ. The result is:

$$\int_{V_P} \nabla \cdot (\rho\Gamma_\phi \nabla\phi)\, dV = \sum_f \mathbf{S} \cdot (\rho\Gamma_\phi \nabla\phi)_f$$
$$= \sum_f (\rho\Gamma_\phi)_f (\mathbf{S} \cdot \nabla\phi)_f, \tag{3.49}$$

where the terms $(\mathbf{S} \cdot \nabla\phi)_f$ and $(\rho\Gamma_\phi)_f$ need further treatment. The latter is interpolated onto the faces using Eqn. (3.14), where ϕ is substituted by $\rho\Gamma_\phi$. Approximation

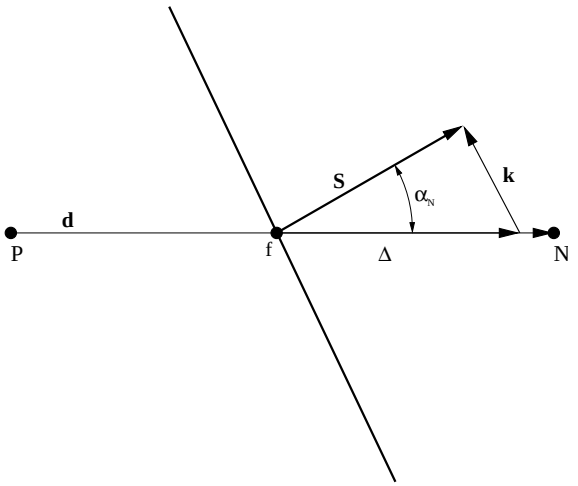


Figure 3.5: Non-orthogonality treatment

of $(\mathbf{S} \cdot \nabla\phi)_f$ on a non-orthogonal mesh, Fig. 3.5, when vectors $\mathbf{d}$ and $\mathbf{S}$ are not parallel, is performed using the following expression [69]:

$$\underline{(\mathbf{S} \cdot \nabla\phi)_f} = |\boldsymbol{\Delta}| \frac{\phi_N - \phi_P}{|\mathbf{d}|} + \mathbf{k} \cdot \underline{(\nabla\phi)_f}, \tag{3.50}$$

where $\underline{(\nabla\phi)_f}$ can be evaluated using Eqn. (3.16). Here, $\boldsymbol{\Delta}$ is parallel with $\mathbf{d}$ where $\boldsymbol{\Delta}$ and $\mathbf{k}$ have the property:

$$\mathbf{S} = \boldsymbol{\Delta} + \mathbf{k}. \tag{3.51}$$

In [69] Jasak has tested different treatments of $\boldsymbol{\Delta}$ and $\mathbf{k}$. The one for which $\mathbf{k}$ is orthogonal to $\mathbf{S}$, Fig. 3.5, performed best in terms of accuracy and convergence and is adopted here. The length of $\boldsymbol{\Delta}$ can be expressed as follows:

$$|\boldsymbol{\Delta}| = \frac{|\mathbf{S}|}{\cos\alpha_N}, \tag{3.52}$$

and the length of $\mathbf{k}$ can be calculated from:

$$|\mathbf{k}| = |\mathbf{S}| \tan\alpha_N. \tag{3.53}$$

The truncation error for the approximation of the $(\mathbf{S} \cdot \nabla\phi)_f$ term can be obtained using the following Taylor expansions:

$$\phi_P = \phi_f + (\mathbf{x}_P - \mathbf{x}_f) \cdot (\nabla\phi)_f + \frac{1}{2}(\mathbf{x}_P - \mathbf{x}_f)^2 : (\nabla\nabla\phi)_f + \frac{1}{6}(\mathbf{x}_P - \mathbf{x}_f)^3 :: (\nabla\nabla\nabla\phi)_f, \tag{3.54}$$

$$\phi_N = \phi_f + (\mathbf{x}_N - \mathbf{x}_f) \cdot (\nabla\phi)_f + \frac{1}{2}(\mathbf{x}_N - \mathbf{x}_f)^2 : (\nabla\nabla\phi)_f + \frac{1}{6}(\mathbf{x}_N - \mathbf{x}_f)^3 :: (\nabla\nabla\nabla\phi)_f. \tag{3.55}$$

By substituting Eqn. (3.54) and Eqn. (3.55) into Eqn. (3.50) and by adding the error from the interpolation of $\underline{(\nabla\phi)_f}$, the truncation error for $(\mathbf{S}\cdot\nabla\phi)_f$ is obtained, thus:

$$\begin{aligned}
e_{snGrad} &= (\mathbf{S} \cdot \nabla\phi)_f - \underline{(\mathbf{S} \cdot \nabla\phi)_f} \\
&= -\frac{1}{2}\frac{|\mathbf{\Delta}|}{|\mathbf{d}|}(|\mathbf{x}_N - \mathbf{x}_f|^2 - |\mathbf{x}_P - \mathbf{x}_f|^2)\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_f \\
&- \frac{1}{6}\frac{|\mathbf{\Delta}|}{|\mathbf{d}|}(|\mathbf{x}_N - \mathbf{x}_f|^3 + |\mathbf{x}_P - \mathbf{x}_f|^3)\hat{\mathbf{d}}^3 :: (\nabla\nabla\nabla\phi)_f \\
&- \frac{1}{2}|\mathbf{x}_N - \mathbf{x}_f||\mathbf{x}_P - \mathbf{x}_f|\mathbf{k} \cdot (\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_f) \\
&= -\frac{|\mathbf{S}|}{\cos\alpha_N}\frac{|\mathbf{d}|}{2}(2f_x - 1)\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_f \\
&- \frac{|\mathbf{S}|}{6\cos\alpha_N}|\mathbf{d}|^2\left((1-f_x)^3 + {f_x}^3\right)\hat{\mathbf{d}}^3 :: (\nabla\nabla\nabla\phi)_f \\
&- |\mathbf{S}|\tan\alpha_N\frac{|\mathbf{d}|^2}{2}f_x(1-f_x)\hat{\mathbf{k}} \cdot (\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_f),
\end{aligned} \tag{3.56}$$

where $\hat{\mathbf{d}}$ and $\hat{\mathbf{k}}$ are unit vectors in directions of $\mathbf{d}$ and $\mathbf{k}$, respectively, and f_x is the linear interpolation factor defined in Eqn. (3.17). From the dependence of Eqn. (3.56) on the f_x, it follows that the approximation is of first-order except for $f_x = 0.5$, *i.e.* present when the mesh is uniform. It is therefore advisable to keep the mesh as uniform as possible to obtain best accuracy. If the mesh is uniform, the approximation is of second-order. The error is also dependent on the angle of non-orthogonality and is minimal when $\alpha_N = 0$.

Finally, the discrete form of the diffusion term can be written:

$$\begin{aligned}
\int_{V_P} \nabla \cdot (\rho\Gamma_\phi\nabla\phi)\, dV &= \sum_f (\underline{(\rho\Gamma_\phi)_f} + e_{interpolation})(|\mathbf{\Delta}|\frac{\phi_N - \phi_P}{|\mathbf{d}|} + \mathbf{k} \cdot \underline{(\nabla\phi)_f} + e_{snGrad}) \\
&= \sum_f \underline{(\rho\Gamma_\phi)_f}(|\mathbf{\Delta}|\frac{\phi_N - \phi_P}{|\mathbf{d}|} + \mathbf{k} \cdot \underline{(\nabla\phi)_f}) + e_{diff}
\end{aligned} \tag{3.57}$$

where the truncation error for the diffusion term e_{diff} has the form:

$$\begin{aligned} e_{diff} &= \sum_f \underline{(\rho\Gamma_\phi)_f}\, e_{snGrad} \\ &+ \sum_f (|\mathbf{\Delta}| \frac{\phi_N - \phi_P}{|\mathbf{d}|} + \mathbf{k} \cdot \underline{(\nabla\phi)_f})\, e_{interpolation} \\ &+ \sum_f e_{interpolation}\, e_{snGrad} \\ &\approx \sum_f (C_{int}\, e_{snGrad} + C_{snGrad}\, e_{int} + e_{snGrad} e_{int}), \end{aligned} \tag{3.58}$$

from which it follows that the order of the approximation is equal to the lowest order found in Eqs. (3.56) and (3.23). Thus, the discretisation is of second-order on uniform meshes and reduces to first-order on the non-uniform ones. The behaviour of the truncation error and the achievable accuracy on CV's of different shapes will be compared in Section 3.6.

3.3.3 Source Terms

As it was previously mentioned, all the terms in the equations which cannot be expressed as convection, diffusion or temporal terms are grouped into the so-called source term. If the source term is dependent on ϕ, linearisation should be performed [115], such as:

$$S(\phi, \mathbf{x}) = Su(\mathbf{x}, \phi) + Sp(\mathbf{x}, \phi)\, \phi(\mathbf{x}). \tag{3.59}$$

When the Eqn. (3.59) is integrated over the control volume using Eqn. (3.11) the discretised form of the source term is obtained, thus:

$$\int_{V_P} S_\phi(\phi)\, dV = Su\, V_P + Sp\, V_P\, \phi_P + e_{source}. \tag{3.60}$$

The truncation error for the source term can be estimated by using the following Taylor expansion:

$$\begin{aligned} \int_{V_P} S(\mathbf{x}, \phi)\, dV = &\int_{V_P} \left(S(\mathbf{x}_P, \phi_P) + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla S(\mathbf{x}, \phi))_P + \Delta\phi \frac{\partial S(\mathbf{x}, \phi)}{\partial \phi} \right) dV \\ &+ \int_{V_P} \left(\frac{1}{2} (\mathbf{x} - \mathbf{x}_P)^2 :: (\nabla\nabla S(\mathbf{x}, \phi))_P \right) dV \\ &+ \int_{V_P} \left(\frac{1}{2} (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla S(\mathbf{x}, \phi))_P \Delta\phi \frac{\partial S}{\partial \phi} \right) dV \\ &+ \int_{V_P} \left(\frac{1}{2} (\Delta\phi)^2 \frac{\partial^2 S(\mathbf{x}, \phi)}{\partial \phi^2} \right) dV, \end{aligned} \tag{3.61}$$

where $\Delta\phi$ can be substituted by:

$$\Delta\phi = (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P. \tag{3.62}$$

After substituting $\Delta\phi$ and $S(\mathbf{x}_P, \phi_P) + \Delta\phi \frac{\partial S(\mathbf{x},\phi)}{\partial\phi}$ in Eqn. (3.61) using Eqn. (3.62) and Eqn. (3.59), respectively, the truncation error can be found as the difference between Eqn. (3.61) and Eqn. (3.60), thus:

$$\begin{aligned} e_{source} = &\frac{1}{2}(\mathbf{x} - \mathbf{x}_P)^2 : (\nabla\nabla S)_P \, V_P + \frac{1}{2}(\mathbf{x} - \mathbf{x}_P)^2 : ((\nabla\phi)_P \, (\nabla S)_P) \frac{\partial S}{\partial \phi} \, V_P \\ &+ \frac{1}{2}(\mathbf{x} - \mathbf{x}_P)^2 : ((\nabla\phi)_P)^2 \frac{\partial^2 S(\mathbf{x}, \phi)}{\partial \phi^2} \, V_P. \end{aligned} \tag{3.63}$$

The error is a function of $(\mathbf{x} - \mathbf{x}_P)^2$ but it also depends on the linearisation practice used.

3.4 Temporal Discretisation

Temporal discretisation is performed on a semi-discretised form of the transport equation, where the spatial terms have already been approximated using the practices described in the previous section. This form reads [62]:

$$\begin{aligned} \int_t^{t+\Delta t} \left[\left(\frac{\partial \rho\phi}{\partial t} \right)_P V_P + \sum_f F\phi_f - \sum_f (\rho\Gamma_\phi)_f \mathbf{S} \cdot (\nabla\phi)_f \right] dt \\ = \int_t^{t+\Delta t} (Su \, V_P + Sp \, V_P \, \phi_P) \, dt. \end{aligned} \tag{3.64}$$

and can be written in a shorter form [48]:

$$V_P \int_t^{t+\Delta t} \frac{\partial \rho\phi}{\partial t} \, dt = \int_t^{t+\Delta t} f(t, \phi(\mathbf{x}, t)) \, dt, \tag{3.65}$$

where $f(t, \phi(\mathbf{x}, t))$ contains all spatial terms from Eqn. (3.64). After performing integration of Eqn. (3.65) there results:

$$V_P \int_t^{t+\Delta t} \frac{\partial \rho\phi}{\partial t} \, dt = V_P(\phi^n - \phi^o) = \int_t^{t+\Delta t} f(t, \phi(\mathbf{x}, t)) \, dt, \tag{3.66}$$

where the subscripts o and n represent old and new time levels, respectively. The second important part of the temporal discretisation process is to choose an approximation for $\int_t^{t+\Delta t} f(t, \phi(\mathbf{x}, t)) \, dt$, which cannot be evaluated exactly. Taylor series expansion gives:

$$f(t + \Delta t, \phi(\mathbf{x}, t + \Delta t)) = f(t^o, \phi(\mathbf{x}, t^o)) + \Delta t \frac{df(t^o, \phi(\mathbf{x}, t^o))}{dt} + \frac{1}{2}(\Delta t)^2 \frac{d^2 f(t^o, \phi(\mathbf{x}, t^o))}{dt^2}, \tag{3.67}$$

which will be used later to establish the order of temporal discretisation. Some of the most common temporal discretisation practices are:

- **Euler Explicit method.** This method approximates Eqn. (3.66) by assuming [48]:

$$\int_t^{t+\Delta t} f(t, \phi(\mathbf{x}, t))\, dt = f(t, \phi(\mathbf{x}, t^o))\Delta t \tag{3.68}$$

i.e. it integrates the spatial terms by using the values at the beginning of the time interval. These spatial terms can be calculated because ϕ^o is known and the value at the end of the interval can be obtained directly for every node without having to solve a system of equations. It is shown in [48, 62] that this scheme is stable for Courant numbers $Co = \frac{|\mathbf{U}|\Delta t}{h} < 0.5$. Here, $\mathbf{U}$ is the transport velocity and h is the cell size.

The truncation error for this practice can be found as a difference between Eqn. (3.67) and Eqn. (3.68) which yields:

$$e_{expEuler} = \Delta t \frac{df(t^o, \phi(\mathbf{x}, t^o))}{dt} + \frac{1}{2}(\Delta t)^2 \frac{d^2 f(t^o, \phi(\mathbf{x}, t^o))}{dt^2} \tag{3.69}$$

This error is proportional to Δt, *i.e.* this a first-order method.

- **The Crank-Nicholson** discretisation practice assumes linear variation of $f(t, \phi(\mathbf{x}, t))$ in time [48, 62], giving:

$$\int_t^{t+\Delta t} f(t, \phi(\mathbf{x}, t))\, dt = \frac{1}{2}\left[f(t, \phi(\mathbf{x}, t^o)) + f(t, \phi(\mathbf{x}, t^n))\right]\Delta t \tag{3.70}$$

Eqn. (3.70) requires evaluation of spatial terms for old and new time steps. Because the values of ϕ^n are not known at the new time level this method requires a solution of a system of algebraic equations for each time step [62, 69].

The truncation error for this practice is:

$$e_{CN} = \frac{1}{2}(\Delta t)^2 \frac{d^2 f(t^o, \phi(\mathbf{x}, t^o))}{dt^2} \tag{3.71}$$

Thus, it is second-order time discretisation.

- **Euler Implicit method.** This method approximates Eqn. (3.66) by assuming [48]:

$$\int_t^{t+\Delta t} f(t, \phi(\mathbf{x}, t))\, dt = f(t, \phi(\mathbf{x}, t^n))\Delta t \tag{3.72}$$

Here the spatial terms in $f(t, \phi(\mathbf{x}, t))$ are evaluated using the ϕ^n which are obtained by solving a system of algebraic equations [62, 69]. This method is unconditionally stable for even for large time steps [48, 62].

The truncation error is:

$$e_{impEuler} = -\Delta t \frac{df(t^o, \phi(\mathbf{x}, t^o))}{dt} + \frac{1}{2}(\Delta t)^2 \frac{d^2 f(t^o, \phi(\mathbf{x}, t^o))}{dt^2} \tag{3.73}$$

Thus, as in the case of explicit method, the error is proportional to Δt and is therefore a first-order method.

3.5 Boundary Conditions

3.5.1 Boundary Conditions for the General Transport Equation

The solution of the general transport equation, Eqn. (2.32), is not complete without boundary and initial conditions. The latter are necessary for transient calculations in order to determine the initial state of the problem under consideration. Some terms in the equation (*eg.* convection and diffusion terms) require fluxes through CV faces which have to be specified at the boundary or evaluated from internal and boundary data. Most commonly, boundary conditions are given by prescribing the value of ϕ at the boundary (Dirichlet boundary condition) or its gradient (von Neumann boundary condition).

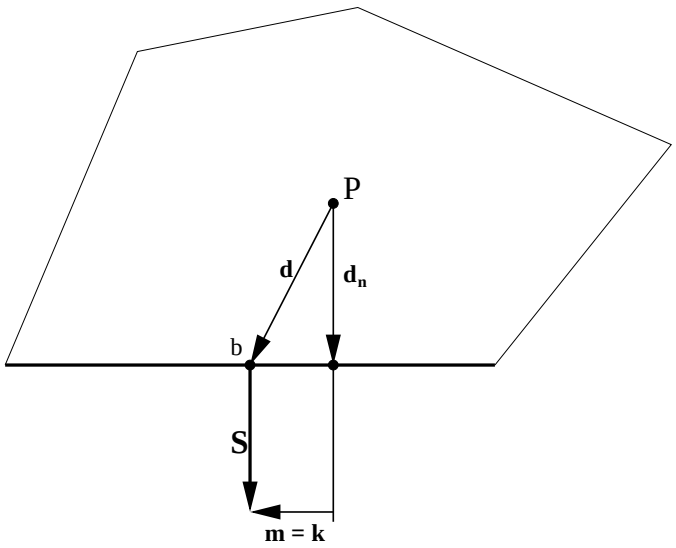


Figure 3.6: Boundary cell

Convective fluxes require a value of ϕ at a face, which is known exactly in the case of Dirichlet boundary condition, but in the case of von Neumann boundary

condition, it can be extrapolated from the neighbouring nodes onto the face by using:

$$\underline{\phi_b} = \phi_P + |\mathbf{d}_n| \, (\frac{\mathbf{S}}{|\mathbf{S}|} \cdot \nabla\phi)_b + \mathbf{m} \cdot (\nabla\phi)_b, \tag{3.74}$$

where $\mathbf{d}_n$ is a vector normal to the face, $\mathbf{m}$ is a skewness vector, Fig. 3.6, and $(\nabla\phi)_b$ is the prescribed boundary gradient. The truncation error for this extrapolation practice can be found using the following Taylor series expansion:

$$\phi_P = \phi_b - \mathbf{d} \cdot (\nabla\phi)_b + \frac{1}{2}\mathbf{d}^2 : (\nabla\nabla\phi)_b. \tag{3.75}$$

When ϕ_P in Eqn. (3.74) is substituted with Eqn. (3.75), the following truncation error is obtained:

$$e_{extrapolation} = \phi_b - \underline{\phi_b} = -\frac{|\mathbf{d}|^2}{2} \, \hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_b, \tag{3.76}$$

which shows that the approximation Eqn. (3.74) is second-order accurate.

Diffusive fluxes require the gradient of ϕ normal to the boundary face which is known exactly in the case of a von Neumann boundary condition. In the case of a Dirichlet boundary condition it can be evaluated as follows:

$$(\mathbf{S} \cdot \nabla\phi)_b = \frac{|\mathbf{S}|}{|\mathbf{d}_n|}(\phi_b - \phi_P - \mathbf{k} \cdot (\nabla\phi)_P), \tag{3.77}$$

where $\mathbf{k}$ is a non-orthogonality vector, Fig. 3.6. Note that the contribution of $\mathbf{k}$ to the expression Eqn. (3.77) is negative, and that is why it points in the same direction as the vector $\mathbf{m}$, see Fig. 3.6. The truncation error for $(\mathbf{S} \cdot \nabla\phi)_b$ approximated using Eqn. (3.77) can be derived by using the following Taylor series expansion:

$$\phi_b = \phi_P + \mathbf{d} \cdot (\nabla\phi)_P + \frac{1}{2}\mathbf{d}^2 : (\nabla\nabla\phi)_P. \tag{3.78}$$

Substituting ϕ_b in Eqn. (3.77) with Eqn. (3.78), after some algebra it gives:

$$e_{bouSnGrad} = |\mathbf{S}| \frac{|\mathbf{d}|}{\cos\alpha_N}(\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_P). \tag{3.79}$$

$\hat{\mathbf{d}}$ is a unit vector in the direction of $\mathbf{d}$. The error is a function of $|\mathbf{d}|$ and is therefore first-order accurate. This method corresponds to backward and forward differences in Finite Difference Method which are first-order methods [48]. The error is minimal when the mesh is orthogonal ($\alpha_N = 0$).

3.5.2 Boundary Conditions for the Navier-Stokes Equations

The most common boundary conditions which occur when solving the laminar and turbulent flows are:

- **Inlet.** The velocity distribution is prescribed and the surface-normal gradient is set to zero for the pressure. Values of turbulent properties (*eg.* k, ϵ, q and ζ) are prescribed.

- **Outlet.** The pressure distribution is prescribed there. Surface-normal gradients of the velocity field and turbulent properties are set to zero.

- **Symmetry plane.** The surface-normal gradients of all scalar fields are zero at the symmetry plane. The convection flux through the symmetry plane is zero. Viscous stress of the velocity component parallel to the boundary is zero and it must be imposed directly [48] while the viscous stress of the velocity component normal to the wall is evaluated using Eqn. (3.77).

- **Impermeable no-slip walls.** The velocity field is fixed to the velocity of the wall and the surface-normal gradient of the pressure is set to zero there. The convection flux through the wall is zero. Viscous stress of the velocity component normal to the wall is zero and it must be imposed as an additional condition [48]. Viscous stress of the velocity component parallel to the wall is not zero and its evaluation is dependent on the near-wall modelling employed [48]. Conditions imposed on the turbulence properties depend on the turbulence model and the type of near-wall treatment. The values of q and ζ fields, defined in Chapter 2, are fixed at the wall. Zero surface-normal gradient is prescribed for k and ϵ fields in order to ensure zero contribution from the boundary face into the equation for the near-wall cell. ϵ in the near-wall cell is set to:

$$\epsilon_P = C_\nu^{0.75} \frac{k_P^{\frac{3}{2}}}{\kappa |\mathbf{d}_n|}, \tag{3.80}$$

  where the subscript P denotes the near-wall cell and $|\mathbf{d}_n|$ is the distance from the wall, shown in Fig. 3.6. Another term which requires special treatment near the wall is the production term P for the k equation, defined in Eqn. (2.19). Its value in the near-wall cell is set to:

$$P_P = \nu_t C_\nu^{0.25} \frac{\overline{U_t}}{|\mathbf{d}_n|} \frac{\sqrt{k_P}}{\kappa |\mathbf{d}_n|}, \tag{3.81}$$

where $\overline{U_t}$ is the velocity parallel to the wall. The values of the k field in the near-wall cells are calculated from its differential equation using the values for ϵ_P and P_P defined above.

3.6 Discretisation Errors on different types of meshes

Discretisation errors are dependent both on the type of computational mesh (*i.e.* tetrahedral, hexahedral, polyhedral) and the approximations for different terms in the governing equations. This section will present a comparison of accuracy achievable on different mesh types. This is done by using the expressions for the truncation errors presented in the preceeding sections of this chapter. The comparison will be presented for convection and diffusion terms. Discretisation procedures for divergence and gradient terms are very similar to the discretisation procedure for the convection term and everything that will be said about error for the convection term can be applied to them.

Four types of meshes will be considered for the analysis:

- **Square mesh.** A representative cell with its neighbours which influence it are shown in Fig. 3.7.

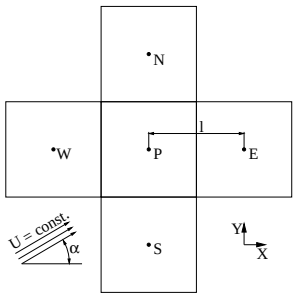


Figure 3.7: Square mesh

This mesh is orthogonal, it is not skewed and $f_x = 0.5$ because the centre of each internal face lies midway between the neighbouring nodes.

- **Triangular mesh.** A mesh consisting of equilateral triangles is uniform, orthogonal and not skewed, see Fig. 3.8.

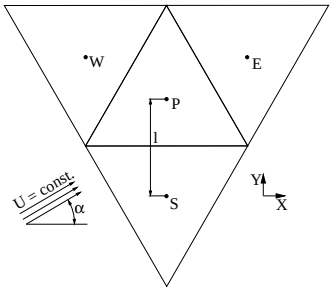


Figure 3.8: Triangular mesh

- **Regular hexagonal mesh.** This type of mesh can be generated using a Delaunay algorithm, see Chapter 7. It consists of hexagons and is uniform, orthogonal and not skewed, Fig. 3.9.

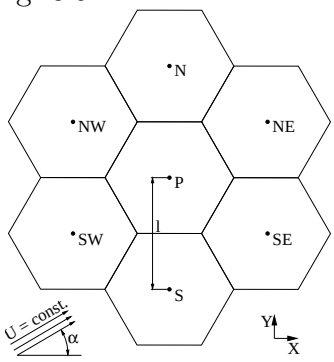


Figure 3.9: Hexagonal mesh

- **Split-hexahedron mesh.** This type of mesh is produced when local refinement of hexahedral cells is done by splitting cells, such that a cell face gets divided into two or four faces. An example is shown in Fig. 3.10.

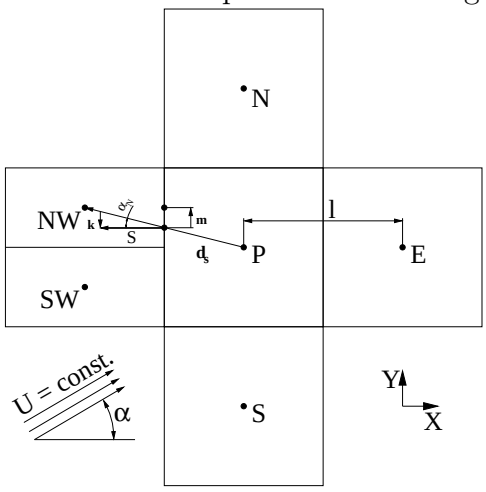


Figure 3.10: Split-hexahedron mesh

Unlike the previous examples, a split-hexahedron mesh is skewed and not orthogonal but $f_x = 0.5$. The distances between nodes at the split face are also larger than on other faces. Thus, for the example above:

$$\mathbf{d}_s = l_s \hat{\mathbf{d}} = \frac{l}{\cos \alpha_N} \hat{\mathbf{d}}, \tag{3.82}$$

where α_N can be calculated from:

$$\tan \alpha_N = \frac{0.25\, l}{l} = \frac{1}{4}. \tag{3.83}$$

3.6.1 Convection Term

The analysis of errors in the convection term will be performed by using the Central Differencing scheme and taking the fluid velocity and density to be constant:

$$\mathbf{U} = const.$$

$$\rho = const.$$

in order not to include errors coming from evaluation of mass fluxes, Eqn. (3.33). The truncation error for the CD scheme was given in Section 3.3.1 and it is:

$$e_{conv} = -\frac{1}{2} \sum_f F |\mathbf{d}|^2 \left(f_x (1 - f_x) (\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_{f_i}) + \psi |\mathbf{d}|\, \hat{\mathbf{m}} \cdot (\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_{f_i}) \right)$$
$$+ \sum_f F \frac{\psi^2}{2} |\mathbf{d}|^2 (\hat{\mathbf{m}}^2 : (\nabla\nabla\phi)_{f_i}). \tag{3.84}$$

Here, f_x is a linear interpolation factor defined in Eqn. (3.17). The second and the third term in Eqn. (3.84) are zero on meshes which are not skewed.

- **Error on square mesh.** The distance between the two nodes is equal to the length of a side. Hence, the mass flux is:

$$F = \rho\, |\mathbf{U}|\, l \tag{3.85}$$

because $|\mathbf{S}| = l$; and $f_x = 0.5$. Replacing $|\mathbf{d}|$ in Eqn. (3.84) with l yields:

$$\begin{aligned} e_{convSquare} = & -\frac{1}{8} F \cos\alpha\, l^2 (\mathbf{n}_e^2 : (\nabla\nabla\phi)_e) \\ & - \frac{1}{8} F \sin\alpha\, l^2 (\mathbf{n}_n^2 : (\nabla\nabla\phi)_n) \\ & + \frac{1}{8} F \cos\alpha\, l^2 (\mathbf{n}_w^2 : (\nabla\nabla\phi)_w) \\ & + \frac{1}{8} F \sin\alpha\, l^2 (\mathbf{n}_s^2 : (\nabla\nabla\phi)_s) \\ = & -\frac{1}{8} l^2\, F \cos\alpha\, (\mathbf{n}_e^2 : ((\nabla\nabla\phi)_e - (\nabla\nabla\phi)_w)) \\ & - \frac{1}{8} l^2\, F \sin\alpha\, (\mathbf{n}_n^2 : ((\nabla\nabla\phi)_n - (\nabla\nabla\phi)_s)), \end{aligned} \tag{3.86}$$

where $\hat{\mathbf{d}}^2$ from Eqn. (3.84) is replaced by $\mathbf{n}$ which represents a unit-face-normal vector pointing outwards from the cell P. The subscript $\mathbf{n}_e$ denotes that the face is shared with a neighbour E. α is the angle between the velocity vector and the x axis.

- **Equilateral Triangular Mesh.** For this the mass flux is:

$$F = \sqrt{3}\,\rho\,|\mathbf{U}|\,l. \tag{3.87}$$

The truncation error can be found using Eqn. (3.84), yielding:

$$\begin{aligned} e_{convTri} = &-\frac{\sqrt{3}}{16} l^2 F \cos\alpha\,((\mathbf{n}_e^2 : (\nabla\nabla\phi)_e) - (\mathbf{n}_w^2 : (\nabla\nabla\phi)_w)) \\ &- \frac{1}{8} l^2 F \sin\alpha(\frac{1}{2}(\mathbf{n}_e^2 : (\nabla\nabla\phi)_e) - (\mathbf{n}_s^2 : (\nabla\nabla\phi)_s) - \frac{1}{2}(\mathbf{n}_w^2 : (\nabla\nabla\phi)_w)). \end{aligned} \tag{3.88}$$

- **Hexagonal Mesh.** Here the expression for the mass flux is:

$$F = \frac{\sqrt{3}}{3}\,\rho\,|\mathbf{U}|\,l, \tag{3.89}$$

and Eqn. (3.84) gives the truncation error as:

$$\begin{aligned} e_{convHexagon} = &-\frac{\sqrt{3}}{16} l^2 F \cos\alpha\ \left[\mathbf{n}_{ne}^2 : ((\nabla\nabla\phi)_{ne} - (\nabla\nabla\phi)_{sw})\right] \\ &- \frac{\sqrt{3}}{16} l^2 F \cos\alpha\ \left[\mathbf{n}_{se}^2 : ((\nabla\nabla\phi)_{se} - (\nabla\nabla\phi)_{nw})\right] \\ &- \frac{1}{8} l^2 F \sin\alpha\ \left[\mathbf{n}_{n}^2 : ((\nabla\nabla\phi)_{s} - (\nabla\nabla\phi)_{s})\right] \\ &- \frac{1}{16} l^2 F \sin\alpha\ \left[\mathbf{n}_{ne}^2 : ((\nabla\nabla\phi)_{ne} - (\nabla\nabla\phi)_{sw})\right] \\ &- \frac{1}{16} l^2 F \sin\alpha\ \left[\mathbf{n}_{se}^2 : ((\nabla\nabla\phi)_{se} - (\nabla\nabla\phi)_{nw})\right]. \end{aligned} \tag{3.90}$$

- **Split-hexahedron Mesh.** The truncation error for the configuration shown in Fig. 3.10 is:

$$\begin{aligned} e_{convSplitHex} = &-\frac{1}{8} l^2 F \sin\alpha\ \left[\mathbf{n}_{n,s}^2 : ((\nabla\nabla\phi)_{n} - (\nabla\nabla\phi)_{s})\right] \\ &- \frac{1}{8} F \cos\alpha \left[l^2\,\mathbf{n}_e^2 : (\nabla\nabla\phi)_e - \frac{1}{2} l_s^2\,\hat{\mathbf{d}}_{nw}^2 : (\nabla\nabla\phi)_{nw}\right] \\ &- \frac{1}{8} F \cos\alpha \left[\frac{1}{2} l_s^2\,\hat{\mathbf{d}}_{sw}^2 : (\nabla\nabla\phi)_{sw}\right] \\ &+ \frac{1}{2} F \cos\alpha\,\psi\,l_s^3 \left[\mathbf{m}_{nw} \cdot (\hat{\mathbf{d}}_{nw}^2 :: (\nabla\nabla\nabla\phi)_{nw})\right] \\ &+ \frac{1}{2} F \cos\alpha\,\psi\,l_s^3 \left[\mathbf{m}_{sw} \cdot (\hat{\mathbf{d}}_{sw}^2 :: (\nabla\nabla\nabla\phi)_{sw})\right] \\ &- F \cos\alpha\,\frac{\psi^2}{2}\,l_s^2\,\left[\hat{\mathbf{m}}_{nw}^2 : ((\nabla\nabla\phi)_{nw} + (\nabla\nabla\phi)_{sw})\right] \end{aligned} \tag{3.91}$$

where F is defined in Eqn. (3.85), l_s can be calculated from Eqn. (3.82) and ψ can be obtained from Eqn. (3.8), thus:

$$\psi = \frac{|\mathbf{m}|}{l_s} = \frac{l}{4\, l_s}. \tag{3.92}$$

Evidently, the error in Eqs. (3.86), (3.88), (3.90) and (3.91) is dependent both on the spatial resolution and the form of the solution itself, as expressed through its gradients. A comparison of accuracy can therefore only be performed by examining different forms of solutions, as follows:

1. $\nabla\phi = constant$. When the solution field has a uniform and fixed gradient the error is zero on all types of meshes. This is consistent with the assumption expressed in Eqn. (3.3).

2. $\nabla\nabla\phi = constant$. This class of solutions with uniform curvature reveals differences in accuracy between different mesh types. The discretisation on square and hexagonal meshes still produces exact solutions Eqs. (3.86) and (3.90), while triangular meshes and meshes with split-hexahedra produce errors. The truncation error for triangular meshes, Eqn. (3.88), can be simplified to:

$$\begin{aligned} e_{convTri} = &-\frac{\sqrt{3}}{16} l^2 F \cos\alpha \,((\mathbf{n}_e^2 - \mathbf{n}_w^2) : \nabla\nabla\phi) \\ &- \frac{1}{8} l^2 F \sin\alpha ((\frac{1}{2}\mathbf{n}_e^2 - \mathbf{n}_s^2 - \frac{1}{2}\mathbf{n}_w^2) : \nabla\nabla\phi) \end{aligned} \tag{3.93}$$

while the truncation error for a split-hexahedron mesh, Eqn. (3.91), can be reduced to:

$$\begin{aligned} e_{convSplitHex} = &-\frac{1}{8} F \cos\alpha \left[(l^2\, \mathbf{n}_e^2 - \frac{1}{2} l_s^2\, \hat{\mathbf{d}}_{nw}^2 - \frac{1}{2} l_s^2\, \hat{\mathbf{d}}_{sw}^2) : \nabla\nabla\phi \right] \\ &- F \cos\alpha\, \frac{\psi^2}{2}\, l_s^2 \left[(\hat{\mathbf{m}}_{nw}^2 + \hat{\mathbf{m}}_{sw}^2) : \nabla\nabla\phi \right]. \end{aligned} \tag{3.94}$$

The above equations show that the errors exist because these mesh types have not got face pairs, such that the error on each pair cancel. Two cell faces form a face pair if the sum of their outward-pointing normal vectors is a zero vector. From there it follows that the difference between the second tensors of the face normals, *i.e.* $(\mathbf{n}_e^2 - \mathbf{n}_w^2) = \mathbf{0}$, is a zero tensor for every face pair; which results in zero discretisation error. Triangular meshes have not got any face pairs because $(\mathbf{n}_e^2 - \mathbf{n}_w^2)$ and $(\frac{1}{2}\mathbf{n}_e^2 - \mathbf{n}_s^2 - \frac{1}{2}\mathbf{n}_w^2)$ in Eqn. (3.93) are not zero tensors.

Skewness contributes to the error in Eqn. (3.94) because $(\hat{\mathbf{m}}_{nw}^2 + \hat{\mathbf{m}}_{sw}^2)$ is not a zero tensor.

3. $\nabla\nabla\nabla\phi = constant$. This type of problem cannot be resolved exactly on any type of mesh. Hence, it should show which of the square or hexagon shapes should be more accurate. The variation of $\nabla\nabla\phi(\mathbf{x})$ within the cell is:

$$\nabla\nabla\phi(\mathbf{x}) = (\nabla\nabla\phi)_P + (\mathbf{x} - \mathbf{x}_P) \cdot \nabla\nabla\nabla\phi \tag{3.95}$$

When $\nabla\nabla\phi$ in Eqn. (3.84) is substituted with Eqn. (3.95), the following expressions for the truncation errors result.

$$e_{convSquare} = -\frac{1}{8} l^3 F \cos\alpha \, \mathbf{n}_e^3 :: \nabla\nabla\nabla\phi - \frac{1}{8} l^3 F \sin\alpha \, \mathbf{n}_n^3 :: \nabla\nabla\nabla\phi. \tag{3.96}$$

$$\begin{aligned} e_{convHexagon} = & -\frac{1}{8} l^3 F \cos\alpha \, \frac{\sqrt{3}}{2} (\mathbf{n}_{ne}^3 + \mathbf{n}_{nw}^3) :: \nabla\nabla\nabla\phi \\ & - -\frac{1}{8} l^3 F \sin\alpha \, (\mathbf{n}_n^3 + \frac{1}{2}\mathbf{n}_{ne}^3 - \frac{1}{2}\mathbf{n}_{nw}^3) :: \nabla\nabla\nabla\phi. \end{aligned} \tag{3.97}$$

In order to determine which type of mesh is the more accurate, the problem is simplified such that $\frac{\partial^3\phi}{\partial x^3} = constant$ while all other components of $\nabla\nabla\nabla\phi$ are zero. The velocity $\mathbf{U}$ is at angle $\alpha = 0$. When the above is inserted in Eqs. (3.96) and (3.97) there results:

$$e_{convSquare} = -\frac{1}{8} l^3 F \frac{\partial^3\phi}{\partial x^3}, \tag{3.98}$$

$$e_{convHexagon} = -\frac{9}{64} l^3 F \frac{\partial^3\phi}{\partial x^3}. \tag{3.99}$$

For the same number of cells in the domain the area of the square cell is equal to the area of the hexagon. From there it follows that:

$$l_{hexagon} = \frac{2\sqrt{3}}{3} l_{square}. \tag{3.100}$$

By knowing the ratio between $l_{hexagon}$ and l_{square}, Eqn. (3.100) the ratio between the errors is:

$$\frac{e_{convHexagon}}{e_{convSquare}} = 1.155. \tag{3.101}$$

Eqn. (3.101) shows that the error on the hexagonal mesh is expected to be higher than for the square mesh. The reason for this lies in a fact that hexagons always have more faces with non-zero mass fluxes than the square such that the sum of errors committed on those faces is larger than for the square.

3.6.2 Diffusion Term

The accuracy of the diffusion term discretisation on different types of meshes can be performed by comparing the truncation errors defined by Eqn. (3.58). In order to simplify the analysis, the fluid density and the diffusion coefficient Γ are assumed to be constant, *i.e.* :

$$\rho = constant.$$

$$\Gamma = constant.$$

This makes the error arising from interpolation of $\rho\Gamma$ onto the cell faces equal to zero; and the only remaining error comes from the approximation of surface-normal gradients, Eqn. (3.50):

$$\begin{aligned} e_{diff} = &-\sum_f (\rho\Gamma)\frac{|\mathbf{S}|}{\cos\alpha_N}\frac{|\mathbf{d}|}{2}(2f_x - 1)\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_f \\ &-\sum_f (\rho\Gamma)\frac{|\mathbf{S}|}{6\,\cos\alpha_N}|\mathbf{d}|^2\left((1-f_x)^3 + {f_x}^3\right)\hat{\mathbf{d}}^3 :: (\nabla\nabla\nabla\phi)_f \\ &-\sum_f (\rho\Gamma)|\mathbf{S}|\tan\alpha_N\frac{|\mathbf{d}|^2}{2}f_x(1-f_x)\hat{\mathbf{k}}\cdot(\hat{\mathbf{d}}^2 : (\nabla\nabla\nabla\phi)_f), \end{aligned} \tag{3.102}$$

This will now be evaluated for the different mesh types.

- **Square Mesh.** For the configuration depicted in, Fig. 3.7, the magnitude of the surface vector is:

$$|\mathbf{S}| = l. \tag{3.103}$$

Noting that $|\mathbf{d}| = l$, $\alpha_N = 0$ and $f_x = 0.5$, the truncation error can be written:

$$\begin{aligned} e_{diffHex} = &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2(\mathbf{n}_e^3 :: (\nabla\nabla\nabla\phi)_e) \\ &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2(\mathbf{n}_w^3 :: (\nabla\nabla\nabla\phi)_w) \\ &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2(\mathbf{n}_n^3 :: (\nabla\nabla\nabla\phi)_n) \\ &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2(\mathbf{n}_s^3 :: (\nabla\nabla\nabla\phi)_s) \\ = &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2\left[\mathbf{n}_e^3 :: ((\nabla\nabla\nabla\phi)_e - (\nabla\nabla\nabla\phi)_w)\right] \\ &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2\left[\mathbf{n}_n^3 :: ((\nabla\nabla\nabla\phi)_n - (\nabla\nabla\nabla\phi)_s)\right] \end{aligned} \tag{3.104}$$

- **Equilateral Triangular Mesh.** The truncation error for this cell type is:

$$e_{diffTri} = -\frac{1}{24}\rho\Gamma|\mathbf{S}|l^2\left[\mathbf{n}_e^3 :: (\nabla\nabla\nabla\phi)_e + \mathbf{n}_w^3 :: (\nabla\nabla\nabla\phi)_w + \mathbf{n}_s^3 :: (\nabla\nabla\nabla\phi)_s\right], \tag{3.105}$$

where $|\mathbf{S}| = \sqrt{3}l$.

- **Equilateral Hexagonal Mesh.**

$$\begin{aligned} e_{diffHexagon} = &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2\left[\mathbf{n}_n^3 :: ((\nabla\nabla\nabla\phi)_n - (\nabla\nabla\nabla\phi)_s)\right] \\ &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2\left[\mathbf{n}_{ne}^3 :: ((\nabla\nabla\nabla\phi)_{ne} - (\nabla\nabla\nabla\phi)_{sw})\right] \\ &-\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^2\left[\mathbf{n}_{nw}^3 :: ((\nabla\nabla\nabla\phi)_{nw} - (\nabla\nabla\nabla\phi)_{se})\right] \end{aligned} \tag{3.106}$$

where $|\mathbf{S}| = \frac{\sqrt{3}}{3}l$.

- **Split-Hexahedron Mesh.**

$$\begin{aligned} e_{diffSplitHex} = &-\frac{1}{24}\rho\Gamma|\mathbf{S}|l^2\left[\mathbf{n}_n^3 :: ((\nabla\nabla\nabla\phi)_n - (\nabla\nabla\nabla\phi)_s)\right] \\ &-\frac{1}{24}\rho\Gamma|\mathbf{S}|l^2\mathbf{n}_e^3 :: (\nabla\nabla\nabla\phi)_e \\ &-\frac{1}{48}\rho\Gamma|\mathbf{S}|\frac{l^2}{\cos^3\alpha_N}\left[\hat{\mathbf{d}}_{nw}^3 :: (\nabla\nabla\nabla\phi)_{nw} + \hat{\mathbf{d}}_{sw}^3 :: (\nabla\nabla\nabla\phi)_{sw})\right] \\ &-\frac{1}{16}\rho\Gamma|\mathbf{S}|\frac{l^2\,\sin\alpha_N}{\cos^3\alpha_N}\left[\hat{\mathbf{k}}_{nw}\cdot(\hat{\mathbf{d}}_{nw}^2 : (\nabla\nabla\nabla\phi)_{nw})\right] \\ &-\frac{1}{16}\rho\Gamma|\mathbf{S}|\frac{l^2\,\sin\alpha_N}{\cos^3\alpha_N}\left[\hat{\mathbf{k}}_{sw}\cdot(\hat{\mathbf{d}}_{sw}^2 : (\nabla\nabla\nabla\phi)_{sw})\right]. \end{aligned} \tag{3.107}$$

Again, the comparison will be made for different forms of solution fields:

1. $\nabla\nabla\phi$ = *constant*. Eqs. (3.104), (3.105), (3.106) and (3.107) show that the error is zero for all mesh types.

2. $\nabla\nabla\nabla\phi$ = *constant*. For such fields, exact solutions are produced on meshes consisting of squares Eqn. (3.104) and hexagons Eqn. (3.106). The truncation errors for the triangular and split-hexahedral meshes are, respectively:

$$e_{diffTri} = -\frac{1}{24}\rho\Gamma|\mathbf{S}|l^2\left[(\mathbf{n}_e^3 + \mathbf{n}_w^3 + \mathbf{n}_s^3) :: (\nabla\nabla\nabla\phi)\right], \tag{3.108}$$

$$\begin{aligned} e_{diffSplitHex} = &-\frac{1}{24}\rho\Gamma|\mathbf{S}|l^2\left[(\mathbf{n}_e^3 + \frac{\hat{\mathbf{d}}_{nw}^3 + \hat{\mathbf{d}}_{sw}^3}{2\cos^3\alpha_N}) :: (\nabla\nabla\nabla\phi)\right] \\ &-\frac{1}{16}\rho\Gamma|\mathbf{S}|\frac{l^2\,\sin\alpha_N}{\cos^3\alpha_N}\left[\hat{\mathbf{k}}_{nw}\cdot(\hat{\mathbf{d}}_{nw}^2 - \hat{\mathbf{d}}_{sw}^2) : (\nabla\nabla\nabla\phi)\right]. \end{aligned} \tag{3.109}$$

The above errors are a consequence of the fact that these types of cells have an odd number of faces, some of which are not paired; thus the error committed on them cannot cancel out. For example, third rank tensor $(\mathbf{n}_e^3 + \mathbf{n}_w^3 + \mathbf{n}_s^3)$ in

Eqn. (3.108) is not a zero tensor. On the contrary, cell face pairs make the sum of those tensors equal to zero, *i.e.* $(\mathbf{n}_n^3 + \mathbf{n}_s^3) = \mathbf{0}$ in Eqs. (3.104) and (3.106). Non-orthogonality increases the error because $(\hat{\mathbf{d}}_{nw}^2 - \hat{\mathbf{d}}_{sw}^2)$ in Eqn. (3.109) is not a zero tensor because $\hat{\mathbf{d}}_{nw}$ and $\hat{\mathbf{d}}_{sw}$ are not parallel.

3. $\nabla\nabla\nabla\nabla\phi = constant$. Exact solutions for this type of problem cannot be obtained on any mesh type. The variation of $\nabla\nabla\nabla\phi(\mathbf{x})$ within the domain is:

$$\nabla\nabla\nabla\phi(\mathbf{x}) = (\nabla\nabla\nabla\phi)_P + (\mathbf{x} - \mathbf{x}_P) \cdot \nabla\nabla\nabla\nabla\phi. \tag{3.110}$$

The error for the square cell reads:

$$e_{diffSquare} = -\frac{1}{24}\rho\Gamma|\mathbf{S}|l^3(\mathbf{n}_e^4 + \mathbf{n}_n^4) ::: \nabla\nabla\nabla\nabla\phi, \tag{3.111}$$

and the error for hexagon can be written:

$$e_{diffHexagon} = -\frac{1}{24}\rho\Gamma|\mathbf{S}|l^3(\mathbf{n}_{ne}^4 + \mathbf{n}_n^4 + \mathbf{n}_{nw}^4) ::: \nabla\nabla\nabla\nabla\phi. \tag{3.112}$$

In order to get an insight into the error behaviour on such problems, the problem is simplified to $\frac{\partial^4\phi}{\partial x^4} = constant$ while all other components of $\nabla\nabla\nabla\nabla\phi$ are zero. By inserting this into Eqs. (3.111) and (3.112), there results:

$$e_{diffSquare} = -\frac{1}{24}(\rho\Gamma)|\mathbf{S}|l^3\frac{\partial^4\phi}{\partial x^4}, \tag{3.113}$$

$$e_{diffHexagon} = -\frac{1}{24}(\rho\Gamma)l^3\frac{9}{8}\frac{\partial^4\phi}{\partial x^4}. \tag{3.114}$$

By comparing Eqs. (3.113) and (3.114) on meshes with the same number of cells there results:

$$\frac{e_{diffHexagon}}{e_{diffSquare}} = 1.155. \tag{3.115}$$

This is the same result as the one obtained for the convection term.

The results of the analysis lead to conclusion that cell shapes with paired faces should be used whenever possible. Such cells have the number of symmetry planes equal to the number of face pairs which cancel the error. Taking all of the above into account suggests that square is the best possible shape from the aspect of accuracy followed by a hexagon, a triangle and a split-hexahedron.

3.7 Solution of Linear Equation Systems

Discretisation produces systems of algebraic equations equal in number to the product of the numbers of cells and dependent variables. The equations may be linear or

non-linear, according to the nature of the parent differential equations. Non-linear equations can be linearised in some manner and then solved as for linear problems, but in an iterative or time-marching fashion. Solution of the algebraic systems is the most time consuming part of the CFD analysis [49, 80].

The system of linear algebraic equations can be written [48]:

$$a_P\phi_P + \sum_N a_N\phi_N = Q_P. \tag{3.116}$$

There are two main classes of solution algorithms.

Direct methods provide a solution of the system in a fixed number of operations. The most popular methods of this type are: Thomas algorithm (TDMA), Gauss Elimination, LU decomposition [48]. They are appropriate for small systems as the number of required operations increases rapidly with the number of equations, except in case of TDMA for which it increases linearly. Memory requirement for direct methods also increases with the number of equations squared making this approach even less attractive for large systems.

Iterative methods start from an initial solution and improve the solution until the error tolerance imposed by the user is met. In CFD applications, discretisation and modelling errors are generally larger than the machine accuracy and it is not necessary to solve the system exactly, making iterative methods very attractive. However, these solvers impose some conditions on the structure and characteristics of the matrix. Matrices resulting from the FV discretisation process are sparse, with most coefficients equal to zero. The memory requirement for such matrices can be reduced if the solver can preserve the sparseness of the matrix. Some iterative methods preserve sparseness, making their use even more attractive.

Some iterative solvers require the matrix to be diagonally dominant in order to guarantee convergence. Diagonal dominance occurs if the diagonal coefficient is equal to the sum of magnitudes of the off-diagonal coefficients for every equation, with at least one diagonal coefficient greater than the sum of magnitudes of non-diagonal terms [134]. Diagonal dominance can be increased by under-relaxation. The dominant diagonal coefficient can be calculated as follows [70]:

$$\overline{a_P} = max(a_P, \sum_N |a_N|). \tag{3.117}$$

After some algebra the under-relaxed system can be written as:

$$\frac{\overline{a_P}}{\alpha}\phi_P^n + \sum_N a_N\phi_N^n = Q_P + (\frac{\overline{a_P}}{\alpha} - a_P)\phi_P^o, \tag{3.118}$$

where $0 < \alpha < 1$ is the under-relaxation factor, ϕ_P^o represents the value from the previous iteration and ϕ_P^n is the new value of the field. When the system has converged to its solution, the additional terms cancel out as ($\phi_P^n = \phi_P^o$). Under-relaxation is not used for transient calculations because the transient term enhances the diagonal dominance of the matrix and has the same effect as under-relaxation.

In this study the Conjugate Gradient (CG) based methods are used to solve the system of algebraic equations. This method was proposed by Hestens and Steifel [60]. For symmetric matrices the Incomplete Cholesky preconditioned Conjugate Gradient (ICCG) is used, described in detail in [68]. Bi-CGSTAB by van der Vorst [148] is the solver used for asymmetric matrices.

3.8 Solution Algorithm for the Navier-Stokes System

The discretisation of all terms in the general transport equation, Eqn. (2.32), have been described in the preceeding sections. The same principles apply to the terms of the Navier-Stokes equations with some additional problems which require further treatment.

The system of N-S equations is dependent on the pressure through the pressure-gradient term in the momentum equation but the pressure lacks its own equation. If the flow is compressible the continuity equation can be used to obtain the density field, which can be used to calculate the pressure from an equation of state. In contrast, for incompressible flows the continuity equation becomes an additional constraint on the velocity field. A way to overcome this difficulty is to construct the pressure field such that velocity satisfies the continuity equation. This is done by modifying the continuity equation into an equation for pressure [48].

3.8.1 Pressure Equation

The derivation of the pressure equation for the collocated variable arrangement [48, 120] will be presented in this section.

The pressure equation will be derived by using the incompressible isothermal

N-S equations as the model, which read:

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathbf{U}\mathbf{U} - \nabla \cdot \nu \nabla \mathbf{U} = -\nabla p, \tag{3.119}$$

$$\nabla \cdot \mathbf{U} = 0, \tag{3.120}$$

where ν and p are the kinematic viscosity and the kinematic pressure, respectively. Temporal discretisation is performed using some implicit temporal scheme, such as:

$$\int_t^{t+\Delta t} f(t, \mathbf{U}(\mathbf{x}, t))\, dt = (1-C)\Delta t f(t, \mathbf{U}(\mathbf{x}, t^o)) + C\Delta t f(t, \mathbf{U}(\mathbf{x}, t^n)), \tag{3.121}$$

where different values of C can recover temporal schemes defined in Section 3.4. When the momentum equation is approximated by using Eqn. (3.121) and the discretisation procedures for spatial terms described in the preceeding sections are applied, there results:

$$(\mathbf{U}_P^n - \mathbf{U}_P^o)V_P + C\Delta t \left[\sum_f F^n U_f^n - \sum_f \nu_f (\mathbf{S} \cdot \nabla \mathbf{U})_f^n + \nabla p^n V_P \right]$$
$$+ (1-C)\Delta t \left[\sum_f F^o U_f^o - \sum_f \nu_f (\mathbf{S} \cdot \nabla \mathbf{U})_f^o + \nabla p^o V_P \right].$$

The above equation can be written [69]:

$$a_P^n \mathbf{U}_P^n = \mathbf{H}(\mathbf{U}) - \nabla p^n, \tag{3.122}$$

where $\mathbf{H}(\mathbf{U})$ includes all terms apart from the pressure gradient at the new time step and the diagonal term $a_P^n \mathbf{U}_P^n$ where the subscript n represents the new time level.

The continuity equation is discretised as follows:

$$\sum_f \mathbf{S} \cdot \mathbf{U}_f = 0, \tag{3.123}$$

and is enforced for every time step. By interpolating $\mathbf{U}_P$ obtained from Eqn. (3.122) onto the cell faces and inserting it into Eqn. (3.123) there results:

$$\sum_f \mathbf{S} \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_P^n} \right)_f - \sum_f \left(\frac{1}{a_P^n} \right)_f (\mathbf{S} \cdot \nabla p^n)_f = 0. \tag{3.124}$$

The term $(\mathbf{S} \cdot \nabla p^n)_f$ is evaluated using Eqn. (3.50) to ensure strong coupling between the velocity and pressure; as originally proposed by Rhie and Chow procedure [130]. The terms $\left(\frac{\mathbf{H}(\mathbf{U})}{a_P^n} \right)_f$ and $(\frac{1}{a_P^n})_f$ are interpolated onto the face by using Eqn. (3.14), which is a second-order scheme. If the interpolation is not of second-order accuracy,

it may result in unboundedness of the pressure field [87, 120]. The pressure equation can be written in a differential form, thus [69]:

$$\nabla \cdot \left(\frac{1}{a_P} \nabla p \right) = \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right). \tag{3.125}$$

The mass flux through a cell face can be obtained by using Eqn. (3.124) as follows [69]:

$$F = \mathbf{S} \cdot \left[\left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f - \left(\frac{1}{a_P} \right)_f (\nabla p)_f \right]. \tag{3.126}$$

Eqn. (3.126) ensures that the face fluxes satisfy the continuity equation if the pressure equation is satisfied. Even though the continuity is satisfied, face fluxes may be under or over-predicted as a consequence of approximations used for different terms in Eqn. (3.126). Therefore, the mass-flux error comes from interpolation of $\frac{\mathbf{H}(\mathbf{U})}{a_P}$, $\frac{1}{a_P}$ and from the evaluation of $(\mathbf{S} \cdot \nabla p)_f$. This can be written in the following form:

$$\begin{aligned} F &= \mathbf{S} \cdot \left(\underline{\frac{\mathbf{H}(\mathbf{U})}{a_P}} + \mathbf{e}_{interpolation}(\frac{\mathbf{H}(\mathbf{U})}{a_P}) \right)_f \\ &\quad - \left(\underline{\frac{1}{a_P}} + e_{interpolation}(\frac{1}{a_P}) \right)_f (\mathbf{S} \cdot \nabla p + e_{snGrad}(\mathbf{S} \cdot \nabla p))_f \\ &= \mathbf{S} \cdot \left(\underline{\frac{\mathbf{H}(\mathbf{U})}{a_P}} \right) - \left(\underline{\frac{1}{a_P}} \right)_f (\mathbf{S} \cdot \nabla p)_f + e_{flux} \end{aligned} \tag{3.127}$$

where e_{flux} from Eqn. (3.127) has the following form:

$$\begin{aligned} e_{flux} &= \mathbf{S} \cdot \mathbf{e}_{interpolation}(\frac{\mathbf{H}(\mathbf{U})}{a_P}) - \left(\frac{1}{a_P} \right)_f e_{snGrad}(\mathbf{S} \cdot \nabla p) - e_{interpolation}(\frac{1}{a_P})(\mathbf{S} \cdot \nabla p)_f \\ &\quad - e_{interpolation}(\frac{1}{a_P}) e_{snGrad}(\mathbf{S} \cdot \nabla p). \end{aligned} \tag{3.128}$$

Here

- $\mathbf{e}_{interpolation}(\frac{\mathbf{H}(\mathbf{U})}{a_P})$ is the error originating from the interpolation of $\frac{\mathbf{H}(\mathbf{U})}{a_P}$.
- $e_{interpolation}(\frac{1}{a_P})$ is the error in the interpolation of $\frac{1}{a_P}$.
- $e_{snGrad}(\mathbf{S} \cdot \nabla p)$ is the error in the pressure-surface-normal gradient approximated by Eqn. (3.50).

The truncation errors for interpolated terms and the surface-normal gradient are given by Eqn. (3.23) and Eqn. (3.56), respectively. e_{flux} is dependent on the error in the pressure-surface-normal gradient $e_{snGrad}(\mathbf{S} \cdot \nabla p)$, thus the influence of pressure on mass fluxes may be large in the regions where the approximation of $(\mathbf{S} \cdot \nabla p)_f$

is not accurate. This error reduces with mesh refinement and is of second-order on uniform meshes, for which $f_x = 0.5$ in Eqs. (3.56) and (3.23), but it becomes a first-order method when the mesh is not uniform due to the pressure-surface-normal gradient which is of first-order on non-uniform meshes.

3.8.2 Algorithms for Pressure-Velocity Coupling

The pressure and momentum equations defined in Eqs. (3.122) and (3.124) are non-linear, because the a_P and $\mathbf{H}(\mathbf{U})$ depend on the velocity field. This non-linearity can be resolved using an iterative procedure. When high temporal accuracy is required the equations must be solved iteratively within each time step until the solution is satisfied to the desired tolerance. Steady flows can be solved by iterating the steady-state equations until the tolerance is met.

The most widely used approach for solving coupled equations is the **segregated approach**, where equations are linearised and solved in sequence one after another [67, 114, 115]. These algorithms have gained popularity within the CFD community and are regarded as the best compromise between accuracy and computational cost [48]. One of the most widely used algorithms based on this approach is SIMPLE [114, 115] which will be presented next.

The SIMPLE Algorithm

The SIMPLE algorithm is widely used for steady-state calculations. SIMPLE stands for Semi-Implicit Method for Pressure-Linked Equations. It consists of the following steps:

1. **Momentum predictor.** The velocity field is calculated from the momentum equation by using the pressure and velocity from the previous iteration or an initial guess [115]. The equation is under-relaxed, using an under-relaxation factor α_U, $(0 < \alpha_U < 1)$, to reduce the influence of non-linearity and to increase the diagonal dominance of the matrix. The resulting velocity field does not necessarily satisfy the continuity equation.

2. **Pressure solution.** The pressure equation is formulated using the predicted velocities and then solved to obtain the new pressure field.

3. **Velocity correction.** Using the new pressure field, a new field of conserved fluxes is calculated using Eqn. (3.126). The associated velocity field which satisfies continuity is calculated from Eqn. (3.122), but it will not satisfy the momentum equation until the overall process has converged. The solution of the pressure field is under-relaxed to minimise the effect of non-linearity [69]:

$$p^{new} = p^{old} + \alpha_p(p^p - p^{old}). \tag{3.129}$$

Here p^p is the solution of the pressure equation, p^{old} is the solution from the previous iteration or an initial guess. p_{new} stands for the pressure field which will be used in the next iteration and α_p is the **pressure under-relaxation factor** with limits $0 < \alpha_p < 1$.

4. The procedure is stopped if the solution has converged; otherwise it is repeated from Step 1.

The energy and turbulence equations described in Chapter 2 are solved after step 3, using the corrected fluxes and velocities.

3.9 Summary and Conclusions

This chapter has presented a second-order FV discretisation practice together with the errors which may arise during the discretisation process. It has been shown that the accuracy is dependent on the mesh quality and it reduces with the increase of skewness and/or non-orthogonality. Analysis of errors on different types of cells has shown that cell shapes which have face pairs are expected to give better accuracy because the errors committed on the faces of the face pair cancel each other out. It was also shown that the discretisation of the surface-normal gradient is of first-order when the mesh is not uniform, and it is therefore advisable to keep the mesh as uniform as possible, especially for diffusion-dominated problems. The mass flux, Eqn. (3.126), is dependent on pressure and the influence of pressure reduces as a second-order error on uniform meshes while it becomes a first-order error on non-uniform ones.

Chapter 4

Error Estimation

4.1 Introduction

In the previous chapter the Finite Volume Discretisation has been described. A new estimator of the discretisation error in the FV solution, developed during this study, will be presented in this chapter.

Before any judgement on the quality of error estimators can be given, it is necessary to define their desired properties. The properties required for an error estimator are described by Jasak [69]:

1. It should provide reliable information about the error distribution and its magnitude.

2. The absolute value of the estimated error should have the dimensions of the field it is applied to, so that it is easily understood.

3. It has to work well on meshes which may be too coarse for adequate resolution.

4. It should be applicable to the various kinds of transport equations encountered in CFD.

5. The estimated error should tend asymptotically to the exact error with mesh refinement.

6. The error estimator should modestly overestimate the error.

7. The cost of error estimation should not be too large.

The **Effectivity index** ζ is a measure of the quality of error estimation and is defined as:

$$\zeta = \frac{|e|}{|E|} \tag{4.1}$$

where e is the estimated disretisation error and E is the exact-discretisation error in the solution. From Eqn. (4.1) it is clear that it is desirable to have effectivity indices close to unity from above. Effectivity indices can be assembled on a cell-by-cell basis or evaluated globally for the whole domain.

The remainder of this chapter is organised as follows. A literature survey of previous studies is given in Section 4.2. An analysis of errors on mesh faces, performed in this study, will be presented in Section 4.3. The result is a new error estimator called the Face Residual Error Estimator (FREE) described in Section 4.4.

The performance of FREE is tested in Section 4.5 on a set of cases for which analytical solutions are available.

Some closing remarks are given in Section 4.6.

4.2 Literature Survey

Error detection can be performed by using *error indicators* which can pinpoint the regions where better resolution is needed, but they do not provide information about the error magnitude. Error indicators which monitor pressure and/or density gradients give useful information for problems with discontinuous features (*i.e.* shocks) where fine resolution is needed around them [2, 123, 152], but they may fail to provide useful information when the solution varies smoothly [152]. The Cell Reynolds number can also be used to provide information about where better resolution is needed to suppress spurious oscillations from the CD convection scheme [42].

Error estimators provide additional information, namely the error magnitude, which is important when judging the quality of the solution. A review of error estimation methods used in the FEM will be given first, followed by the methods used in the FVM.

4.2.1 Methods Used in FEM Analysis

A summary of the error estimation methods used in FEM is given by Ainsworth and Oden in [6]. They can be divided into three major groups:

1. **Gradient recovery estimators** are based on an approach of obtaining a continuous approximation of the solution gradient from the solution obtained by the FEM calculation. The difference between the continuous solution gradient, evaluated using a higher-order technique, and the discontinuous FEM solution gradient is then used as an error estimator.

   Error estimates of this kind were proposed and introduced by Zienkiewicz and Zhu. In [164] they presented a technique for calculating the continuous approximation of stresses in an elastic body by using the linear variation operator for the stresses. After the improved solution is known it is compared with the stresses calculated from the FEM solution to compute the errors. Further research of the above error-estimation technique was described in [165, 166] where Zienkiewicz and Zhu have shown that the higher-order variation of the solution should be computed in the Gauss points, where the convergence of the solution is one order higher than the discretisation.

2. **Explicit residual error estimates.** Early work on estimates of this type was presented in [15] by Babuska and Rheinboldt, who proposed an estimator where the residual is weighted by a normalisation factor to give the error bound. Sandri in [133] derived explicit error estimates for the non-Newtonian fluids governed by the power law stress model.

   The accuracy of Explicit Residual Estimators is unsatisfactory as they tend to substantially over-estimate the error [6]. This approach has been improved and extended to the FVM by Jasak [69] by making the normalisation factor dependent on FV fluxes, which improved the efficiency of the estimate.

3. **Implicit residual error estimates** require the solution of a "local problem" posed on each element or a patch of elements. The solution of the "local problem" provides the approximation of the discretisation error in some norm. Examples of this approach can be found in [1, 17, 18, 77, 106, 127, 128] and many more.

   Equilibrated residual error estimates are a group of implicit methods which require that the internal error fluxes and the boundary error fluxes are conserved. Equilibrated error estimators have been studied in [3–5, 7, 8, 108, 109] and shown to produce good estimates of the discretisation error. In [81] Kelly

has discussed the issue of flux equilibration for the implicit error estimates. He has also shown that non-equilibrated fluxes result in over-estimation of the error. This approach has been extended to the FVM by Jasak [69] where it tends to under-estimate the error because of the limited accuracy of the method used to solve the local error problem.

4.2.2 Methods Used in FV Analysis

Error-estimation techniques used in the FV framework can be divided into four main groups:

1. **Truncation analysis.** Every smooth solution can be written in a form of a Taylor series expansion about a given point in space. The series is potentially infinite depending on the number of non-zero gradients, thus:

$$\Phi(\mathbf{x}) = \phi_P + \sum_{n=1}^{\infty} \frac{1}{n!} (\mathbf{x} - \mathbf{x}_P)^n \underbrace{::}_{n} (\underbrace{\nabla\nabla..\nabla}_{n} \phi)_P. \tag{4.2}$$

The FVM approximates the solution by using a truncated form of the Taylor series. The error in the solution is then equal to the sum of truncated terms:

$$e(\mathbf{x}) = \Phi(\mathbf{x}) - \phi(\mathbf{x}) = \sum_{n=p}^{\infty} \frac{1}{n!} (\mathbf{x} - \mathbf{x}_P)^n \underbrace{::}_{n} (\underbrace{\nabla\nabla..\nabla}_{n} \phi)_P, \tag{4.3}$$

where p is the order of discretisation. The error can be estimated by approximating higher gradients from the existing FV solution and inserting them into Eqn. (4.3).

One of the first error-estimation techniques used in the framework of the FVM is Richardson Extrapolation (RE). Examples can be found in [23, 47, 69, 102] *etc.* The RE method requires the existence of solutions of the problem on two or more meshes with different cell sizes to estimate the leading term of the truncation error by solving the system:

$$\begin{aligned} \Phi(\mathbf{x}) - \phi_1(\mathbf{x}) &= C\, h_1^p, \\ \Phi(\mathbf{x}) - \phi_2(\mathbf{x}) &= C\, h_2^p, \end{aligned} \tag{4.4}$$

where $C\, h_i^p$ is the leading term of the truncation error, $\Phi(\mathbf{x})$ is the unknown exact solution and $\phi_i(\mathbf{x})$ is the FV solution on a mesh with density h_i. h_i is defined as a ratio between the volume and area of the cell [69]:

$$h_i = \frac{V}{\sum_f |\mathbf{S}|}. \tag{4.5}$$

By solving the system in Eqn. (4.4), C and $\Phi(\mathbf{x})$ can be obtained and the error can be expressed as follows:

$$e = |\Phi(\mathbf{x}) - \phi_2(\mathbf{x})| = \frac{|\phi_2(\mathbf{x}) - \phi_1(\mathbf{x})|}{\left(\frac{h_1}{h_2}\right)^p - 1}. \tag{4.6}$$

This is not a reliable method because it generally under-estimates the error, especially if the finer mesh is still coarse [69, 96]. Another drawback is the fact that it requires two solutions which make it time consuming.

Methods which can estimate the error by using a single solution obtained on a single mesh started appearing relatively recently [10, 40, 56, 64, 69, 71, 139, 161], *etc.*. A method for the first-order FVM discretisation presented by Ilinca *et al.* in [64] estimates the error as the difference between the computed solution, assumed to be of constant value within the CV, and a reconstructed linearly-varying solution, thus:

$$\begin{aligned} e &= \sqrt{\int_{V_P} [(\phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P) - \phi_P]^2 dV} \\ &= \sqrt{\int_{V_P} (\mathbf{x} - \mathbf{x}_P)^2 dV : (\nabla\phi)_P^2} \\ &= \sqrt{M : (\nabla\phi)_P^2}, \end{aligned} \tag{4.7}$$

where M is the moment of inertia of the CV and $(\nabla\phi)_P$ is the reconstructed gradient at the node P. The estimator accounts for the flow physics through the gradient of ϕ and it also accounts for the size and shape of the control volumes through the moments of inertia. The same approach was followed by Jasak in [69, 71] where the error is estimated using the second-order term in the Taylor Series and applied to the second-order FVM:

$$\begin{aligned} e &= \frac{1}{2} \int_{V_P} [(\mathbf{x} - \mathbf{x}_P)^2 : (\nabla\nabla\phi)_P] \, dV \\ &= \frac{1}{2} \mathbf{M} : (\nabla\nabla\phi)_P, \end{aligned} \tag{4.8}$$

where $(\nabla\nabla\phi)_P$ is calculated from the available solution. This estimator tends to under-estimate the error when the computational mesh is coarse because the higher-order terms neglected by the estimator are still of considerable importance.

An interesting method for error estimation based on truncation analysis can be found in [10, 40, 56, 139]. The error is computed on mesh edges as follows:

$$e = \int_0^l \sqrt{\mathbf{s}(l)^T \bar{\mathbf{H}}(l)\mathbf{s}(l)}\, dl \tag{4.9}$$

where $\mathbf{s}(l)$ is parametric representation of the edge and $\bar{\mathbf{H}}(l)$ is a modified Hessian obtained from the second gradient. The estimator is used to drive a mesh-adaptation procedure aimed at equidistributing the error over every edge in the domain.

An approach to error estimation where the convection and diffusion fluxes are calculated using a fourth-order scheme and compared with the ones from the second-order FV solution was presented by Muzaferija *et al.* in [102, 103]. The source of the truncation error τ_P is defined as:

$$\tau_P = \sum_f [(\rho \mathbf{U} \cdot \mathbf{S})_f (\phi_f^h - \underline{\phi_f}) - (\rho\Gamma)_f ((\mathbf{S} \cdot \nabla\phi)_f^h - \underline{(\mathbf{S} \cdot \nabla\phi)_f})] \tag{4.10}$$

where ϕ_f^h and $(S \cdot \nabla\phi)_f^h$ are evaluated from the FV solution using a fourth-order scheme while $\underline{\phi_f}$ and $\underline{(\mathbf{S} \cdot \nabla\phi)_f}$ are known from the FV discretisation process. The authors assumed that τ_P serves as a source which can be added to the discretised form of the governing differential equation in order to obtain the exact solution. This method under-estimates the error and is even less accurate than Richardson Extrapolation because it cannot detect non-local phenomena. A similar approach to error estimation developed for hyperbolic equations, for which the error may manifest itself away from its source, was presented by Zhang *et al.* in [161]. The source term of the error equation consists of the truncated higher-order terms which are evaluated from the existing solution. The estimated errors are in good agreement with the exact errors in terms of their locations but the magnitudes are under-estimated. It has also been shown that an efficient grid adaptation can be achieved using the estimated error source to drive the adaptive algorithm instead of the error distribution itself, especially on cases where the error is convected away from its source.

2. **Methods based on derived quantities of the solution variable.** The second group of estimators used in the FVM framework was derived upon a fact that the discretisation schemes enforce conservation of the primary quantities for which the equations are being solved, regardless of the mesh resolution, but

none of them conserve their derived properties (*i.e.* kinetic energy ($\frac{1}{2}\mathbf{U}\cdot\mathbf{U}$) and angular momentum ($\mathbf{x}\times\mathbf{U}$) are derived from velocity) except when the mesh is sufficiently fine. Haworth *et al.* [58] presented a method which measures cell-to-cell imbalances of the angular momentum and kinetic energy in order to characterise the solution error. The imbalance of kinetic energy can be written:

$$\begin{aligned} Res(\frac{1}{2}\mathbf{U}\cdot\mathbf{U}) = \nabla\cdot(\rho(\frac{1}{2}\mathbf{U}\cdot\mathbf{U})\mathbf{U}) + \nabla\cdot(P\,\mathbf{U}) - P\,\nabla\cdot\mathbf{U} \\ +\nabla\cdot\left(\frac{2}{3}\,\mu(\nabla\cdot\mathbf{U})\,\mathbf{U}\right) - \nabla\cdot\left[\mu\left(\nabla\mathbf{U}+(\nabla\mathbf{U})^T\right)\cdot\mathbf{U}\right] \\ +\nabla\mathbf{U}:\left[\mu\left(\nabla\mathbf{U}+(\nabla\mathbf{U})^T\right)+\left(\frac{2}{3}\,\mu\nabla\cdot\mathbf{U}\right)\mathbf{I}\right], \end{aligned} \tag{4.11}$$

and the imbalance of angular momentum can be written as follows:

$$Res(\mathbf{x}\times\mathbf{U}) = \nabla\cdot[\rho(\mathbf{x}\times\mathbf{U})\mathbf{U}] - \mathbf{x}\times(\nabla\cdot\sigma) - \rho(\mathbf{x}\times\mathbf{g}), \tag{4.12}$$

where all fields are known from the FV solution and the terms are evaluated using the same approximation as the ones used for the FV solution. The drawback of the method is that it is not capable of estimating the absolute error levels in the velocity field. Jasak [69] generalised the same approach and made it capable of estimating absolute error levels. The error is estimated by calculating the imbalance in the transport equation for the second moment of ϕ defined as $m_\phi = \frac{1}{2}\phi^2$. The equation for the second moment can be obtained by multiplying the general transport equation Eqn. (2.32) with ϕ, thus [69]:

$$Res(m_\phi) = \int_{\Omega_P}\left[\nabla\cdot(\rho\mathbf{U}m_\phi) - \nabla\cdot(\rho\Gamma_\phi\nabla m_\phi) - S_\phi(\phi)\,\phi + \rho\Gamma_\phi(\nabla\phi.\nabla\phi)\right]\,dV. \tag{4.13}$$

The values of all fields in the above equation are known and the imbalance is calculated using the same approximation as the ones used for FV equations. Multiplying Eqn. (4.13) with a normalisation factor enables recovery of the error magnitude, thus:

$$e = 2\sqrt{\frac{|Res(m_\phi)|\,T_{trans}}{V_P}}. \tag{4.14}$$

V_P is the cell volume and T_{trans} is the characteristic time scale of m_ϕ defined as:

$$T_{trans} = \frac{h^2}{|\mathbf{U}|\,h + \Gamma_\phi}, \tag{4.15}$$

where h is a cell size defined in Eqn. (4.5), $|\mathbf{U}|$ is a transport cell velocity and Γ is the diffusion factor. A problem with this error estimator is its inaccuracy on fine meshes. This is due to the higher order variation of the second moment. The transport equation for the higher moment has to be discretised in a higher-order manner than the transport equation for the property to scale the errors correctly. It may happen the error estimate indicates the error exists even if it is zero. This undesirable behaviour is most severe in the regions where the variation of the variable is modest while in the regions of steep gradients it does not have a significant influence on the estimated error.

3. **Residual-based methods** for error estimation in the FV analysis have their origin in similar work in the Finite Element field. In the FVM the concept of residual-based error estimation has been introduced by Jasak [69] and Angermann [12, 13]. The residual is a function measuring how well the governing differential equations are approximated over the computational cell, and is defined as [69]:

$$\begin{aligned} Res(\phi) = {} & SuV_P + Sp\,\phi_P V_P \\ & - \sum_f F_f(\phi_P + (\mathbf{x}_f - \mathbf{x}_P)\cdot(\nabla\phi)_P) + \sum_f (\rho\Gamma)_f(\mathbf{S}_f \cdot (\nabla\phi)_P), \end{aligned} \quad (4.16)$$

where all fields are known from the available FV solution. The residual is consistent with the discretisation practice and is zero when the solution variation is linear. It does not give information about the error magnitude directly because of its different dimension from ϕ. Jasak [69] has proposed a normalisation practice to extract the magnitude of the error from the residual which consists of the convection and diffusion transport coefficients and the linearised part of the source term Sp, thus:

$$F_{norm} = \sum_f max(F, 0) + \sum_f \left[|\mathbf{S}| \frac{(\rho\Gamma_\phi)_f}{|\mathbf{d}|} \right] + SpV_P. \quad (4.17)$$

and the error is calculated from:

$$e = \frac{Res(\phi)}{F_{norm}} \quad (4.18)$$

The normalisation practice assumes first-order UD discretisation of the convection term, the first term on the r.h.s. in Eqn. (4.17), which ensures that the convection contribution to the normalisation factor is always positive. This has

an adverse effect on the accuracy of error estimation especially on meshes with insufficient resolution where it may result in an incorrect distribution of the estimated discretisation error. In [12] Angermann described a local $a-posteriori$ error estimates for convection-dominated problems. In [13] Angermann *et al.* modified the method and applied it to density-driven time dependent problems. The proposed estimate is assembled from four parts which represent the local indicators. The full error estimate is defined as a sum of local error indicators. The first indicator is the residual and the other indicators account for the imbalance of convection fluxes over the computational cell. This method is not easy to understand and it does not measure the absolute value of the error. It is also not accurate enough for engineering application because of severe over-estimation of the error.

4. **The Equilibrated residual method** is a technique which was used first in the framework of FEM. Its extension to FVM was presented by Jasak in [69, 75]. The method is not as successful as its FEM counterpart because the procedure used for solving local problems is not accurate enough, especially on coarse meshes.

Even though the FV error estimation has undergone a rapid development during the last decade there is still some room for improvement. Truncation methods which can estimate the error from a single solution have been developed, but they remain prone to under-estimation of the error. Residual-based methods are considered to be the most promising group of methods and will be utilised in this study. In order to be accurate, residual-based methods require conserved error fluxes through the cell faces and a reliable procedure for extracting the error magnitude from the residual. The approach taken during this study is aimed towards developing an estimator on mesh faces where balanced error fluxes and the data needed for extraction of error magnitude are available.

4.3 Error Transport Through a Face

The FV discretisation practice is conservative for each CV and is performed by approximating fluxes through the CV faces such that they are in balance with the sources/sinks acting within the CV. The approximated fluxes may contain errors

which are transported through cell faces and these errors influence the accuracy in the neighbouring cells.

Due to the property of the FV discretisation, the general transport equation, Eqn. (2.32), can have three equally valid solutions on every internal face: two extrapolated from both sides of the face $(\phi_f)_P$, $(\phi_f)_N$, Fig. 4.1, and the interpolated value onto the face $\underline{\phi_f}$ [69].

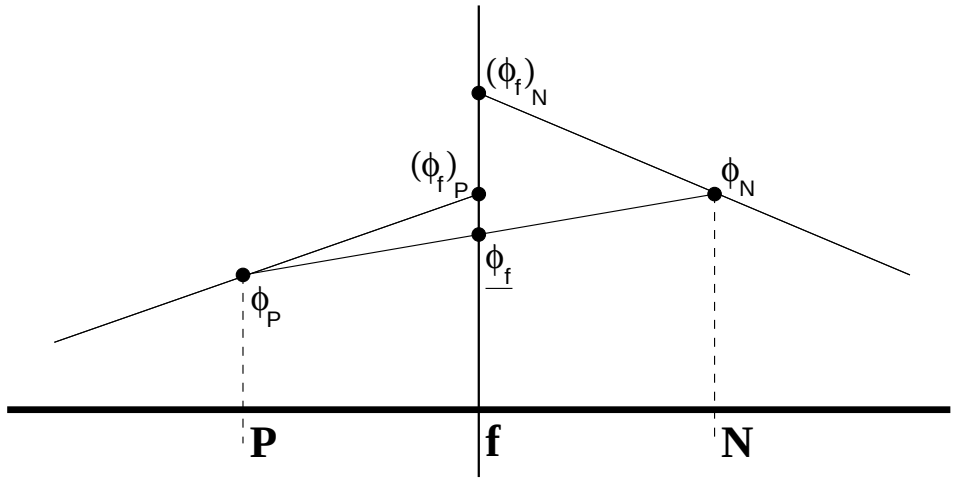


Figure 4.1: Inconsistency of the interpolated values on the face

The value on the face extrapolated from the cell P can be calculated using a linear variation of ϕ in the cell, Eqn. (3.3), such as [69]:

$$(\phi_f)_P = \phi_P + (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla\phi)_P, \tag{4.19}$$

while the value from the cell N can be obtained in the same manner as for the cell P, thus:

$$(\phi_f)_N = \phi_N + (\mathbf{x}_f - \mathbf{x}_N) \cdot (\nabla\phi)_N. \tag{4.20}$$

The interpolated value on the face can be calculated using Eqn. (3.14) and the gradients in the cells P and N can be evaluated using the second-order accurate methods, Eqs. (3.26) and (3.30). When the variation of the solution is linear, the gradients in the cells and on the face are equal and the extrapolated values $(\phi_f)_P$ and $(\phi_f)_N$ become equal to the interpolated value ϕ_f resulting in the discretisation error equal to zero [69].

A transport equation for error can be derived by substituting Eqn. (1.2) into the

transport equation Eqn. (2.32). For steady state conditions it reads:

$$\int_{V_P} \nabla \cdot (\rho \mathbf{U}(E(\mathbf{x}) + \phi(\mathbf{x}))) \, dV - \int_{V_P} \nabla \cdot (\rho \Gamma \nabla (E(\mathbf{x}) + \phi(\mathbf{x}))) \, dV = \int_{V_P} S_\phi(\phi) \, dV. \tag{4.21}$$

The transient term is omitted as the focus here is on the discretisation error arising from the spatial terms. The terms can be reordered such that the terms known from the FV solution are put on the right-hand side and denoted by $Res(\phi)$, and the terms governing the error transport are put on the left-hand side. The transport equation for error then has the form [6]:

$$\int_{V_P} \nabla \cdot (\rho \mathbf{U} E(\mathbf{x})) \, dV - \int_{V_P} \nabla \cdot (\rho \Gamma (\nabla E(\mathbf{x}))) \, dV = Res(\phi) \tag{4.22}$$

where the residual term $Res(\phi)$ on the right-hand side has the following form:

$$Res(\phi) = \int_{V_P} [S_\phi(\phi) - \nabla \cdot (\rho \mathbf{U} \phi(\mathbf{x})) + \nabla \cdot (\rho \Gamma (\nabla \phi(\mathbf{x})))] \, dV. \tag{4.23}$$

The residual acts as a source for the error, Eqn. (4.22), and has the same form as the transport equation Eqn. (2.32). Its approximated form reads [69]:

$$Res(\phi) = S_\phi(\phi) V_P - \sum_f F_f \phi_f + \sum_f (\rho \Gamma)_f (\mathbf{S} \cdot \nabla \phi)_f. \tag{4.24}$$

If the evaluation of ϕ_f and $(\mathbf{S} \cdot \nabla \phi)_f$ terms in the residual is performed using same procedures as for the convection Section 3.3.1 and the diffusion term Section 3.3.2, respectively, the residual is zero even though the error exists. This is a consequence of conservation which is then satisfied. But ϕ_f can also be approximated by using Eqn. (4.19) which is valid within the computational cell and $(\mathbf{S} \cdot \nabla \phi)_f$ can be approximated by:

$$(\mathbf{S} \cdot \nabla \phi)_f = (\mathbf{S}_f \cdot (\nabla \phi)_P). \tag{4.25}$$

When the variation of the solution is of higher-order than linear the residual evaluated by using Eqn. (4.19) and Eqn. (4.25) [69]:

$$Res_P(\phi) = S_\phi(\phi) V_P - \sum_f F_f (\phi_P + (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla \phi)_P) + \sum_f (\rho \Gamma)_f (\mathbf{S}_f \cdot (\nabla \phi)_P), \tag{4.26}$$

may not be equal to zero if the mesh is not sufficiently fine. Thus, it provides an estimate of the magnitude of the error source in the computational cell. Regarding the source term, the discretised source term Section 3.3 cannot be written as sum over cell faces and its influence on the discretisation error is neglected. The residual is consistent with the discretisation practice and it is zero when the variation of ϕ

is linear. It is sensitive to the shape of the cells and should therefore capture the mesh induced errors because the extrapolated values onto the faces are dependent on $(\mathbf{x}_f - \mathbf{x}_P)$.

As was shown by Ainsworth *et al.* [3, 8] and Jasak [69], the residual can be split into a sum of partial face residuals from all the faces bounding the computational cell, thus:

$$Res_P(\phi) = \sum_f Res_f^P(\phi), \tag{4.27}$$

where the partial face residual $Res_f^P(\phi)$ is defined as [69]:

$$Res_f^P(\phi) = F_f(\underline{\phi_f} - (\phi_P + (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla\phi)_P)) + (\rho\Gamma)_f((\mathbf{S} \cdot (\nabla\phi)_P) - \underline{(\mathbf{S} \cdot \nabla\phi)_f}). \tag{4.28}$$

Here, $\underline{\phi_f}$ represents the interpolated value on the face shown in Fig. 4.1 and $\underline{(\mathbf{S} \cdot \nabla\phi)_f}$ is the surface-normal gradient on the face calculated using Eqn. (3.50). The partial face residual represents the imbalance in values and gradients on the face originating from cell P Fig. 4.1, only. The partial face residual can detect how well the governing equation is satisfied on the faces comprising the cell and can therefore give information about the direction in which the error is transported.

When the left-hand side of Eqn. (4.22) is discretised in a second-order manner for a representative cell, we get:

$$\sum_f F_f e_f^P - \sum_f (\rho\Gamma)_f (\mathbf{S}_f \cdot (\nabla e)_f^P) =$$
$$\sum_f F_f(\underline{\phi_f} - (\phi_P + (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla\phi)_P)) + \sum_f (\rho\Gamma)_f((\mathbf{S} \cdot (\nabla\phi)_P) - \underline{(\mathbf{S} \cdot \nabla\phi)_f}) \tag{4.29}$$

from which it follows that:

$$\begin{aligned} e_f^P &= \underline{\phi_f} - (\phi_P + (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla\phi)_P), \\ (\mathbf{S} \cdot \nabla e)_f^P &= \underline{(\mathbf{S} \cdot \nabla\phi)_f} - (\mathbf{S} \cdot (\nabla\phi)_P). \end{aligned} \tag{4.30}$$

After some algebra, a link between the error and the error in gradient on the face can be found, thus:

$$e_f^P = (\mathbf{x}_f - \mathbf{x}_P) \cdot (\nabla e)_f^P. \tag{4.31}$$

From Eqn. (4.30) an interesting property of the error field can be found. The error can be transported in the opposite direction to the solution fluxes, because from Eqn. (4.30) it is clear that the convection $F_f e_f^P$ and diffusion $(\mathbf{S}_f \cdot (\nabla e)_f^P)$ error

fluxes may have different signs from the fluxes $F_f\underline{\phi_f}$ and $(\rho\Gamma)_f\underline{(\mathbf{S}\cdot\nabla\phi)_f}$ used to approximate convection and diffusion terms. The following section will show how this can be used to develop an error estimate.

4.4 Face Residual Error Estimator

The estimator developed during this study follows the approach of Jasak [69, 74], but is defined on the faces because the fluxes used for normalisation of the residual are available. Additionally, the estimator defined on the faces provides information on which faces largely contribute to the error. This can be used for anisotropic mesh refinement, see Chapter 5.

The partial face residuals, defined in the previous section Eqn. (4.28), provide the information about the error flux through a face. But, for every internal face there exist two partial face residuals from both sides of that face. The partial face residual from the cell P has the form, Fig. 4.1:

$$Res_f^P(\phi) = F_f(\underline{\phi_f} - (\phi_P + (\mathbf{x}_f - \mathbf{x}_P)\cdot(\nabla\phi)_P)) + (\rho\Gamma)_f\left((\mathbf{S}_f\cdot(\nabla\phi)_P) - \underline{(\mathbf{S}\cdot\nabla\phi)_f}\right) \tag{4.32}$$

and from the cell N:

$$Res_f^N(\phi) = F_f(\underline{\phi_f} - (\phi_N + (\mathbf{x}_f - \mathbf{x}_N)\cdot(\nabla\phi)_N)) + (\rho\Gamma)_f\left((\mathbf{S}_f\cdot(\nabla\phi)_N) - \underline{(\mathbf{S}\cdot\nabla\phi)_f}\right). \tag{4.33}$$

When these partial face residuals from both sides of the face are summed, the full face residual is obtained:

$$\begin{aligned} Res_f(\phi) =& Res_f^P(\phi) + Res_f^N(\phi) \\ =& F_f((\phi_N + (\mathbf{x}_f - \mathbf{x}_N)\cdot(\nabla\phi)_N) - (\phi_P + (\mathbf{x}_f - \mathbf{x}_P)\cdot(\nabla\phi)_P)) \\ &+(\rho\Gamma)_f\mathbf{S}_f\cdot((\nabla\phi)_P - (\nabla\phi)_N). \end{aligned} \tag{4.34}$$

Because $Res_f(\phi)$ is assembled from information in the adjoining cells, only, its calculation is inexpensive and is unique for each face. On boundary faces the residual is calculated using Eqn. (4.32), and is equal to the partial face residual for that face.

The face residual is dependent both on the mesh and the solution itself. Dependence on $(\mathbf{x}_f - \mathbf{x}_N)$ and $(\mathbf{x}_f - \mathbf{x}_P)$ makes the residual sensitive to mesh density and skewness. It becomes larger with the increase of mesh skewness. The solution variables, (*eg.* $\mathbf{S}_f\cdot((\nabla\phi)_P - (\nabla\phi)_N)$) are also sensitive to mesh size, because the

closer P and N are, the smaller is the difference between the gradients at those nodes, but only if the solution is non-singular.

The face residual does not have the same dimension as the ϕ field, so it is necessary to introduce a normalisation procedure which enables extraction of the error magnitude from the residual. The normalisation factor can be assembled using the information available at the face, namely the magnitude of the convection and diffusion transport coefficients, thus:

$$\begin{aligned} F_{conv} &= |F_f|, \\ F_{diff} &\approx (\rho\Gamma)_f \frac{|\mathbf{S}|}{\frac{1}{2}|\mathbf{x}_N - \mathbf{x}_P|}, \\ F_{norm} &= F_{conv} + F_{diff}, \end{aligned} \tag{4.35}$$

where F_f and $(\rho\Gamma)_f$ are the same as in Eqs. (3.32) and (3.49), respectively. The factor $\frac{1}{2}$ in F_{diff} is included to approximate the distances from the cell centre to the face centre on both sides of the face as in Eqn. (4.31). The analysis of the normalisation practice will be presented in the next section.

Finally, on the internal faces the Face Residual Error Estimator (FREE) is defined as:

$$e_f = \frac{Res_f(\phi)}{F_{norm}} \tag{4.36}$$

while on boundary faces it is calculated according to the type of boundary condition imposed. Thus, for example:

$$e_f = \begin{cases} 0 & \text{on fixed value boundaries} \\ \frac{Res_P(\phi)}{F_{norm_b}} & \text{on other boundaries} \end{cases} \tag{4.37}$$

where F_{norm_b} is defined as:

$$\begin{aligned} F_{conv_b} &= |\rho\mathbf{U} \cdot \mathbf{S}|_f \\ F_{diff_b} &= (\rho\Gamma)_f \frac{|\mathbf{S}|}{|\mathbf{x}_f - \mathbf{x}_P|} \\ F_{norm_b} &= F_{conv_b} + F_{diff_b} \end{aligned} \tag{4.38}$$

4.4.1 Analysis of the Normalisation Practice

An analysis of the normalisation practice will now be presented term by term in order to establish a link between the estimated face error and Eqn. (4.30). The estimated face error will also be expressed in a form of truncation error, which provides information about the scaling of the estimated error with mesh refinement.

Convection Term

The face residual for the convection-dominated problems with no diffusion can be written:

$$Res_f(\phi) = (\rho\mathbf{U}\cdot\mathbf{S})_f((\phi_N + (\mathbf{x}_f - \mathbf{x}_N)\cdot(\nabla\phi)_N) - (\phi_P + (\mathbf{x}_f - \mathbf{x}_P)\cdot(\nabla\phi)_P)). \quad (4.39)$$

The estimated error is obtained by normalising the above residual, Eqn. (4.39), with the convection part of the normalisation factor, Eqn. (4.35). The result is:

$$\begin{aligned} e_f &= sgn(\rho\mathbf{U}\cdot\mathbf{S})_f[(\phi_N + (\mathbf{x}_f - \mathbf{x}_N)\cdot(\nabla\phi)_N) - (\phi_P + (\mathbf{x}_f - \mathbf{x}_P)\cdot(\nabla\phi)_P)] \\ &= sgn(F)_f[(\phi_N - \phi_f + (\mathbf{x}_f - \mathbf{x}_N)\cdot(\nabla\phi)_N) - (\phi_P - \phi_f + (\mathbf{x}_f - \mathbf{x}_P)\cdot(\nabla\phi)_P)] \\ &= sqn(F)_f[e_f^P - e_f^N] \end{aligned} \quad (4.40)$$

where $sgn(F)$ is defined as follows:

$$sgn(F) = \begin{cases} 1 & F \geq 0 \\ -1 & F < 0. \end{cases} \quad (4.41)$$

e_f^P and e_f^N can be calculated from Eqn. (4.30). Eqn. (4.40) can be rewritten in a form of truncation error by using the Taylor expansions in Eqs. (3.18) and (3.19), thus:

$$\begin{aligned} e_f &= \phi_P + (\mathbf{x}_P - \mathbf{x}_f)\cdot(\nabla\phi)_f + \frac{1}{2}(\mathbf{x}_P - \mathbf{x}_f)^2 : (\nabla\nabla\phi)_f \\ &\quad + (\mathbf{x}_f - \mathbf{x}_P)\cdot((\nabla\phi)_f + (\mathbf{x}_P - \mathbf{x}_f)\cdot(\nabla\nabla\phi)_f) \\ &\quad - \phi_N + (\mathbf{x}_N - \mathbf{x}_f)\cdot(\nabla\phi)_f + \frac{1}{2}(\mathbf{x}_N - \mathbf{x}_f)^2 : (\nabla\nabla\phi)_f \\ &\quad - (\mathbf{x}_f - \mathbf{x}_N)\cdot((\nabla\phi)_f + (\mathbf{x}_N - \mathbf{x}_f)\cdot(\nabla\nabla\phi)_f) \\ &= -\frac{1}{2}[(\mathbf{x}_N - \mathbf{x}_f)^2 - (\mathbf{x}_f - \mathbf{x}_P)^2] : (\nabla\nabla\phi)_f \\ &= -\frac{1}{2}[(2f_x - 1) - 2\psi]|\mathbf{d}|^2(\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_f), \end{aligned} \quad (4.42)$$

where f_x is a linear interpolation factor defined in Eqn. (3.17), ψ is a measure of skewness, Eqn. (3.8) and $\mathbf{d}$ is a distance vector, Fig. 3.3. The estimator is consistent with the second-order discretisation practice and it is also sensitive to mesh skewness. Eqn. (4.42) is similar to the truncation error for the CD scheme and is expected to provide best results in that case. The truncation error for the UD scheme, Eqn. (3.39), is of first-order and it should not be used because it may result in under-estimation of the error on fine meshes. Examples of such error behaviour is shown by Jasak [69].

Diffusion Term

The residual when convection is not present is:

$$Res_f(\phi) = (\rho\Gamma)_f \mathbf{S} \cdot ((\nabla\phi)_P - (\nabla\phi)_N). \tag{4.43}$$

The expression can be recast using F_{diff} from Eqn. (4.35) and Eqn. (4.43) into Eqn. (4.36) to give:

$$\begin{aligned} e_f &= \frac{1}{2}|\mathbf{x}_N - \mathbf{x}_P|\hat{\mathbf{d}} \cdot ((\nabla\phi)_P - (\nabla\phi)_N) \\ &= \frac{1}{2}|\mathbf{x}_N - \mathbf{x}_P|\hat{\mathbf{d}} \cdot [((\nabla\phi)_P - (\nabla\phi)_f) - ((\nabla\phi)_N - (\nabla\phi)_f)]. \end{aligned} \tag{4.44}$$

As in case of convection term, Eqn. (4.44) can be recast in a form of truncation error, thus:

$$\begin{aligned} e_f &\approx \frac{1}{2}|\mathbf{x}_N - \mathbf{x}_P|\hat{\mathbf{d}} \cdot [\\ &\quad ((\nabla\phi)_f + (\mathbf{x}_P - \mathbf{x}_f) \cdot (\nabla\nabla\phi)_f) \\ &\quad - ((\nabla\phi)_f + (\mathbf{x}_N - \mathbf{x}_f) \cdot (\nabla\nabla\phi)_f)] \\ &\approx -\frac{1}{2}|\mathbf{d}|^2 \, (\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_f) \\ &\approx -\frac{1}{2}\frac{|\mathbf{d}_n|^2}{\cos^2\alpha_N} \, (\hat{\mathbf{d}}^2 : (\nabla\nabla\phi)_f) \end{aligned} \tag{4.45}$$

Eqn. (4.45) is consistent with the second-order discretisation practice as it is a

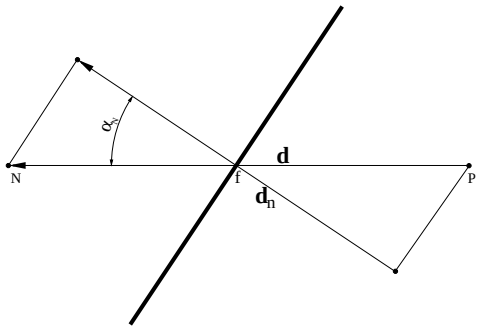


Figure 4.2: Distance between points on non-orthogonal mesh

function of $|\mathbf{d}_n|^2$ which is the distance between the grid nodes in the direction of normal vector Fig. 4.2. According to the discretisation practice for diffusion term, Eqn. (3.50), the relation between $|\mathbf{d}_n|$ and $|\mathbf{d}|$ is:

$$|\mathbf{d}| = \frac{|\mathbf{d}_n|}{\cos\alpha_N}.$$

Eqn. (4.45) is not similar to the truncation error for the surface-normal gradient, Eqn. (3.56). The estimated error is proportional to the second gradient of ϕ while Eqn. (3.56) shows that the error is proportional to the second gradient of ϕ and of first-order on non-uniform meshes, while on uniform meshes it is of second-order and dependent on the third gradient of ϕ. From there it follows that high accuracy of error estimation cannot be expected for diffusion-dominated problems because the truncation errors are not consistent.

4.5 Examples

Tests of the face residual error estimator were performed on three cases where analytical solutions of the governing equations are available. The cases included diffusion and convection-dominated ones.

In order to test the order of the discretisation procedure and the error estimator, the exact and the estimated errors are monitored with uniform mesh refinement. The quality of the error estimation is assessed in terms of the absolute magnitudes of both the maximum and mean errors. The mean error can be monitored for cell-based and face-based variables. For the cell-based variables the mean error is calculated using:

$$e_{meanCell} = \frac{\sum_{i=0}^{n} |e_i| V_i}{\sum_{i=0}^{n} V_i} \tag{4.46}$$

and for the face-based variables:

$$e_{meanFace} = \frac{\sum_{i=0}^{n} |e_i||\mathbf{S}_i|}{\sum_{i=0}^{n} |\mathbf{S}_i|} \tag{4.47}$$

where $|e_i|$ represents the magnitude of the error for the $i-th$ face or cell and n is a number of cells or faces in the mesh.

The values of the exact error magnitudes are calculated at the grid nodes and plotted as cell values, where the cell colour corresponds to the error magnitude for that cell. Estimated errors are also plotted as cell values, where the value in the cell is the weighted average of the error on the faces forming the cell, thus:

$$e_{cell} = \frac{\sum_f |e_f||\mathbf{S}_f|}{\sum_f |\mathbf{S}_f|}. \tag{4.48}$$

4.5.1 Planar Jet

In this test case, a fluid enters a planar two-dimensional domain via a slot, forming a jet, see Fig. 4.5. The solution of the Navier-Stokes equations, which can be found in

[112, 135], was derived by assuming the flow is incompressible with uniform pressure and a small cross-stream velocity. The analytical solution for the axial velocity U_x has the form [112]:

$$U_x = \frac{A}{B} x^{-\frac{1}{3}} sech^2(\frac{y}{B} x^{-\frac{2}{3}}), \tag{4.49}$$

and the cross-stream velocity component U_y is given by:

$$U_y = -\frac{1}{3} A x^{-\frac{2}{3}} tanh(\frac{y}{B} x^{-\frac{2}{3}}) + \frac{2}{3}\frac{A}{B} y x^{-\frac{4}{3}} sech^2(\frac{y}{B} x^{-\frac{2}{3}}) \tag{4.50}$$

where

$$A = (\frac{9}{2}\nu M_j)^{\frac{1}{3}}, \tag{4.51}$$

$$B = (\frac{48\nu^2}{M_j})^{\frac{1}{3}}, \tag{4.52}$$

and $M_j = U_0^2 h$ is the jet momentum. The value of M_j used for calculations is:

$$M_j = 0.1\,[m^3/s^2], \tag{4.53}$$

and the kinematic viscosity is set to:

$$\nu = 0.0012\,[m^2/s]. \tag{4.54}$$

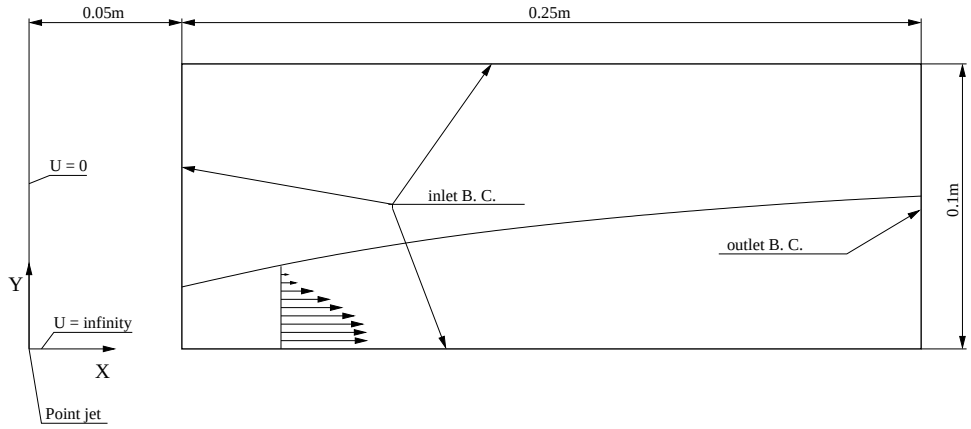


Figure 4.3: Solution domain and boundary conditions for the jet case

The computational domain and the boundary conditions used for this case are shown in Fig. 4.3, where the left, top and bottom boundaries were treated as inlets. The velocities at those boundaries are set to values of the analytical solution. An outlet was prescribed at the right boundary and the dynamic pressure is set to zero there. The convection scheme is Central Differencing (CD). Gradients of all variables

are evaluated by the Gauss-divergence Theorem. Eqn. (4.49) has a singularity at the origin of the coordinate system, hence the domain was moved downstream in order not to have the singularity within the domain. The coordinates of the bottom-left corner of the domain were located at (0.05m, 0m) and the top-right corner was positioned at (0.3m, 0.1m).

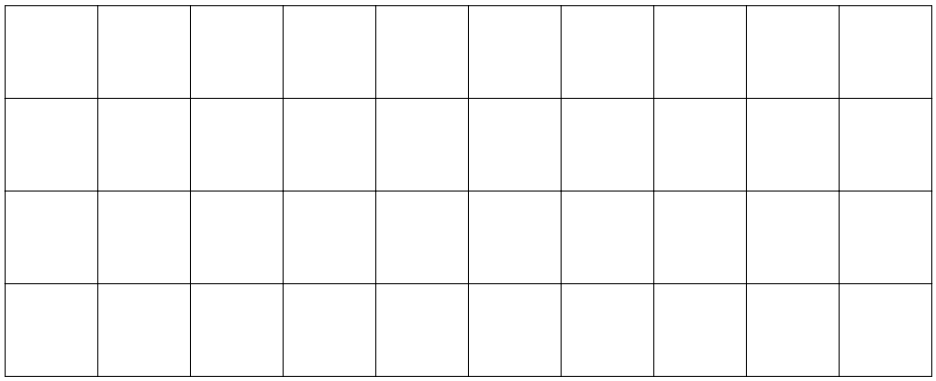


Figure 4.4: Starting mesh for the jet case (10 x 4 cells)

Figs. 4.5 and 4.6 show the velocity and pressure fields. The velocity and its gradient are high in the region where the jet enters the domain and reduce downstream due to the influence of diffusion. Fig. 4.6 shows that the pressure field is not uniform within the domain as was assumed during the derivation of analytical solution.

To examine the quality of error estimation on meshes with different cell size, solutions were obtained on five meshes. The starting uniform mesh had ten cells in the stream-wise direction and only four cells in the cross-stream direction, Fig. 4.4. Subsequent meshes were constructed by halving the cell size in both directions.

Fig. 4.7(a) presents the exact velocity-error-magnitude field on the 80 x 32 cells mesh. The error is largest in the vicinity of the region where the jet enters the domain, where the gradients of pressure and velocity are high. The estimated error distribution obtained by using FREE is shown in Fig. 4.7(b). The locations of large and small errors are in moderate agreement because the error is over-estimated. The differences are also caused by averaging of face errors, Eqn. (4.48), and by the modelling error in the analytical solution. The latter originates from the neglect of pressure gradients and other approximations introduced when developing the analytical solution. Figs. 4.7(a) and 4.7(b) show that the maximum exact error is downstream of the maximum estimated error. This is due to the pressure gradient,

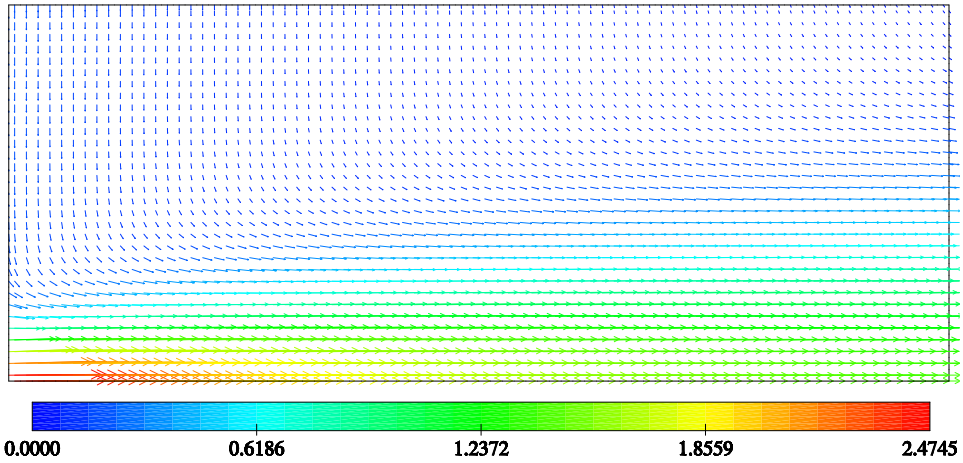


Figure 4.5: Velocity field for the jet case [m/s] (80 x 32 mesh)

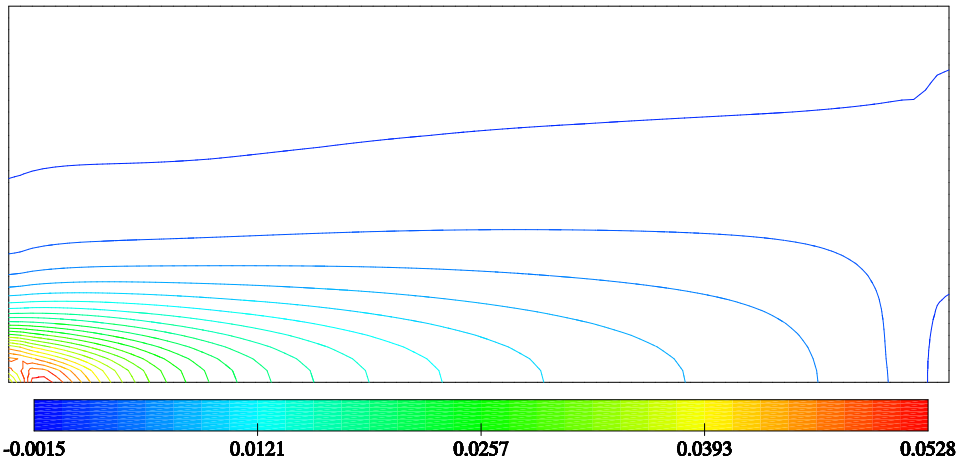
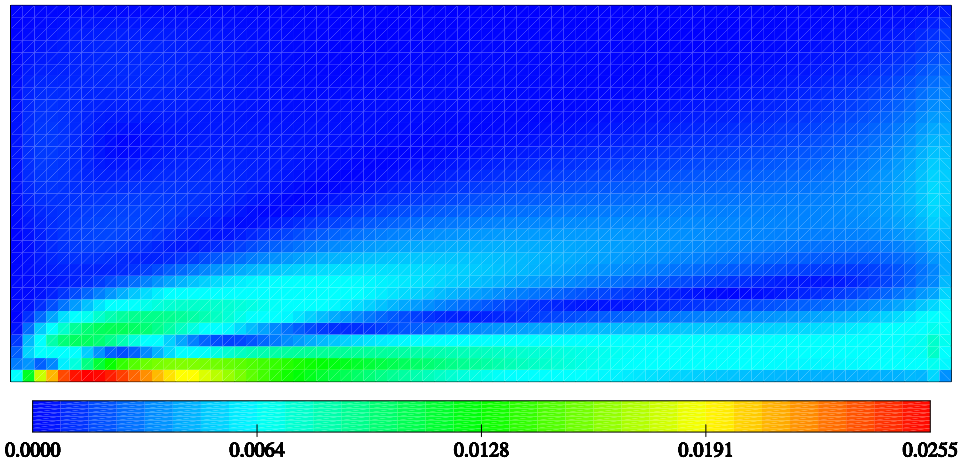
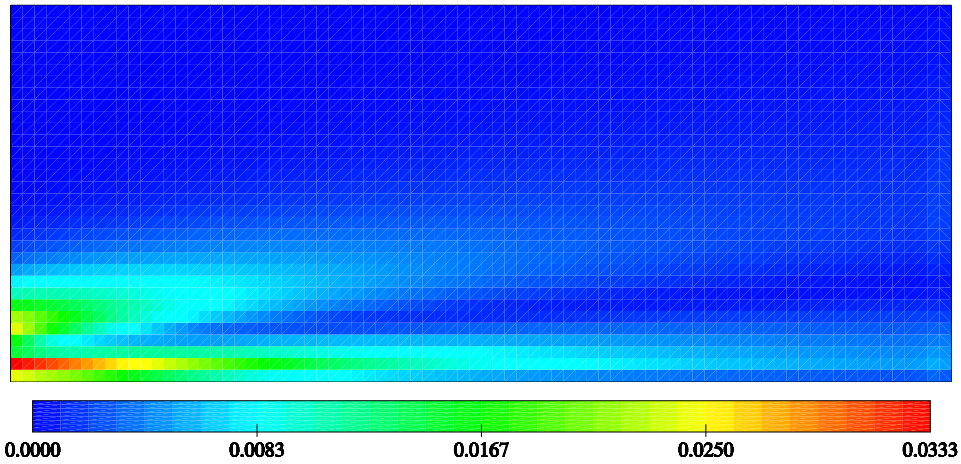


Figure 4.6: Pressure isobars for the jet case [m^2/s^2] (80 x 32 mesh)



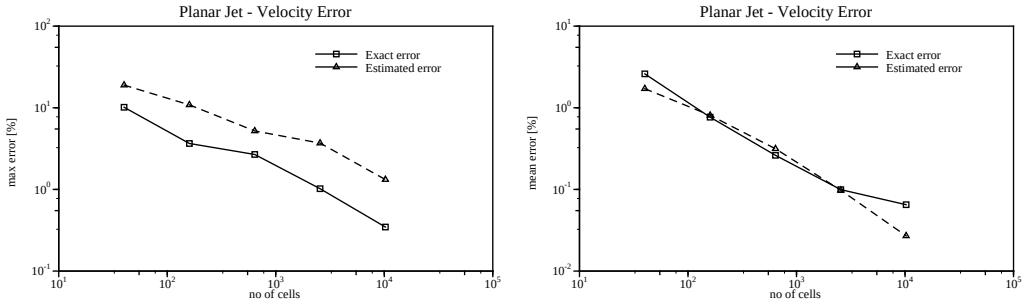
(a) Exact error



(b) Estimated error

Figure 4.7: Velocity-error field for the jet case [m/s] (80 x 32 mesh)

Fig. 4.6, which is assumed zero in the analytical solution. The errors are large in the region where the shear is dominant and are a result of diffusion-term discretisation. The behaviour of both maximum and mean error with uniform mesh refinement



(a) Maximum error (b) Mean error

Figure 4.8: Variation of errors with uniform mesh refinement for the jet case ($|\mathbf{U}_{norm}| = 2.474m/s$)

is shown in Figs. 4.8(a) and 4.8(b), respectively. The errors are normalised by the maximum value of the velocity magnitude $|\mathbf{U}| = 2.474m/s$ and given as percentages. Estimation of the mean-velocity error is considered accurate on the first four meshes and becomes under-estimated on the finest mesh due to the modelling error in the analytical solution. The maximum error, Fig. 4.8(a), is over-estimated. This can be explained by looking into the truncation error for the diffusion term, Eqn. (3.104), which is dependent on the third velocity gradient on uniform meshes, while the estimator is dependent on the second gradient, Fig. 4.45. Unfortunately, the estimator may therefore predict larger errors in the regions where the discretisation practice can produce a more accurate solution.

4.5.2 Creeping Stagnation Flow

This test case was chosen for the purpose of testing the error estimator on a problem where diffusion is the main transport mechanism, *i.e.* for $Re << 1$ and is modelled with the Stokes equations for creeping flows. It is a 2D problem where the fluid impinges vertically into a flat wall forming a stagnation point, Figs. 4.9 and 4.10(a).

The analytical solution of the Stokes equations for this problem is [112]:

$$U_x = -x\frac{1}{\nu}\frac{\partial p}{\partial y}y, \tag{4.55}$$

$$U_y = \frac{1}{\nu}\frac{\partial p}{\partial y}\frac{y^2}{2}, \tag{4.56}$$

where $\frac{\partial p}{\partial y}$ is uniform and here set to:

$$\frac{\partial p}{\partial y} = 1\,[m/s^2] \tag{4.57}$$

and its choice determines the velocities. The pressure gradient in the x-direction is zero. The kinematic viscosity used in the calculations is:

$$\nu = 1\,[m^2/s]. \tag{4.58}$$

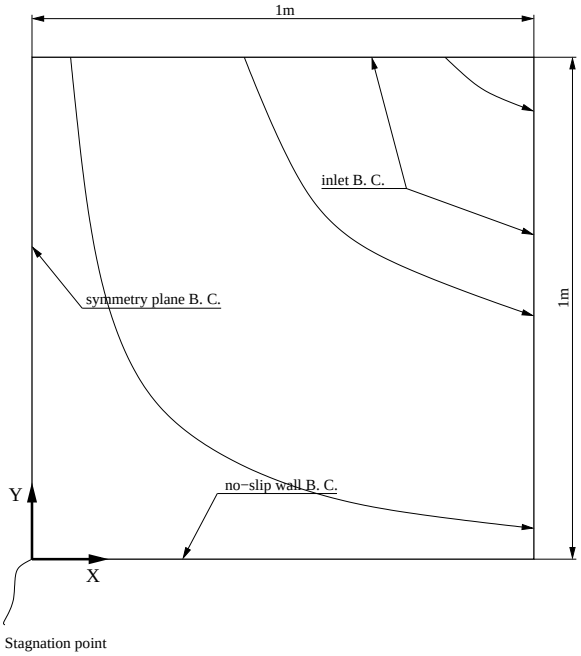
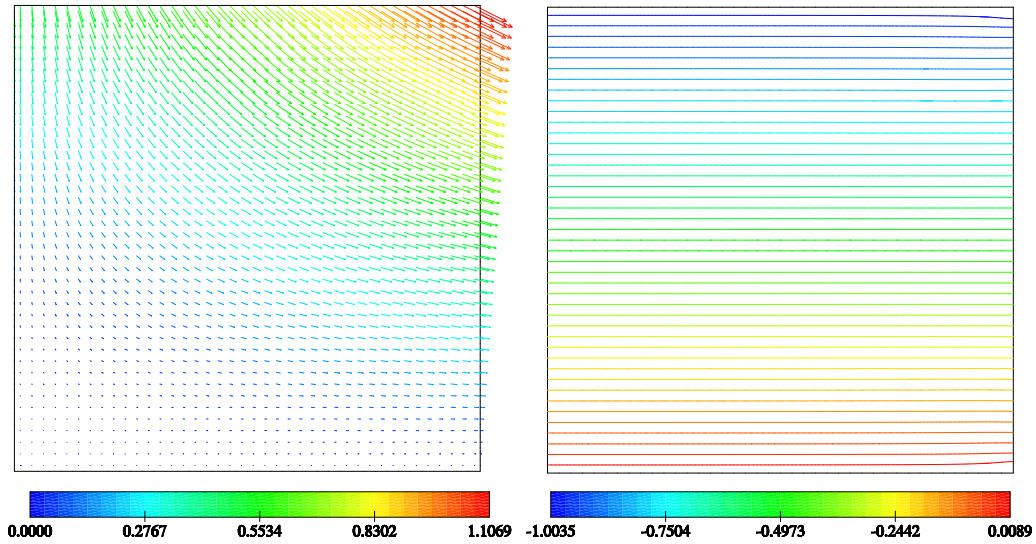


Figure 4.9: Solution domain and boundary conditions for the stagnation flow

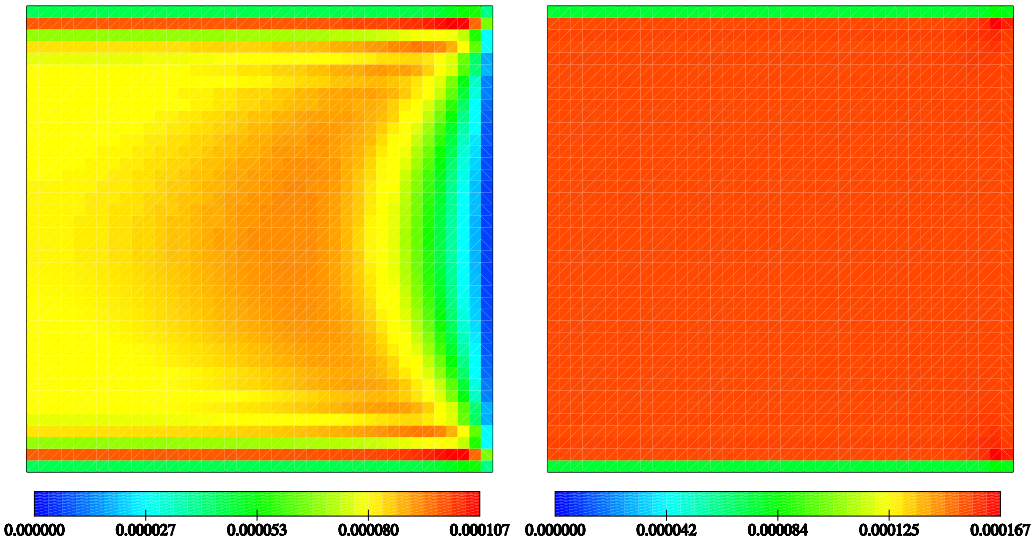
Calculations were performed on a square-shaped domain with a lower-left corner located at the origin of the coordinate system and the upper-right corner positioned at (1m, 1m), as shown in Fig. 4.9. The initial mesh consisted of twenty-five cells, five in each direction, while the three subsequent ones were obtained by halving the cell size in both directions.

The boundary conditions used for calculations are shown in Fig. 4.9. The values of velocity on top and right boundaries are set to match the analytical solution.



(a) Velocity field (m/s) (b) Pressure isobars (m^2/s^2)

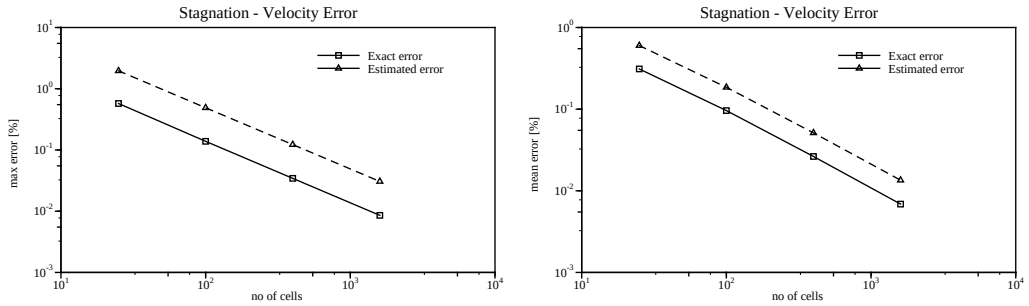
Figure 4.10: Velocity and pressure for the stagnation flow (40 x 40 mesh)



(a) Exact error (b) Estimated error

Figure 4.11: Velocity-error fields for the stagnation flow [m/s] (40 x 40 mesh)

The exact-velocity error is small and of second-order with mesh refinement, Figs. 4.11(a) and 4.12(a). The small error is a consequence of the fact that the second velocity gradient is constant which can be resolved exactly by the discretisation practice, Section 3.6.2, so the main source of error in the solution is the evaluation of the velocity-surface-normal gradients at the fixed value boundaries $(\mathbf{S} \cdot \nabla \mathbf{U})_b$, Eqn. (3.79). In contrast, the FREE is sensitive to the second gradient of velocity, Eqn. (4.45), which results in over-estimation of the error. This is the reason why the estimated velocity-error distribution, shown in Fig. 4.11, is in poor agreement with the exact one, Fig. 4.11(a).



(a) Maximum error (b) Mean error

Figure 4.12: Variation of errors with uniform mesh refinement for the stagnation flow ($|\mathbf{U}_{norm}| = 1.107 m/s$)

The behaviour of the estimator is of second-order, as shown in Figs. 4.12(a) and 4.12(b) where the maximum velocity magnitude used for normalisation is $|\mathbf{U}| = 1.107 m/s$. The estimator over-predicts both maximum and mean-velocity error because of the reasons given above.

4.5.3 Convection Transport of Heat with a Distributed Heat Source

This case was designed for the purpose of testing the FREE on a convection-dominated problem. It involves a temperature field which changes due to convection and heat source/sink, see Fig. 4.14(a). The problem is governed by the equation:

$$\nabla \cdot (\rho \mathbf{U}\, T) = Su(x, y).$$

where $Su(x, y)$ is given below. The fluid density is constant and equal to $\rho = 1kg/m^3$. The velocity field $\mathbf{U}$ is uniform and at 45^o to the mesh, Fig. 4.13.

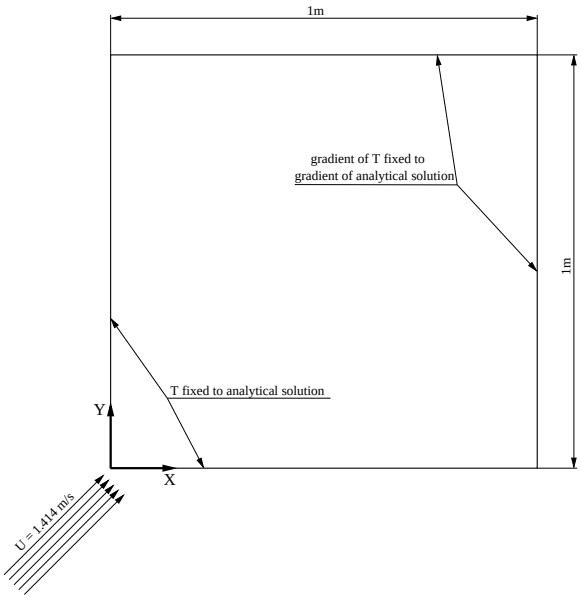


Figure 4.13: Solution domain and boundary conditions for the convection transport case

The solution is a parabolic temperature profile:

$$T(x, y) = Ax(1 - x)y(1 - y) \tag{4.59}$$

and the source term which generates the above profile is:

$$Su(x, y) = A\Big[U_x(y(1 - y)(1 - 2x)) + U_y(x(1 - x)(1 - 2y))\Big], \tag{4.60}$$

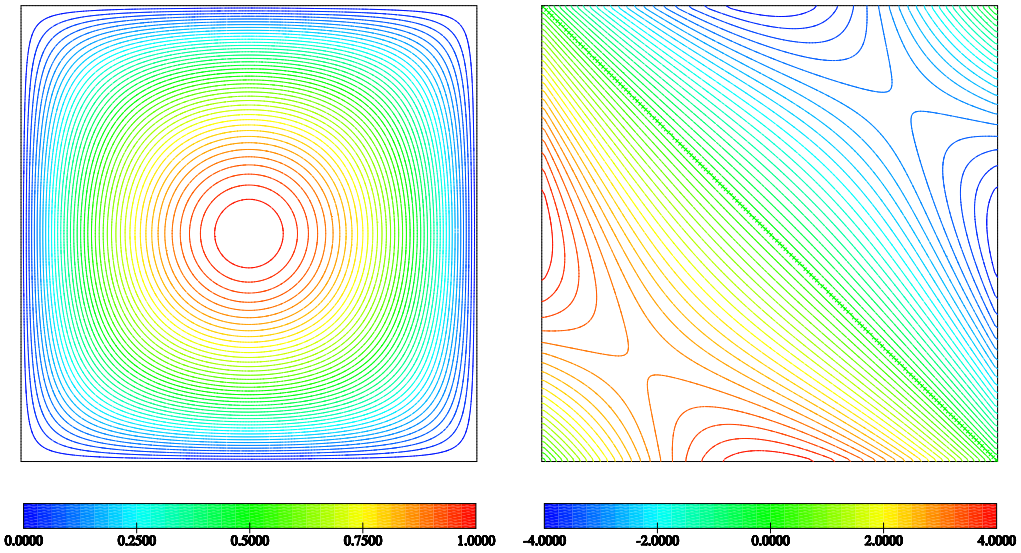
where U_x and U_y are the x and y velocity components, respectively, both 1 m/s. The constant A is set to:

$$A = 16\,[K/m^4]. \tag{4.61}$$

The exact solution and the corresponding source can be seen in Figs. 4.14(a) and 4.14(b), respectively.

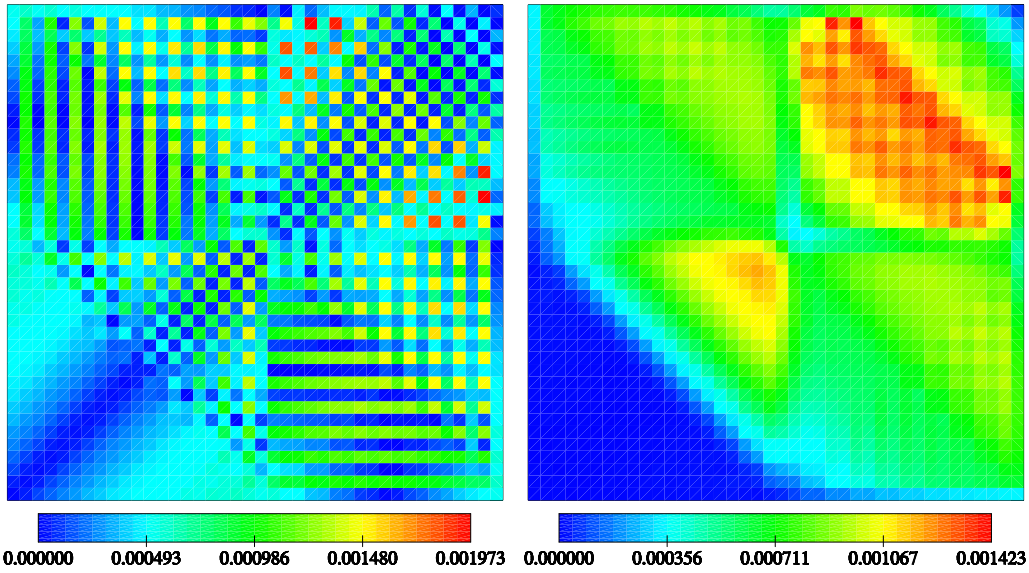
Fig. 4.13 shows the numerical setup with the boundary conditions imposed on the temperature field. The convection differencing scheme used for the calculations is Gamma [69] with the factor $\beta_m = 0.5$.

Four meshes were considered for the analysis. The initial mesh had five cells in both x and y directions and the subsequent ones were generated by doubling the resolution in both directions.



(a) Temperature field (^oC) (b) Source field ($^oC/s$)

Figure 4.14: Temperature and source fields for the convection transport case (40 x 40 mesh)



(a) Exact error (b) Estimated error

Figure 4.15: Error fields for the convection transport case [^oC] (40 x 40 mesh)

The exact error distribution, Fig. 4.15(a), is a consequence of the fact that the Gamma interpolation scheme becomes Central Differencing when the boundedness criterion is satisfied, Section 3.3.1. For the CD scheme a central coefficient in the computational molecule is equal to zero on uniform meshes shown in Fig. 4.16, thus:

$$T_N - T_S + T_E - T_W = Su\frac{2V_P}{F}$$

where N, S, E and W are the neighbouring cells to the cell of interest whose temperature is not present in the equation above.

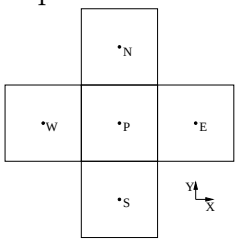
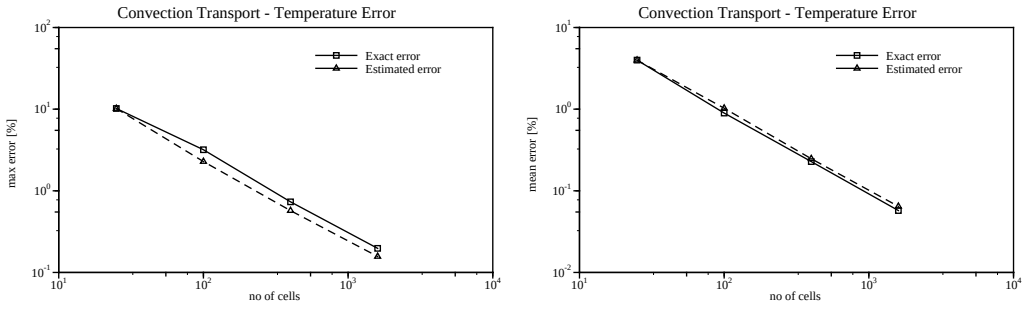


Figure 4.16: Uniform mesh

Because of this, the cells with no boundary faces exchange information with their second neighbours which results in a chequerboard-error field. Rows and columns with small errors, Fig. 4.15(a), are initiated at the boundary due to the known boundary information and they influence the second neighbours while their first neighbours are influenced by the opposite boundary.



(a) Maximum error (b) Mean error

Figure 4.17: Variation of errors with uniform mesh refinement for the convection transport case ($T_{norm} = 1^oC$)

A comparison of the exact and estimated-error fields, Figs. 4.15(a) and 4.15(b), shows moderate agreement. The error in the lower-left quadrant is not detected by the estimator because it is generated by the source-term discretisation which cannot

be detected by the estimator. This has caused under-estimation of the maximum error, Fig. 4.17(a). The estimated mean error, Fig. 4.17(b), is in very good agreement with the exact mean error curve. It is noteworthy that the rate of reduction of mean and maximum errors is of second-order on fine meshes when the Gamma interpolation scheme becomes CD.

4.6 Summary and Conclusions

This chapter has presented a new method for error estimation on mesh faces, namely the FREE, which has resulted from an analysis of discretisation errors on the mesh faces. The FREE is of better accuracy on convection-dominated cases than on diffusion-dominated ones for which it over-estimates the error. The over-estimation is caused by the fact that the FV approximation of the diffusion term may be capable of producing exact solutions even when the variation of the solution is of higher order than linear, as shown in Section 3.6.2, while the estimated error is not zero when the above is present. This undesirable behaviour could be avoided by modifying the face residual for the diffusion term such that its truncation error becomes consistent with the truncation error for the diffusion term.

The estimator will be further tested in Chapters 5 and 6 in conjunction with an adaptive-refinement technique. In addition, the estimator will be tested in conjunction with an adaptive-mesh-generation procedure presented in Chapter 7.

Chapter 5

Mesh Adaptation

5.1 Introduction

Problems of fluid dynamics are rich in phenomena characterised by different length scales which have an impact on the discretisation error. These scales require appropriate mesh resolution to achieve a solution of the desired accuracy. The estimated error distribution can be used to control the distribution of cells in the mesh in order to achieve the requested accuracy with the smallest cost. This chapter will present a mesh-adaptation technique developed during this study which uses the FREE.

The remainder of this chapter is organised as follows. A survey of existing methods is given in Section 5.2. The method developed during this study will be presented in Section 5.3. Examples can be found in Section 5.4. Concluding remarks are given in Section 5.5.

5.2 Literature Survey

Mesh-adaptation strategies can be grouped into four main groups, depending on the type of changes imposed on the mesh and/or the discretisation practice, thus:

- h**-refinement** is a strategy where additional computational cells are inserted in regions of high error. Cells can also be removed from regions of small error using a coarsening procedure. The number of cells may increase or decrease during the calculation process as a consequence of additions and removals. The topology of the mesh changes during the adaptation process to achieve a mesh with optimum cell distribution. Examples can be found in Vilsmeier

and Hänel [151], Becker *et al.* [21], Chen *et al.* [36], Ramakrishnan *et al.* [129], Vijayan *et al.* [150], Erlebacher *et al.* [45], Chang *et al.* [35], Peraire *et al.* [119], Mavriplis [98], Almeida *et al.* [11], Apel *et al.* [14] and many more.

- **r-refinement** keeps the number of cells and their topology constant but changes their distribution in order to minimise the error. The fact that the topology remains constant during the process makes this approach attractive for block-structured meshes. However with this approach there is no guarantee that the required accuracy can be achieved with the specified number of cells. The approach may also cause severe distortion of the mesh and convergence problems. Examples of this approach can be found in Ait-Ali-Yahia *et al.* [10], Gnoffo [55], Nakahashi *et al.* [104], Oden *et al.* [107], Dwyer *et al.* [43], Tattersall *et al.* [142] and others.

- **p-refinement** refers to the group of methods which change the order of the discretisation practice according to the error distribution. This approach is particularly suitable for Finite Element calculations as the nature of the discretisation practice allows easy implementation of higher-order shape functions. The p-refinement method is not suitable for non-smooth solutions as higher-order schemes are more prone to spurious oscillations in regions of steep gradients than their lower order counterparts. Calculations using p-refinement have been presented by Zienkiewicz *et al.* [163], Demkowicz *et al.* [38], *etc.*

- **Composite methods.** These methods combine two or more of the previously mentioned methods. The most popular ones are $h-r$ methods Tam *et al.* [139] and $h-p$ methods [164].

Reviews of mesh-adaptation techniques can be found in [16, 57].

In the framework of h-refinement methods, different ways of inserting and removing cells have been suggested. They mainly differ regarding the type of mesh and problems they are designed to work on. Early mesh-refinement techniques in the FVM were used in conjuction with multigrid technique for solving systems of algebraic equation *i.e.* [29, 146]. In [29] Brandt presented a technique to optimise accuracy and minimise the computational work by changing the spatial discretisation and the order of approximation locally in the domain. Berger *et al.* [22–24] presented a technique based on inserting "patches" of fine mesh covering the regions

of the domain where finer resolution is needed. Error estimation was performed by using Richardson Extrapolation. In order to simplify implementation and apply a safety margin, at the expense of error equidistribution, "patches" of refined mesh were uniform, orthogonal and structured. Transfer of data between the overlapping refined grids is facilitated through boundary conditions. Thompson *et al.* in [146] used a similar methodology where refinement was performed by using overlapping meshes of increasing fineness. The adaptation criterion is based on the truncation error and systems of algebraic equations are solved using a multigrid technique.

Muzaferija [102, 103] has presented a method which works on polyhedral elements. It is based on splitting of hexahedral cells into eight new cells and updating a list holding information about the cells added into the domain. Coarsening is performed by merging cells which were previously refined. The interaction between the refined parts of the mesh and its surroundings is facilitated by using split-hexahedron cells. A split-hexahedron has a shape of a hexahedron but has more than six faces. The method does not require any special treatment at the boundary of the refined regions but requires appropriate mesh addressing.

Jasak and Gosman [69, 72, 73] have presented a method which enables directional refinement and coarsening by splitting and merging hexahedral and split-hexahedral cells. The split-hexahedron cell used in this work can have only one face shared by two neighbouring cells thus making the procedure inefficient for cases where highly localised refinement is needed, because the procedure refines more cells than needed in order to satisfy the criterion that every cell in the mesh is either a hexahedron or a split-hexahedron. Coarsening is performed by merging two cells together and taking into account that the shape of the newly created cell is either a hexahedron or a split-hexahedron. The decision about directions of refinement is based on the solution gradient and directional stretching of the cell. This has a drawback that the directional splitting is dependent on the mesh stretching which in some cases may cause refinement and coarsening being performed in an inappropriate direction. This may result in the increase of the disretisation error causing local over-refinements in such regions.

An $h-r$ method using directional refinement and coarsening has been presented by Habashi *et al.* [9, 10, 39, 40, 56]. Directional refinement is achieved by driving the adaptation by estimating the errors on the computational mesh edges. In [10, 56] the authors discuss the principles of error equidistribution based on the concept

that a mesh adapted to resolve the flow features can be viewed as a uniform mesh in the error space. The r-refinement procedure aimed at equidistributing the error on edges has been discussed in [9]. In [39] this method was applied to several cases, including turbulent ones, and was tested with two different solvers, a Finite Element one and a Finite Volume one, which showed that it yields similar meshes and results for both solvers. The method is capable of producing accurate solutions with saving on the number of cells but requires many adaptation loops, which is not desired in the engineering environment.

There has been a substantial effort recently to apply mesh-adaptation methodology to turbulent flows. Jasak and Gosman [73] have applied the methodology from [69, 72] to a flow over a 2D hill and to a flow over a 3D swept step. The method did not reduce the error monotonically because the quality of the mesh was not preserved in the important regions. Muzaferija and Gosman [102, 103] have tested their methodology on turbulent cases of engineering interest. As the method refines each hexahedral cell into eight new cells, it produces over-refined meshes for features such as boundary layers and shear layers which are highly directional.

In [39] Habashi *et al.* have applied their methodology to a turbulent flow around a 2D car and to a turbulent flow around NACA 0012 profile. The methodology improved predictions of pressure coefficients and clustered most of cells in the boundary layer region where the high gradients reside.

Pelletier *et al.* [63, 65, 66, 116–118, 147] developed a mesh-refinement technique for turbulent flows simulated with two-equation turbulence models. The estimation of error is performed using the method developed by Zienkiewicz and Zhu [164] while the mesh refinement is based on methodology developed by Peraire *et al.* [119]. The method does not produce anisotropic meshes aligned with important flow features, which increases the number of cells required to achieve accurate solutions. In order to reduce errors for turbulence quantities they are solved in logarithmic form which makes their variation smooth and less computationally demanding.

Although adaptive methods have evolved during the past few decades, none can generate high-quality anisotropic meshes, which is required to achieve high accuracy with the smallest possible number of cells, and achieve it in a small number of adaptation cycles. The refinement technique developed during this study is a h-refinement, because h-refinement provides best control of the mesh resolution and mesh quality. It is also considered to be the most suitable for problems of engineering

interest where the geometries are complex.

5.3 Adaptation Procedure

The procedure operates on hexahedral meshes and it consists of the following steps:

1. Solve the problem on the initial mesh.

2. Estimate the discretisation-error distribution. Terminate the calculation if the error on every face is below the required level.

3. Determine faces for which the estimated error is above the limit and mark them. Cells sharing marked faces are marked for refinement parallel to those faces to reduce distances between the nodes which mainly contribute to the error. This rule has some exceptions and will be treated in more detail in Section 5.3.1. The coarsening procedure is not implemented because the author is not certain how to perform coarsening based on information provided by the FREE such that it does not have an adverse effect on mesh quality and error reduction.

4. In order to ensure the correct boundary description, newly created boundary vertices may have to be moved from the position where they were located by the refinement procedure to the closest point at the boundary surface.

5. Map the solution from the previous mesh and use it as an initial guess.

6. Repeat from step 1 until the error is reduced below the required upper limit.

5.3.1 Selection of Cells and Mesh Refinement

The selection of cells is performed by using the information about the error on the faces provided by the FREE. A hexahedral cell can be split parallel to a face without making any changes in other directions. This directional refinement results in reduced computational cost as the number of the cells in the adapted mesh is smaller than in the case when the cells are split in all three directions.

The selection of cells for refinement also has an impact on the quality of the adapted mesh, and it has to be performed in such a way that it does not have

an adverse effect on accuracy. The accuracy may be reduced at the interface between the refined cells and the rest of the mesh where split-hexahedron cells appear, Fig. 5.1. A split-hexahedron can have one or more split faces which obey the property that each split face can be shared with two or four neighbouring cells, Fig. 5.1. In Section 3.6 it was shown that split-hexahedra produce higher errors than the

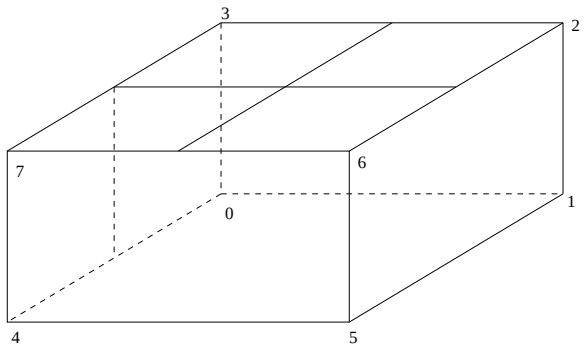


Figure 5.1: A split-hexahedron cell with left face split in one direction shared with two cells. The top face is cross-split and shared with four cells

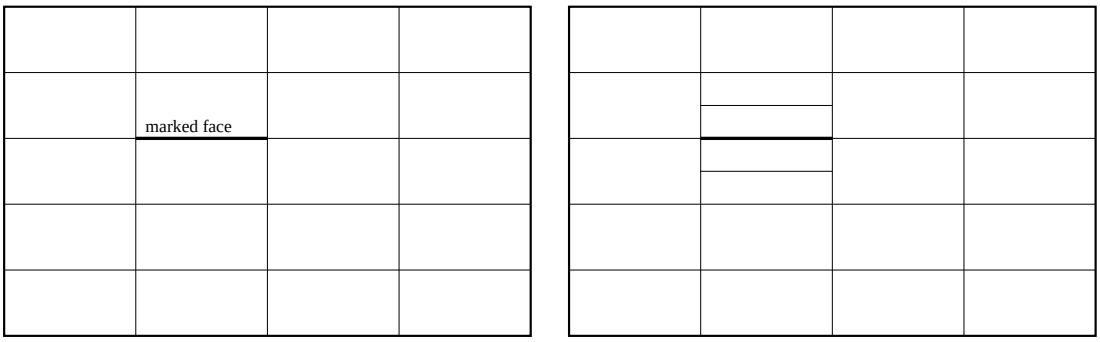
hexahedra for the same cell size when higher-order gradients are non-zero. This happens due to increased distance between the neighbouring nodes of the split face, Eqn. (3.82), skewness and non-orthogonality which reduce the quality of the mesh. It may therefore happen that the solution on the refined mesh has higher errors than on its parent mesh in the regions where split-hexahedra are present. To prevent this from happening, the selection of cells for refinement is performed in two steps:

1. Compare the estimated error on the face e_f with the specified upper limit E. If:

$$e_f > E, \tag{5.1}$$

the face is marked and the adjacent cells are marked for refinement in the direction parallel to that face, Fig. 5.2. This reduces the distance between the neighbouring nodes which are the main contributors to the error on that face. Therefore, the directional information about the error is directly used to drive the adaptation.

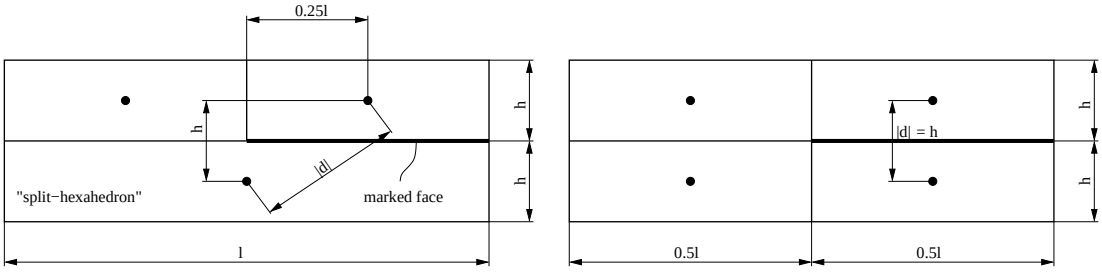
This rule has exceptions. If the marked face is a split face of a split-hexahedron, only the split-hexahedron cell is split such that the face is not a split face any more, in order to minimise skewness, non-orthogonality and the distance



Marked face Refined mesh

Figure 5.2: Directional splitting of cells

between the neighbouring nodes, as shown in Fig. 5.3.



Before refinement After refinement

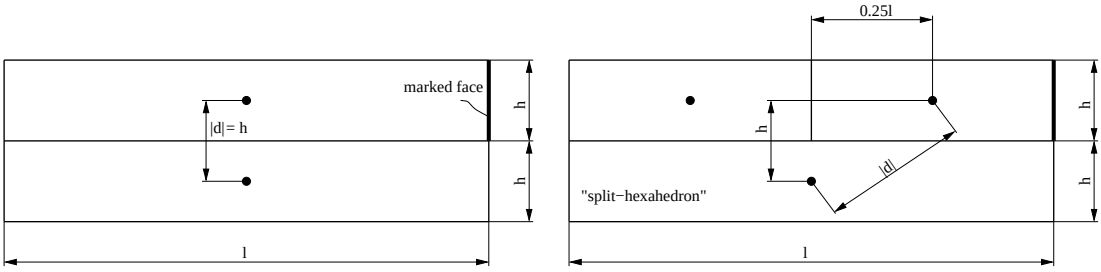
Figure 5.3: Refinement of split-hexahedron cells

If the flow under consideration is turbulent and a high-Re model with wall-functions is used, it is necessary to ensure that the wall-function is still valid on the refined mesh by not allowing refinement of the near-wall cells parallel to the wall if $Y^+ < 50$.

2. The second step is aimed to make the estimated error reduce monotonically with mesh refinement and to minimise the number of cycles required to achieve the desired accuracy. Fig. 5.4 shows that the distance between neighbouring nodes increases at the boundary between the refined regions and the rest of the mesh. The ratio r between node distances before and after the face is split is:

$$r = \sqrt{\frac{h^2 + \frac{l^2}{16}}{h^2}}, \tag{5.2}$$

where h and l are the cell height and the cell length, respectively, see Fig. 5.4. This ratio may become large for high-aspect-ratio cells where $l >> h$. Eqs. (4.42)



Before refinement After refinement

Figure 5.4: Node distances at a split face

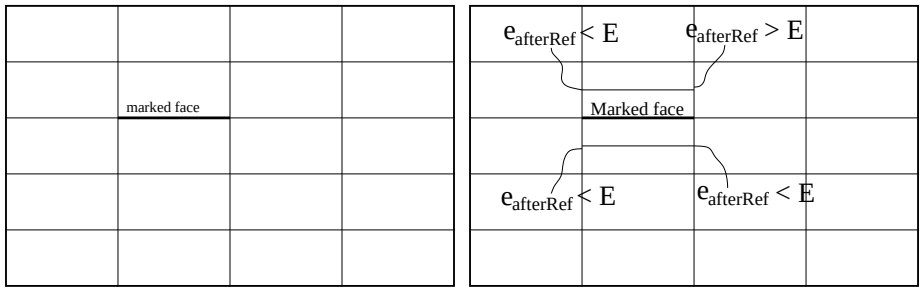
and (4.45) show that the FREE is dependent on the square of the distance between neighbouring nodes. Taking that into account, it is possible to approximate the estimated solution error on the refined mesh before the refined mesh and the solution on that mesh are known and before the error estimation on the refined mesh is performed. The approximated estimated error on the faces of the refined mesh created from a face of the parent mesh which should become a split face of a split-hexahedron is:

$$e_{afterRef} \approx r^2 \, e_f. \tag{5.3}$$

where e_f is the estimated error in the solution on the parent mesh and r is defined in Eqn. (5.2). $e_{afterRef}$ is the approximated error on the refined mesh; which is not a true estimated error because the refined mesh is not known at this stage. It is an approximation of the estimated error which is used to detect regions where split-hexahedra may have a large influence on the accuracy. If:

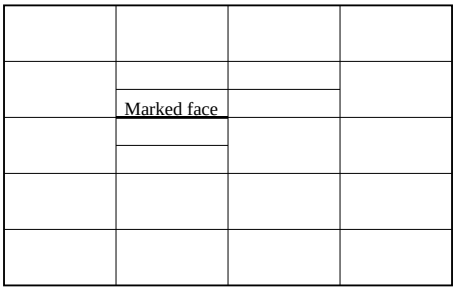
$$e_{afterRef} < E, \tag{5.4}$$

then this is not considered a problematic region because it is unlikely to have the estimated error in the solution on the refined mesh above the upper limit E. If Eqn. (5.4) is not true, it is very likely to have the estimated error above the required upper limit once the refined mesh and the solution on that mesh are known. To prevent this from happening, the cell which would become a split-hexahedron is marked for refinement such that it does not become a split-hexahedron. This process is shown in Fig. 5.5, and it is repeated until Eqn. (5.4) is satisfied for every face which may become a split face of the split-hexahedron. By doing this it is possible to reduce the number of cycles



(a) Faces selected after step 1.

(b) Checking of errors on faces which will become split faces of split-hexahedra



(c) Final selection of cells for refinement

Figure 5.5: Additional splitting of cells

required to reach the required error level because the regions where split-hexahedra may cause problems are treated in the current cycle instead of one cycle after another.

The equations of continuum mechanics are coupled and non-linear, so the errors in a given variable field can have an impact on the accuracy of the solutions of the other equations. The errors are therefore estimated for every field and the selection of cells for refinement is performed for all error fields except the pressure. The error in the pressure field is closely related to the error in the velocity field.

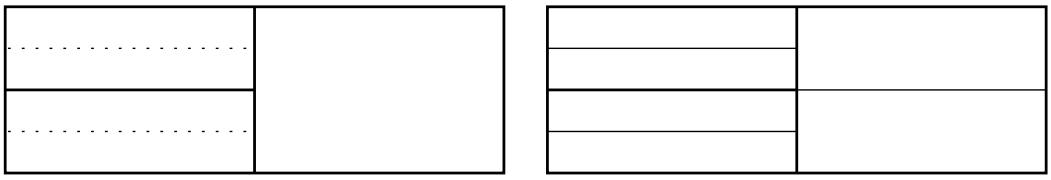
Mesh-Refinement Procedure

So far cells have been marked for refinement in the directions required by the error estimator. Before the refined mesh is generated it is necessary to check if it will consist of cells types which are either a hexahedron or a split-hexahedron, defined above and shown in Fig. 5.1. In order to satisfy this requirement it is sometimes

necessary to split some additional cells or add some additional directions to cells already marked for refinement.

The combinations which are not allowed and their remedies are:

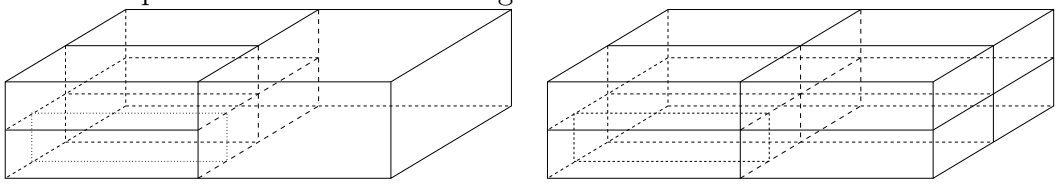
- If the two neighbours which share a split face of a split-hexahedron need to be split such that the split face changes, the split-hexahedron is split into two new cells to ensure that the refined mesh consists only of hexahedra and split-hexahedra, Fig. 5.6.



Selected cells Refined mesh

Figure 5.6: Consistency over a split face in 2D (dotted lines represent the selected refinement)

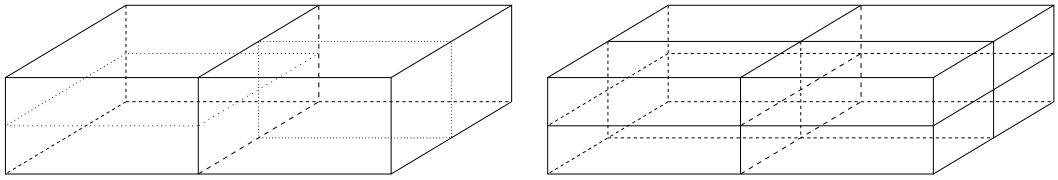
- In any of the four neighbouring cells of a cross-split face of a split-hexahedron should be split such that the cross-split face changes, the split-hexahedron should be split into four cells as in Fig. 5.7.



Selected cells Refined mesh

Figure 5.7: Consistency over a cross-split face in 3D (dotted lines represent the selected refinement)

- It may also happen that two neighbouring cells should be split such that their shared face is cross-split. In such case both cells are split in two directions, Fig. 5.8.



Selected cells Refined mesh

Figure 5.8: Treatment of incompatible cell splitting directions in 3D (dotted lines represent the selected refinement)

This final step is needed to ensure that the refined mesh can be refined again by using the same rules. Otherwise the mesh after every cycle of refinement might require additional rules to treat new combinations and new cell types. It also serves the purpose of preserving mesh quality by not allowing abrupt changes between the refined and coarse regions.

5.3.2 Solution Mapping Between Meshes

It is advisable to re-use the solution on the previous mesh by mapping it to the new mesh to minimise the computational time during each cycle and speed up the solution process.

For properties calculated in the centre of the computational cell *i.e.* velocity, pressure, mapping is performed using the method developed by Jasak [69] which consists of the following steps:

1. For every node N in the refined mesh find a cell P in the old mesh which encloses it.

2. Find the distance vector between the node N and the centre of cell P using:

$$\mathbf{x}_{dist} = \mathbf{x}_N - \mathbf{x}_P \tag{5.5}$$

3. Calculate the ϕ on the refined mesh, thus:

$$\phi_N = \phi_P + \mathbf{x}_{dist} \cdot \nabla(\phi)_P \tag{5.6}$$

   where ϕ_P is a nodal value on the old mesh and ϕ_N is a nodal value on the refined mesh.

Boundary conditions and their data are known before the analysis and are used to set values and/or gradients at the boundaries for every cycle.

5.4 Examples

The procedure presented above has been tested on the set of cases described in Section 4.5 with an additional case involving convection and diffusion of a rectangular temperature profile [152].

The performance of the adaptive-mesh-refinement procedure is examined by monitoring the maximum and average-error reduction with mesh refinement. It is also compared with the error obtained by uniform refinement.

5.4.1 Planar Jet

This test case was described in Section 4.5.1. The calculation was started on a (10 x 4) initial mesh, Fig. 4.4. The maximum allowed velocity error was set to:

$$E = 1\,\% \tag{5.7}$$

of the maximum velocity magnitude $|\mathbf{U}_{norm}| = 2.474 m/s$.

A solution of the desired accuracy was achieved after six cycles of adaptive refinement. Fig. 5.9 shows the resulting mesh after six cycles of refinement. The refinement is concentrated in the region of high gradients where the jet enters the domain, Fig. 4.5. Most of refinement is in the cross-stream direction which is the direction of high velocity variation due to diffusion.

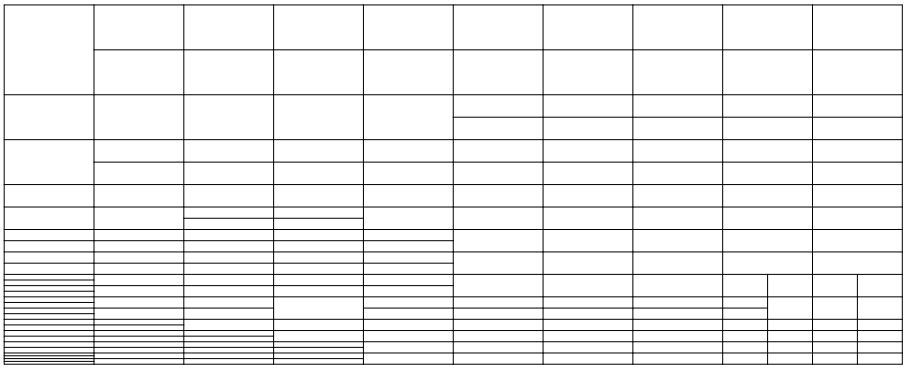
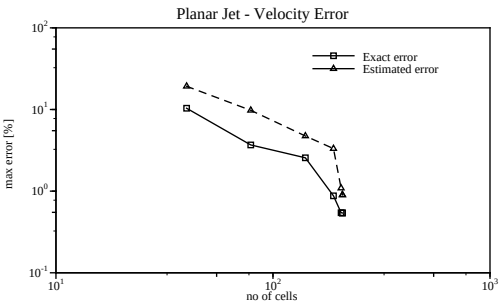
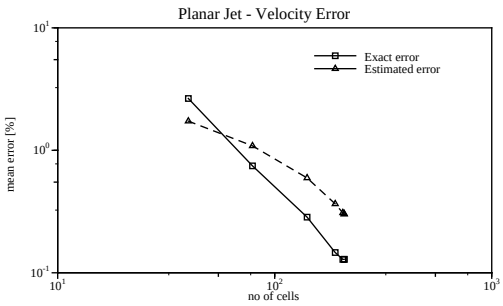


Figure 5.9: Mesh after 6 cycles of refinement for the jet case (209 cells)

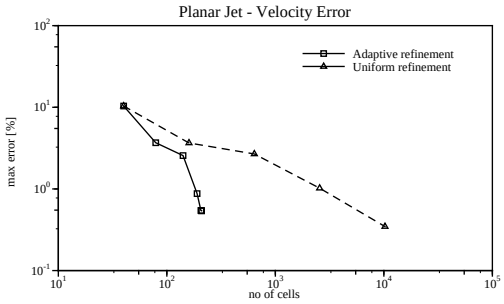
The reduction of error is monotonic for maximum and mean error, as can be seen in Figs. 5.10(a) and 5.10(b). A drawback is the over-estimated error which produces



(a) Maximum error



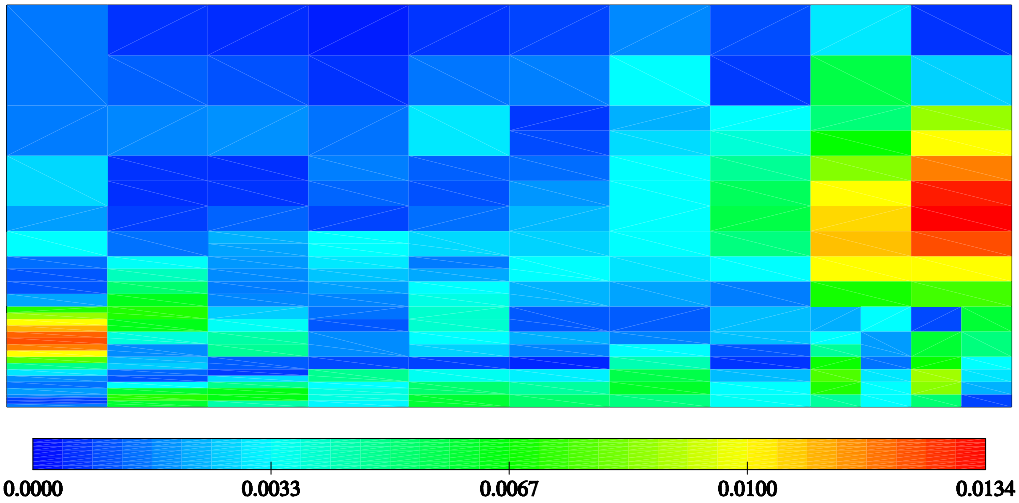
(b) Mean error



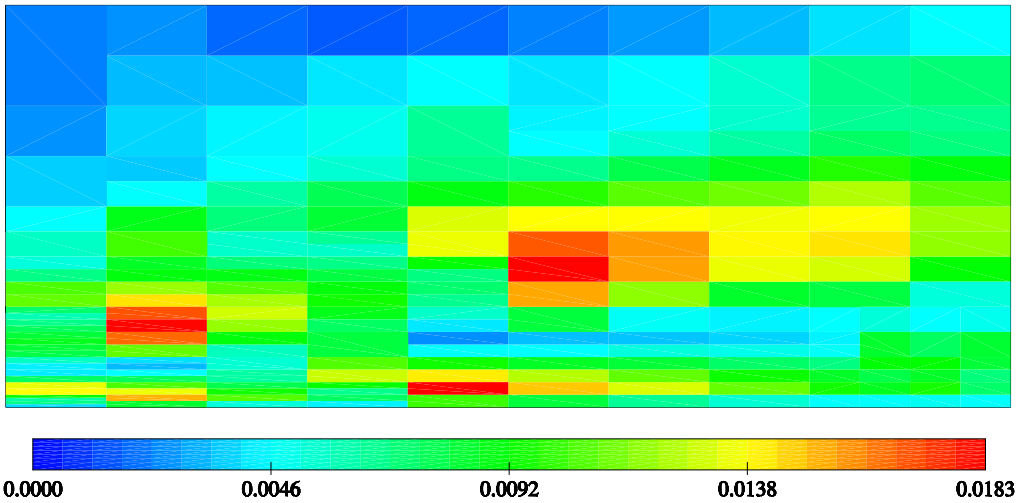
(c) A comparison of the maximum-velocity error for uniform and adaptive refinement

Figure 5.10: Variation of velocity errors with adaptive refinement for the jet case ($|\mathbf{U}_{norm}| = 2.474m/s$)

an over-refined mesh. Even though the error is over-estimated, the procedure is still capable of producing an accurate solution with significant saving in the number cells. Fig. 5.10(c) shows the exact maximum-velocity error for uniform and adaptive refinement, where the solution on the adapted mesh (209 cells) has a comparable maximum error with the one on the uniform mesh (10240 cells), but the adapted mesh has 45 times fewer cells than the uniform one.



(a) Exact error



(b) Estimated error (given as average of face errors)

Figure 5.11: Velocity errors after 6 cycles of refinement for the jet case [m/s] (209 cells)

Figs. 5.11(a) and 5.11(b) show the exact and the estimated-error fields after 6

cycles of refinement. The error is over-estimated in the bulk of the domain.

The distribution of estimated errors on faces is shown in Table 5.1. The columns of the table show the percentage of mesh faces with the estimated error in the given range when compared with the maximum-estimated error. For example, in the starting mesh 73.4% of faces had the estimated error smaller than 10% of the maximum-estimated error on that mesh. The distribution of error improves with

	% of maximum estimated error on a given mesh									
Mesh	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
Starting	73.4	8.5	6.4	6.38	2.12	0	1.06	1.06	0	1.06
Final	25.5	17.3	15.4	14.9	11.5	6.06	4.11	1.73	1.94	1.51

Table 5.1: Estimated errors on the faces of the final mesh with 209 cells (given as percentage of the maximum estimated error on that mesh)

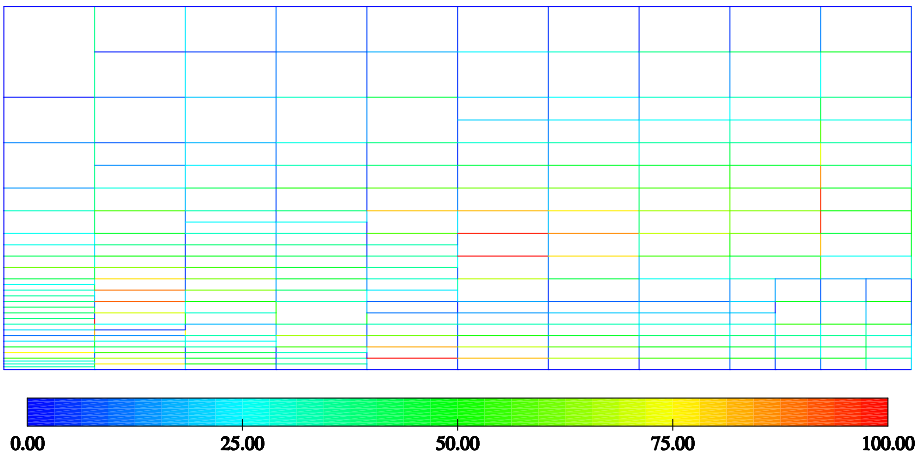


Figure 5.12: Estimated errors on the faces of the final mesh with 209 cells (given as percentage of the maximum-estimated error on that mesh)

mesh refinement but most faces in the final mesh still have low error, Fig. 5.12. These are:

- Faces located in the region of low velocity near the top-left corner. The low error there suggests that the starting mesh is too fine there.
- Boundary faces. This happens because most boundaries are fixed value boundaries on which the error is zero.

- Faces perpendicular to the flow. The error is low there because the velocity field does not exhibit high gradients in the flow direction. The number of these faces does not reduce relative to the total number of faces because their number increases even with directional refinement. These faces can only be eliminated by using some coarsening procedure.

5.4.2 Stokes Stagnation Flow

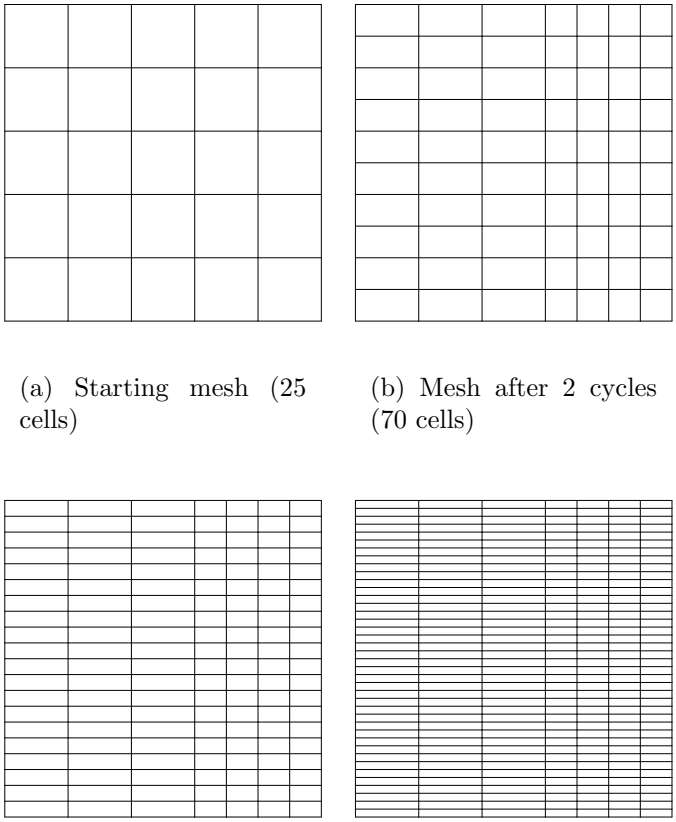
The starting mesh for this case was a uniform (5 x 5) cells mesh, Fig. 5.13(a). The description of the problem and its numerical setup is given in Section 4.5.2. The target maximum-velocity error in the solution was set to:

$$E = 0.1\,\% \tag{5.8}$$

of the maximum velocity magnitude $|\mathbf{U}_{norm}| = 1.107 m/s$. Two calculations were performed in order to determine if the second step of the procedure for selecting cells, described in Section 5.3.1, makes the error reduce monotonically with minimised number of adaptation cycles. The calculation using the complete methodology is presented first, followed by the results of the calculation for the adaptation procedure which does not use that step.

The maximum error was reduced below the desired level after four adaptation cycles when the complete methodology was used. Meshes after every cycle are presented in Figs. 5.13(a), 5.13(b), 5.13(c) and 5.13(d) where the main direction of refinement is in the vertical direction in which the estimated-velocity error is high because the second derivative of the y velocity component in the y direction is not zero. The exact error, shown in Fig. 5.14(a), is almost equidistributed in the bulk of the domain and is small near the bottom, top and right boundaries where the velocity is fixed.

The agreement between the exact and the estimated error in the final adapted solution, shown in Figs. 5.14(a) and 5.14(b), is poor and this undesired behaviour is caused by the estimator and the averaging procedure, Eqn. (4.48), used for plotting the estimated-error field. The estimated error is smaller on the right-hand side of the domain compared to the left-hand side because the weighting factors used for faces with high error, see Eqn. (4.48), are smaller due to smaller area of those faces compared to the sum of areas of all cell faces, as shown in Fig. 5.16.



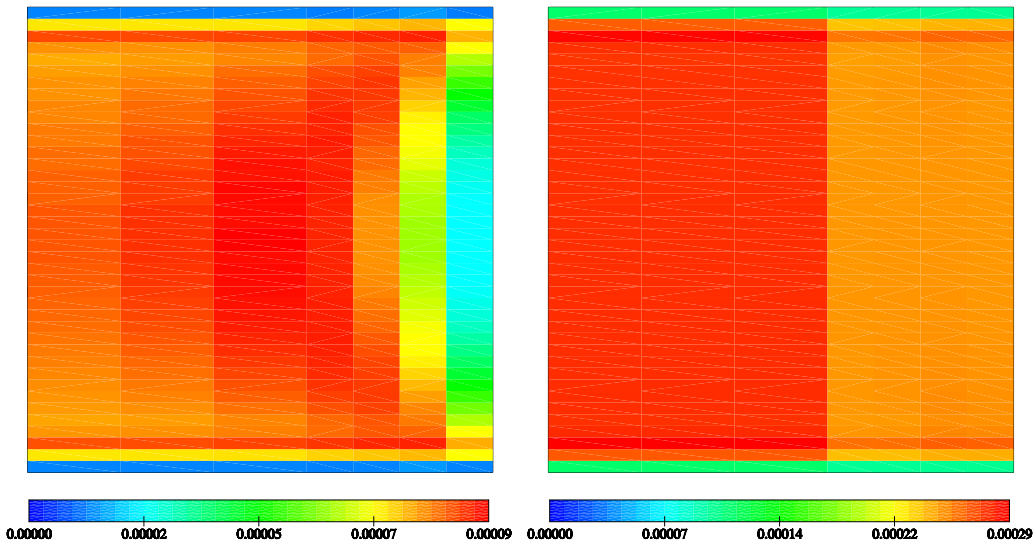
(a) Starting mesh (25 cells)

(b) Mesh after 2 cycles (70 cells)

(c) Mesh after 3 cycles (140 cells)

(d) Final mesh (280 cells)

Figure 5.13: Meshes for the creeping stagnation flow



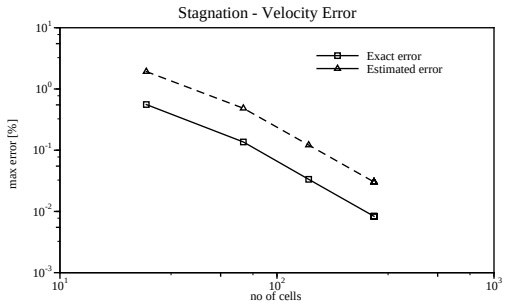
(a) Exact error

(b) Estimated error (given as average of face errors)

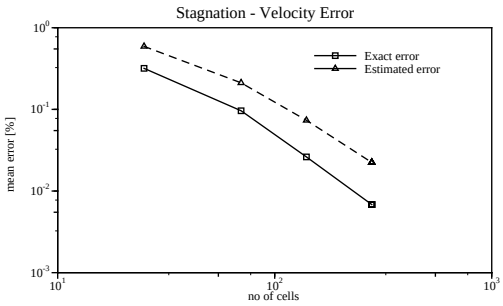
Figure 5.14: Velocity errors after 4 cycles of refinement for the creeping stagnation flow [m/s]

The graphs which show the reduction of exact and estimated errors in their maximum and mean values are presented in Figs. 5.15(a) and 5.15(b). The maximum estimated error is over-estimated by a factor of 3.5 such that the required maximum exact error is achieved after the third cycle on a mesh with 140 cells. The estimated mean error does not tend to the exact one, which is caused by the inconsistency between the FREE and the truncation error for the surface-normal gradient. Fig. 5.15(c) shows that the maximum exact error after four cycles of refinement on a mesh with 280 cells is the same as the maximum error on a uniform mesh with 1600 cells. Even though the refinement is not highly localised, the adapted solution required 17.5 % of the number of cells required on a uniform mesh. This was achieved due to directional refinement which occurs in the vertical direction for which the second derivative of y velocity component is not zero.

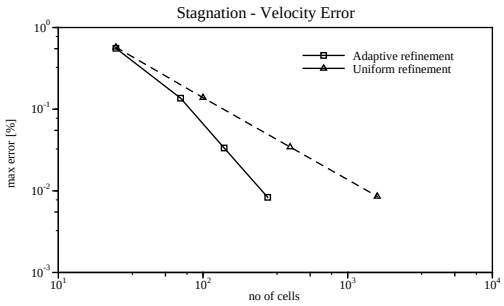
Table 5.2 shows the distribution of estimated face errors on the starting and the final mesh. The distribution of error has improved because the percentage of faces with low error has reduced and the percentage of faces with highest error which correspond to the required error has increased but is not yet optimal because there are still many faces with very small error. Zeros in the table show that there are no faces with the error in the range between 20 and 70% of the maximum-estimated



(a) Maximum error



(b) Mean error



(c) A comparison of the maximum-velocity error for uniform and adaptive refinement

Figure 5.15: Velocity-error scaling with adaptive refinement for the creeping stagnation flow ($|\mathbf{U}_{norm}| = 1.107 m/s$)

error in the final adapted mesh. Fig. 5.16 shows the distribution of the estimated face error on the final mesh. There are three groups of faces which can be identified:

- Horizontal faces have error almost equal to the required error.
- Vertical internal faces. The variation of velocity in the x direction is linear which results in small estimated error there. This suggests that the starting mesh is too fine in the x direction so the coarsening procedure which could remove those faces would have a beneficial influence on the error distribution.
- Boundary faces. The error on these faces is zero because the value of velocity is known there.

The second calculation has produced the same final mesh as in the previous case, as expected, but it required six adaptation cycles instead of four.

Figs. 5.17(a), 5.17(b), 5.17(c), 5.17(d), 5.17(e) and 5.17(f) show the meshes for all six cycles. The reason for the increased number of cycles are split-hexahedra

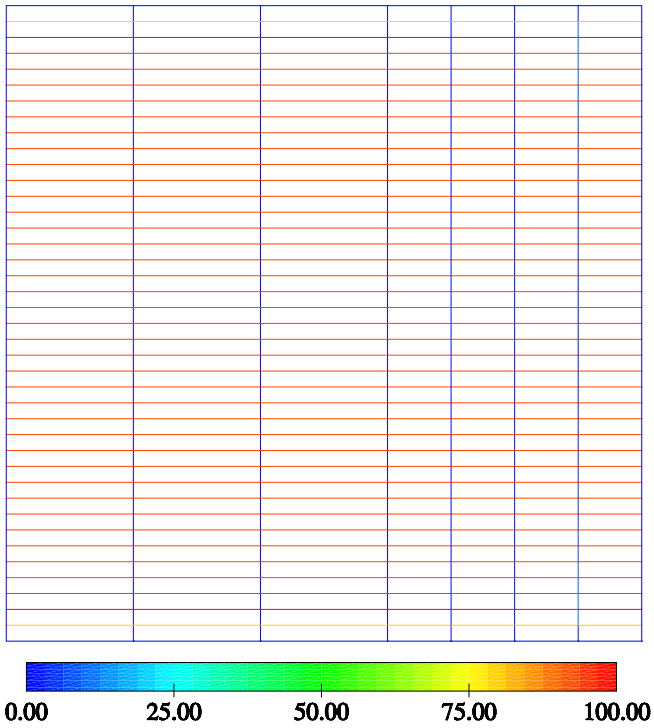
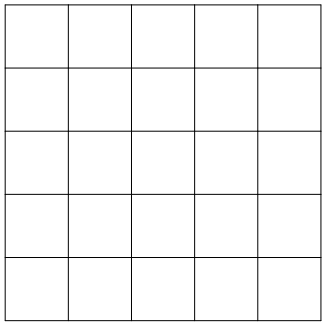


Figure 5.16: Estimated errors on the faces of the final mesh with 280 cells (given as percentage of the maximum-estimated error on that mesh)

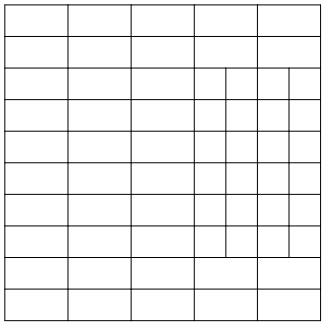
which appear after the second cycle which have the highest error, Figs. 5.19(a) and 5.19(b), and impair the reduction of error, Figs. 5.18(a) and 5.18(b). Split-hexahedra do not appear in the first calculation, meshes shown in Figs. 5.13(a), 5.13(b), 5.13(c) and 5.13(d), because Eqn. (5.4) is not satisfied on faces which would become split faces of split-hexahedra and thus cells which would become split-hexahedra are also refined resulting in the reduced number of adaptation cycles.

	% of maximum estimated error on a given mesh									
Mesh	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
Starting	66.6	0	0	0	0	0	0	0	16.7	16.7
Final	54.4	0.659	0	0	0	0	0	2.31	0	42.6

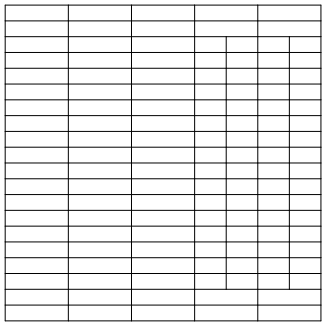
Table 5.2: Estimated errors on the faces of the final mesh with 280 cells (given as percentage of the maximum-estimated error on that mesh)



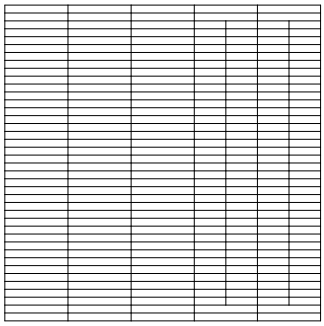
(a) Starting mesh (25 cells)



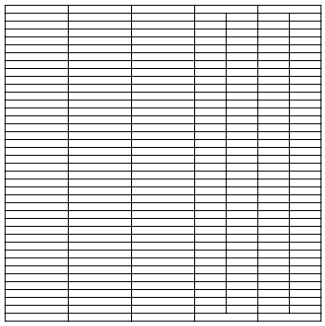
(b) Mesh after 2 cycles (62 cells)



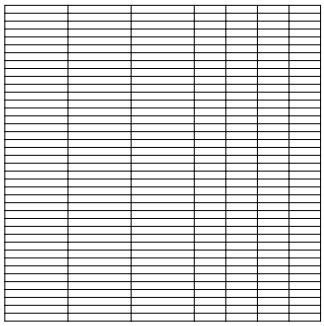
(c) Mesh after 3 cycles (132 cells)



(d) Mesh after 4 cycles (276 cells)

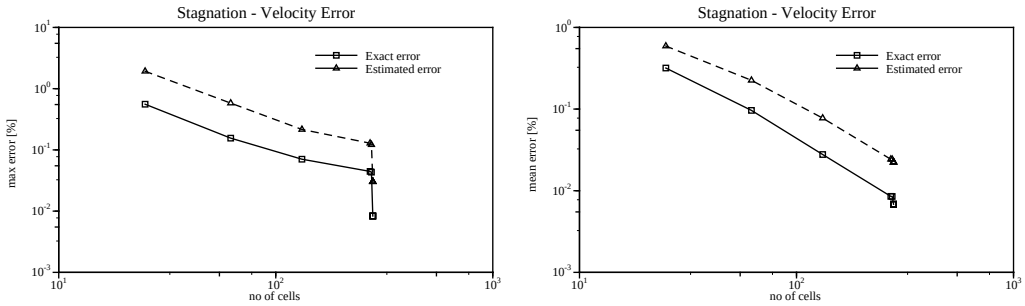


(e) Mesh after 5 cycles (278 cells)



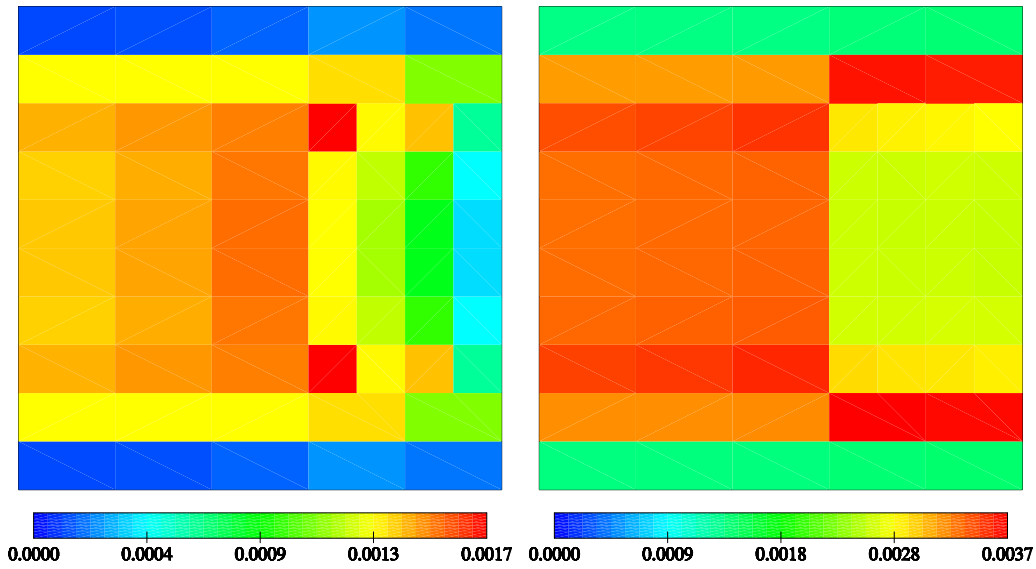
(f) Final mesh (280 cells)

Figure 5.17: Meshes for the creeping stagnation flow (second calculation)



(a) Maximum error (b) Mean error

Figure 5.18: Variation of velocity errors with adaptive mesh refinement for the stagnation flow (Second calculation) ($|\mathbf{U}_{norm}| = 1.107 m/s$)



(a) Exact error (b) Estimated error

Figure 5.19: Velocity-error fields after 2 cycles of adaptive refinement for the stagnation flow (Second calculation) [m/s]

5.4.3 Convection Transport of Heat with a Distributed Heat Source

This problem was described in Section 4.5.3. The adaptive calculation has been started from a (5 x 5) mesh. The maximum temperature found in the field is 1 ^oC and the maximum allowed error is set to:

$$E = 0.1\,\% \tag{5.9}$$

of this.

The required accuracy could not be achieved efficiently by using adaptive refinement so it was stopped after 6 cycles. Figs. 5.22(a) and 5.22(b) show that the error stops reducing effectively after the third cycle of adaptive refinement when the first split-hexahedra appear in the mesh, see Fig. 5.20. This happens because the volume

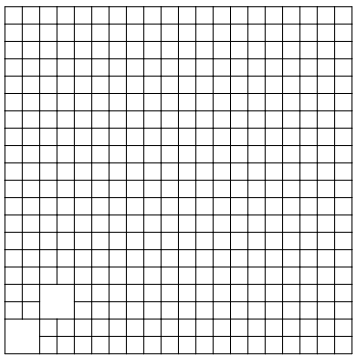
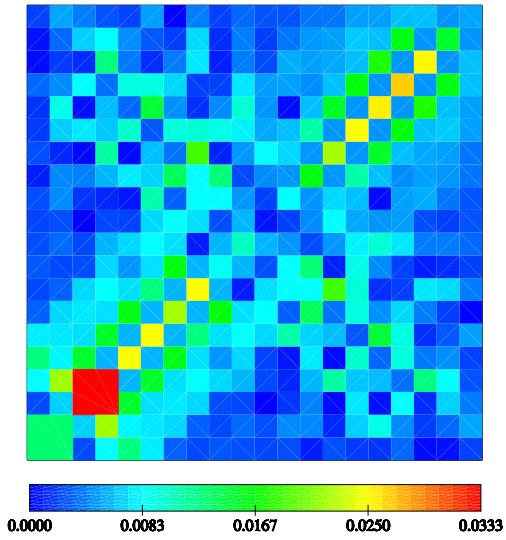
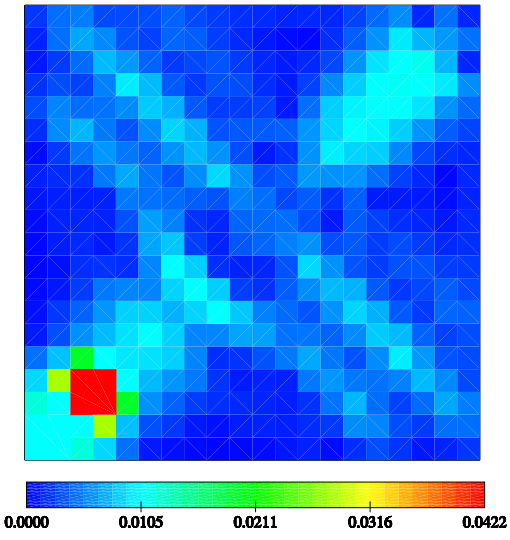


Figure 5.20: Mesh after 3 cycles of refinement for the convection transport case

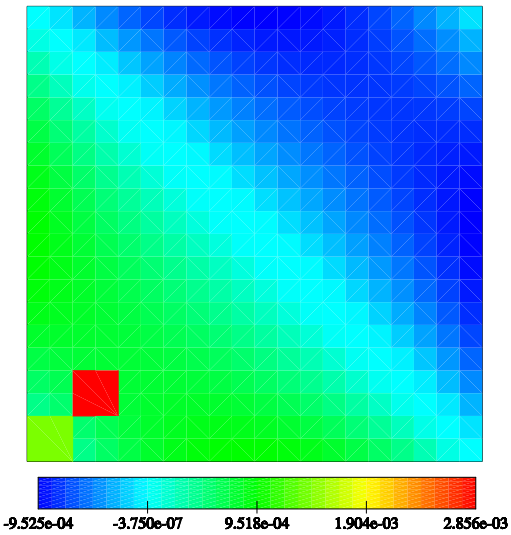
of split-hexahedron cells is four times larger than the volume of their neighbours thus having a large impact on the source-term discretisation, Eqn. (3.60). These differences in cell sizes cause unboundedness of the volume integral of the source term, shown in Fig. 5.21(c), which results in the increased error at the locations where this influence is important, Fig. 5.21(a). The agreement between the exact and the estimated temperature error, shown in Figs. 5.21(a) and 5.21(b) respectively, is considered satisfactory. The estimation of the exact error is very accurate during the first two cycles. The inaccuracy of the error estimation afterwards is attributed to the unboundedness of the temperature gradient field, shown in Fig. 5.21(d), which is used for error estimation.



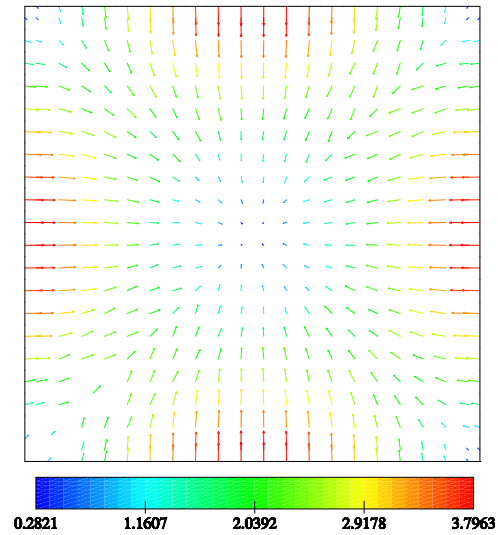
(a) Exact-temperature error (^oC)



(b) Estimated-temperature error (given as average of face errors (^oC)

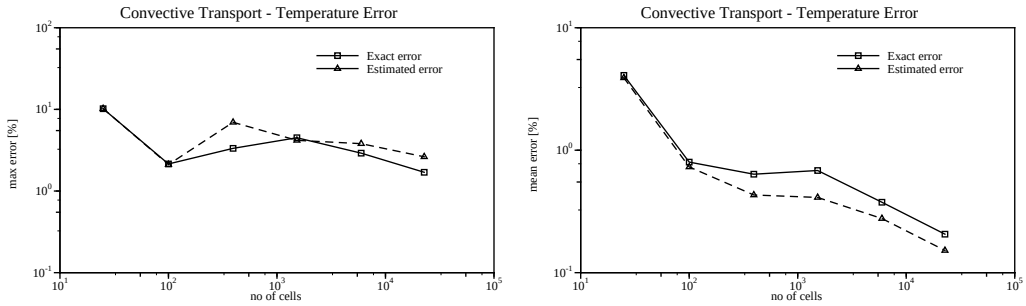


(c) Volume integral of the source term ($\frac{^oCm^3}{s}$)



(d) Temperature gradient ($^oC/m$)

Figure 5.21: Fields after 3 cycles of refinement



(a) Maximum error (b) Mean error

Figure 5.22: Variation of temperature errors with adaptive refinement for the convection transport case ($T_{norm} = 1^oC$)

5.4.4 Convection and diffusion of a Temperature Profile without a Heat Source

This test case is a solution of the general transport equation for temperature without the heat source. Fig. 5.23 shows the temperature field which has high gradients near the inlet which diminish by the influence of diffusion. The purpose of this case is to test the error estimator and the refinement procedure near strong features.

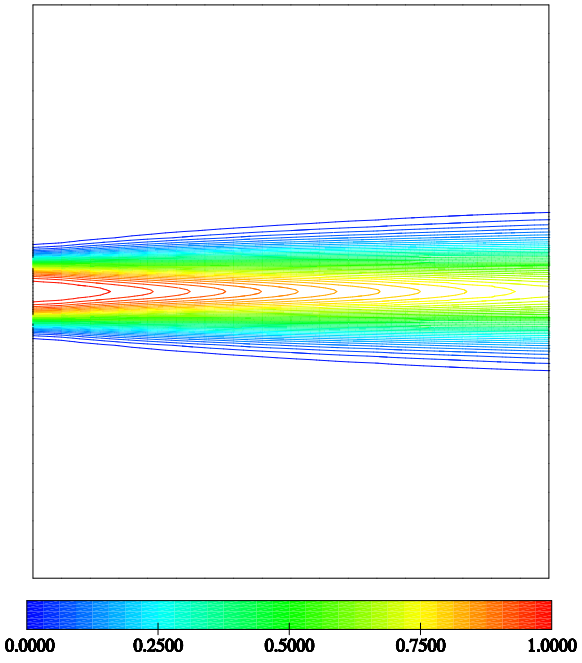


Figure 5.23: Temperature field for the internal layer case [^{o}C]

The analytical solution for this case is [152]:

$$T(x,y) = A\sum_{k=1}^{\infty}(\alpha_k\phi_{1k}(x) + \beta_k\phi_{2k}(x))\psi_k(y), \tag{5.10}$$

where $\phi_{1k}(x)$, $\phi_{2k}(x)$, $\psi_k(y)$ and b_k are:

$$\phi_{1k}(x) = e^{\frac{xPe}{2}}\frac{sinh(b_k(1-x))}{sinh(b_k)}, \tag{5.11}$$

$$\phi_{2k}(x) = e^{\frac{(x-1)Pe}{2}}\frac{sinh(b_k x)}{sinh(b_k)}, \tag{5.12}$$

$$\psi_k(y) = sin(k\pi y), \tag{5.13}$$

$$b_k = \sqrt{\left(\frac{Pe}{2}\right)^2 + (k\pi)^2}, \tag{5.14}$$

and α_k and β_k are constants:

$$\alpha_k = \frac{2}{k\pi}(cos(0.45k\pi) - cos(0.55k\pi)), \tag{5.15}$$

$$\beta_k = \alpha_k\frac{\frac{b_k e^{\frac{Pe}{2}}}{sinhb_k}}{\frac{Pe}{2} + b_k\frac{cosh(b_k)}{sinh(b_k)}}, \tag{5.16}$$

where x and y are dimensionless variables defined as:

$$x = \frac{X}{L}$$
$$y = \frac{Y}{L}$$

$L = 1m$ being a reference length. Peclet number $Pe = \frac{|\mathbf{U}|L}{\nu}$ is 1000, the velocity $\mathbf{U}$ is uniform and the value of U_x component is set to 1 m/s while other components are zero. ν is set to:

$$\nu = 0.001\,[m^2/s]. \tag{5.17}$$

The solution domain and the boundary conditions used for calculations are depicted in Fig. 5.24. The interpolation scheme used for convection is CD.

Calculations for this case were started from a (18 x 20) uniform mesh, Fig. 5.25(a). The maximum allowed temperature error was set to:

$$E = 0.5\,\% \tag{5.18}$$

of the maximum temperature $T_{max} = 1^oC$.

The desired error was achieved after five cycles of adaptive refinement. Fig. 5.25(b) shows the resulting mesh. As expected, the refinement is localised in the region of

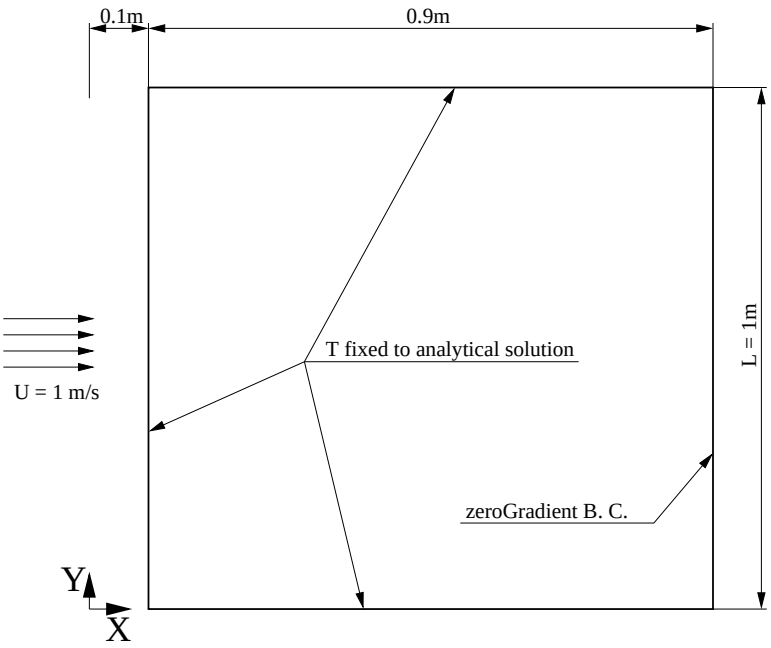
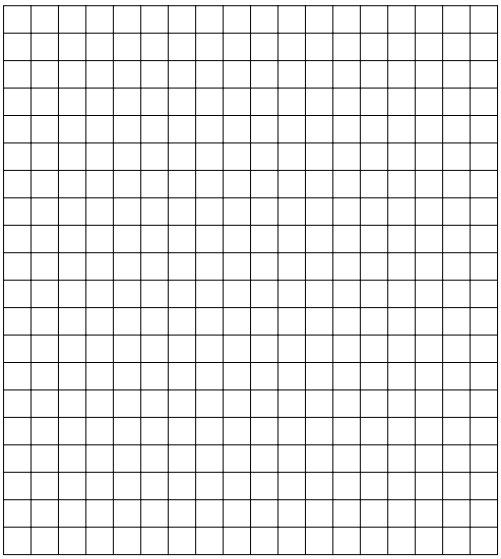


Figure 5.24: Solution domain and boundary conditions for the convection and diffusion of heat

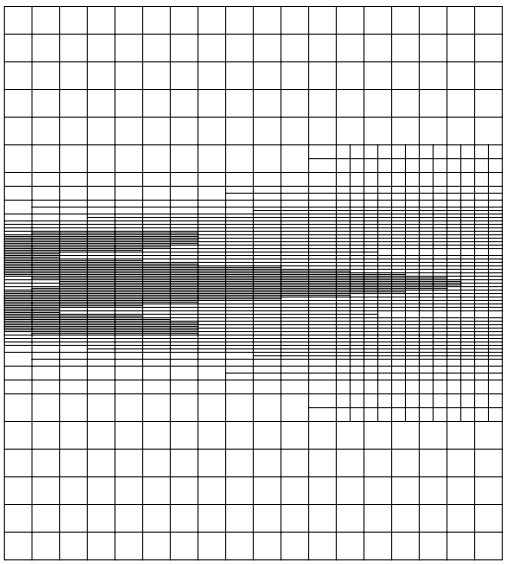
steep gradients. Most of refinement occurs in the cross-stream direction in which the diffusion is dominant. The region near the right boundary is also refined in the stream-wise direction to reduce the oscillations from the CD convection scheme, Fig. 5.27(a).

Figs. 5.25(c) and 5.25(d) show that errors are severely over-estimated. This mostly occurs near the inlet where diffusion is high, Fig. 5.26(b). The reason for this is twofold. It was shown in Chapter 4 that the diffusion part of the estimator is not consistent with the truncation error for the diffusion term. The second issue here is the accuracy of the gradients calculated from the solution, Fig. 5.27(b). On coarse meshes the gradient contains high error itself which also has an impact on the quality of error estimation. Fig. 5.25(e) shows a comparison between the exact errors for adaptive and uniform refinement. The error in the final adapted mesh is comparable to the error on the third uniform mesh consisting of 5760 cells while the adapted mesh consists of 1706 cells. This is 29% of the number of cells in the uniform mesh.

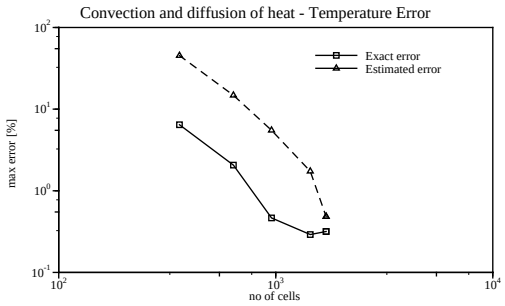
In order to check the efficiency of mesh refinement an additional test was made. The refinement was driven by the exact face error obtained as the difference between the FV solution interpolated onto the faces by using Eqn. (3.14) and the exact solution at the face centres. The maximum face error dropped below the required



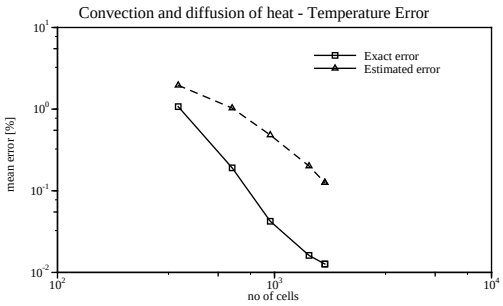
(a) Starting mesh for the convection and diffusion of heat (360 cells)



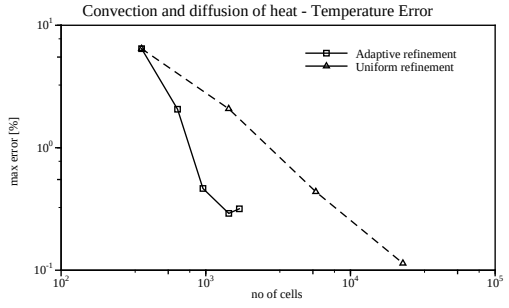
(b) Mesh after 5 cycles of refinement for the convection and diffusion of heat (1706 cells)



(c) Maximum error

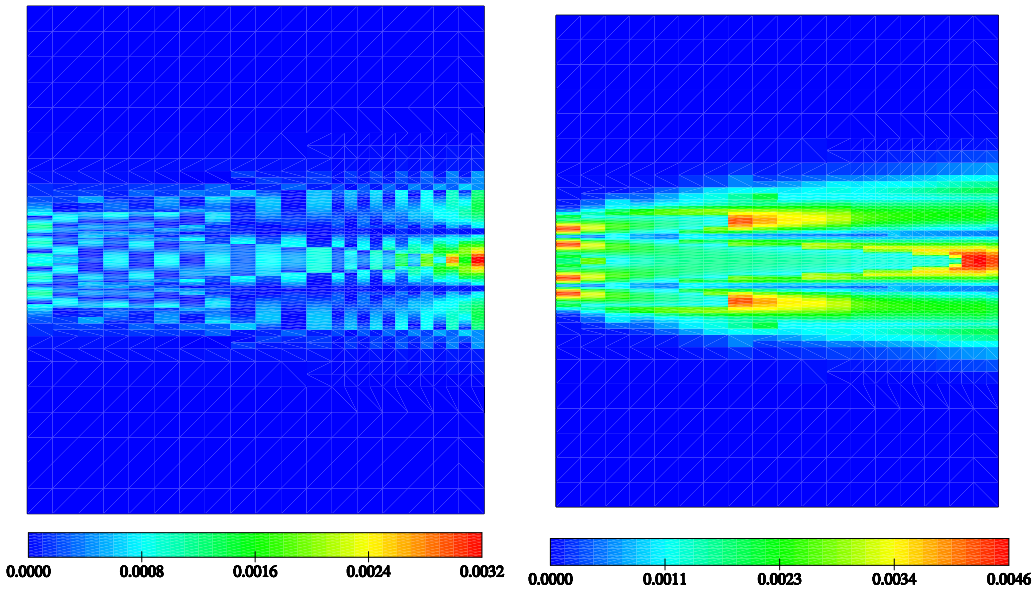


(d) Mean error



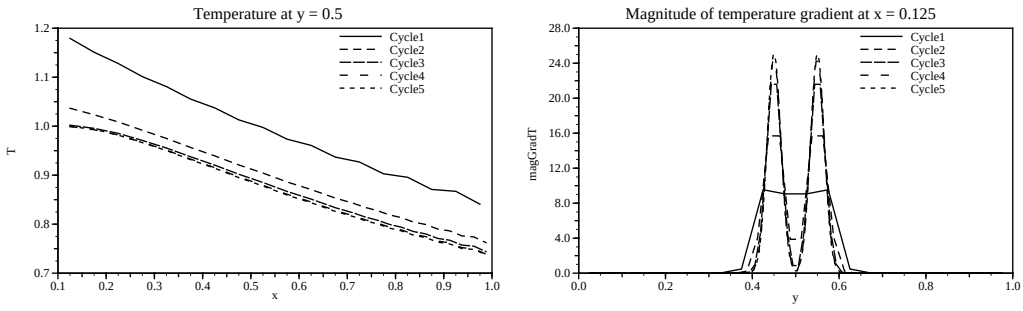
(e) A comparison of the maximum temperature error for uniform and adaptive refinement

Figure 5.25: Variation of temperature errors with adaptive refinement for the convection and diffusion of heat ($T_{norm} = 1^oC$)



(a) Exact error (b) Estimated error

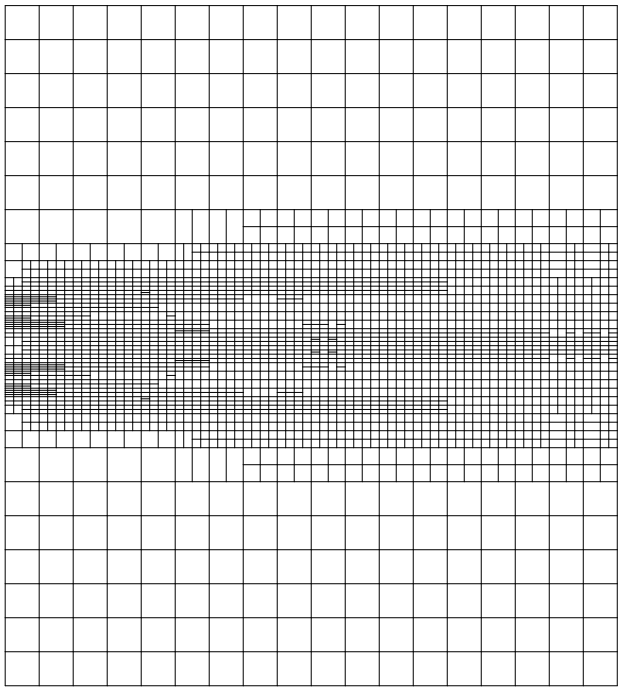
Figure 5.26: Temperature errors after 5 cycles of refinement for the convection and diffusion of heat [^{o}C] (1706 cells)



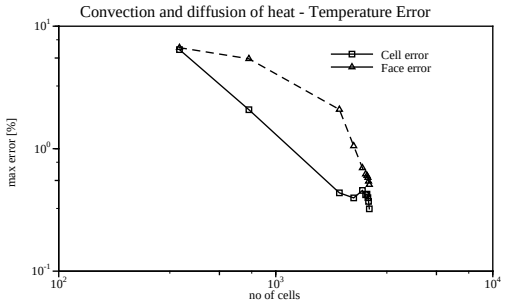
(a) Temperature at y = 0.5 for the convection and diffusion of heat (^oC)

(b) Magnitude of temperature gradient at x = 0.125 for the convection and diffusion of heat (^oC/m)

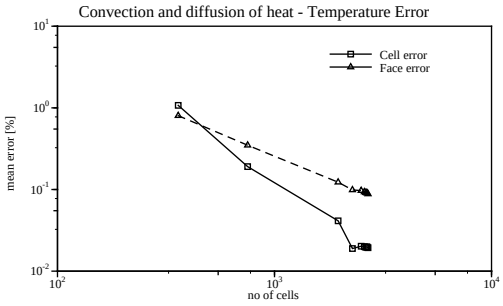
Figure 5.27: Temperature and its gradient for the convection and diffusion of heat



(a) Mesh after 10 cycles of refinement driven by exact face error (2698 cells)



(b) Variation of maximum cell and face errors with adaptive refinement for the convection and diffusion of heat ($T_{norm} = 1^oC$)



(c) Variation of mean cell and face errors with adaptive refinement for the convection and diffusion of heat ($T_{norm} = 1^oC$)

Figure 5.28: Mesh and errors for the calculation driven by the exact face errors

tolerance after 10 cycles of refinement and the final mesh consists of 2698 cells. This is 58% more than 1706 cells obtained by using the FREE even though Figs. 5.25(b) and 5.28(a) show similar patterns of refinement suggesting that the FREE is capable of driving refinement efficiently. Figs. 5.28(b) and 5.28(c) show differences between the errors at face centres and nodal errors. These differences are a consequence of the interpolation practice used and different locations at which the error is measured.

5.5 Summary and Conclusions

A directional-mesh-refinement procedure for hexahedral meshes was presented in this chapter. It has shown variable success in producing solutions of required accuracy. Solutions of required accuracy were produced with savings in the number of cells on most cases and for such cases the analysis of errors on the faces has shown that the distribution of error improves with mesh refinement. Room for improvement still exists because faces with low error are also generated during the refinement process. This suggests that a coarsening procedure could have a beneficial effect on error equidistribution, but only if it is performed such that it does not reduce mesh quality. Problems in reducing the error below the required limit were encountered for the convection case, for which the discretisation of source terms becomes very inaccurate on meshes with rapid variation of cell sizes, and results in unbounded volume integrals of source terms which increase the error.

It was also shown that the FREE over-estimates the error on diffusion-dominated problems, which makes the final adapted meshes over-refined for such cases. But the fact that the error is estimated on mesh faces causes generation of anisotropic meshes, thus resulting in savings on the number of cells when compared to uniform refinement.

Chapter 6

Further Case Studies

6.1 Introduction

In this chapter the directional mesh-adaptation algorithm, presented in Chapter 5, will be tested further on four cases of engineering interest. These include 2D laminar flow and 3D turbulent-flow examples. They were chosen to represent problems occurring in everyday engineering practice. Analytical solutions for them do not exist, so accuracy will be judged by comparing the solution on the adapted mesh with the one obtained on a very fine uniform mesh considered to be a benchmark solution. The accuracy of solutions on adapted meshes will be measured by monitoring some flow features *eg.* pressure drop, shear stress, forces, *etc.* Test cases involving supersonic flows were not considered as the errors are highly localised near the shocks such that most refinement would occur there.

6.2 Flow Over a Cavity

The first case considered is an incompressible laminar channel flow over a cavity. The geometry consists of a 2D channel with a cavity in the bottom wall, shown in Fig. 6.1.

The flow is at $Re = 200$ based on the channel height H and the average inlet velocity U_{avg}. At the inlet a parabolic velocity profile is prescribed:

$$\mathbf{U}_x = 6\,U_{avg}\left[\frac{y}{H} - \left(\frac{y}{H}\right)^2\right].$$

The calculation was started on the mesh with 36 cells depicted in Fig. 6.2. Second-order Central Differencing (CD) was used for discretisation of convective

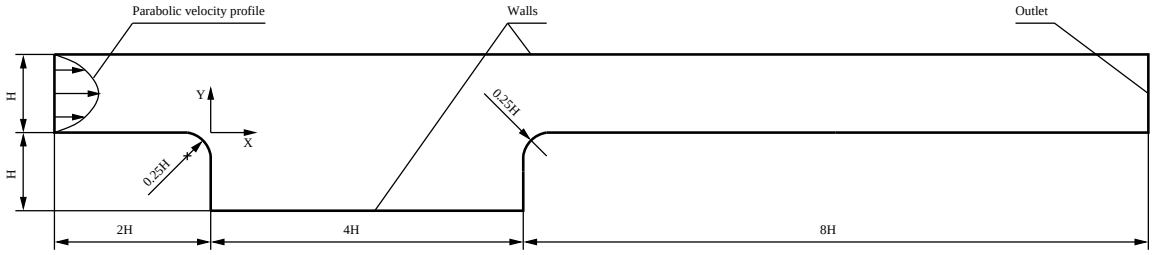


Figure 6.1: Geometry and boundary conditions for the flow over a cavity

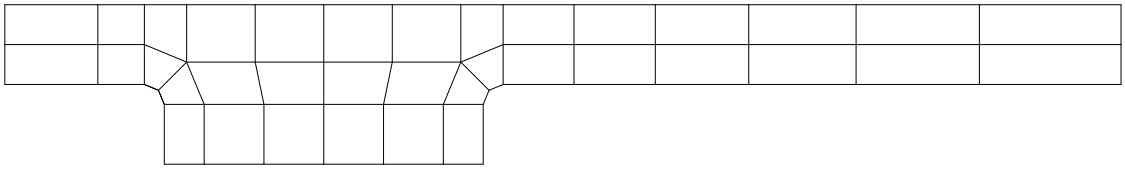


Figure 6.2: Starting mesh for the flow over a cavity (36 cells)

terms. Gradients of all fields are evaluated by using the Gauss-divergence Theorem, Eqn. (3.26).

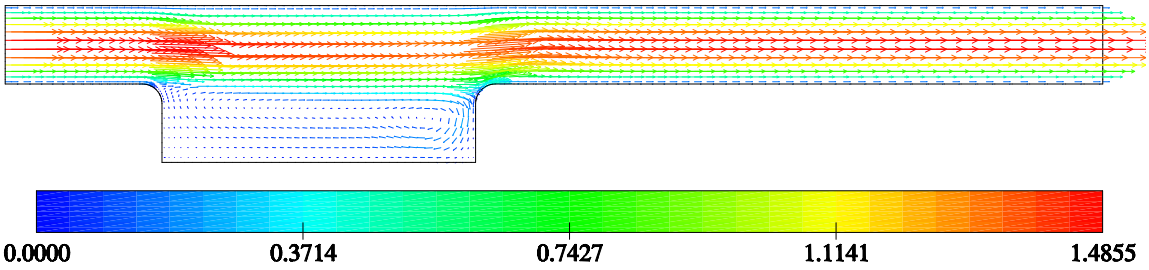
The required maximum velocity error in the solution is set to:

$$E = 1\,\%,$$

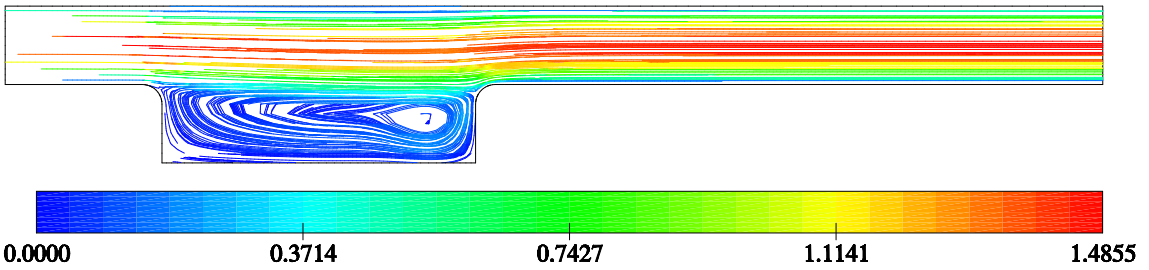
of the average inlet velocity U_{avg}. The solution is compared with a benchmark solution obtained on the 501552 cells mesh with the maximum estimated velocity error of 0.2% of the inlet velocity U_{avg}.

The predicted velocity field on the final adapted mesh with 3257 cells can be found in Figs. 6.3(a) and 6.3(b). The flow is aligned with the x axis within the channel and the variation of velocity in the cross-stream direction is parabolic. A stagnation point at the top-right corner of the cavity, Fig. 6.3(d), is a region of high velocity and pressure gradients, Figs. 6.3(d) and 6.4. The vortex formed within the cavity has an impingement zone near the bottom-right corner where the flow parallel to the right wall meets with the bottom wall. The separation near the upper-left corner of the cavity is a region of moderate velocity and pressure gradients when compared to the zone near the stagnation point, Figs. 6.3(c) and 6.4.

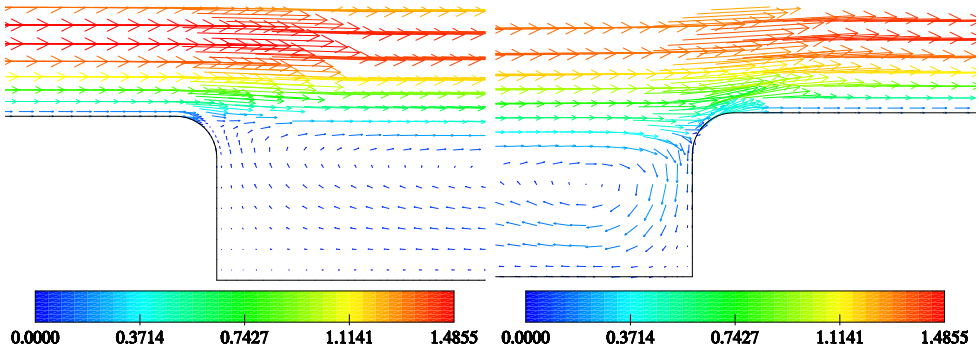
The calculation stopped after 8 cycles of refinement. The final mesh has 3257 cells and is shown in Fig. 6.5. It is highly refined in the region near the stagnation



(a) Velocity



(b) Streamlines



(c) Zoom around the top-left corner of the cavity

(d) Zoom around the top-right corner of the cavity

Figure 6.3: Velocity field for the flow over a cavity on the final adapted mesh with 3257 cells (normalised by U_{avg})

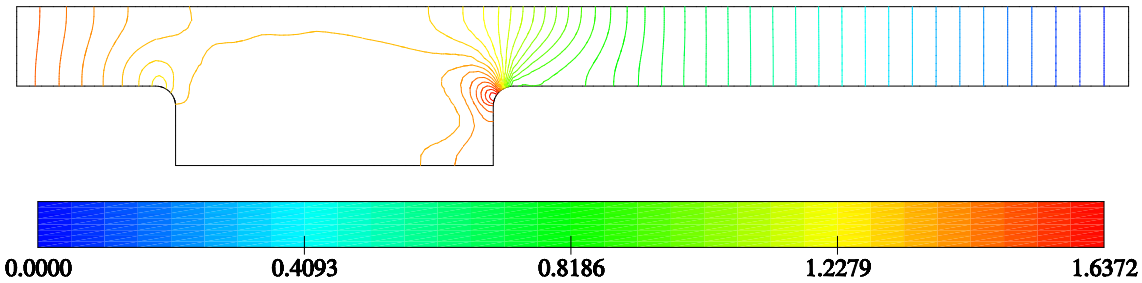


Figure 6.4: Pressure coefficient field for the flow over a cavity on the final adapted mesh with 3257 cells

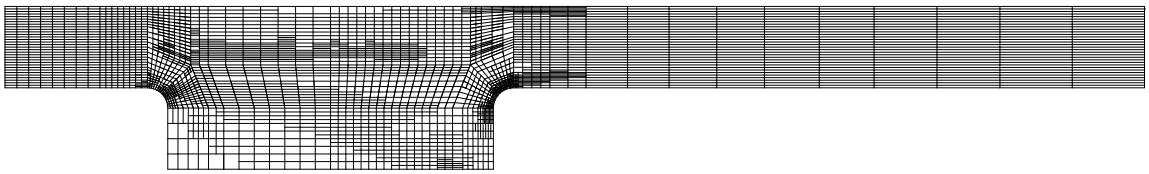
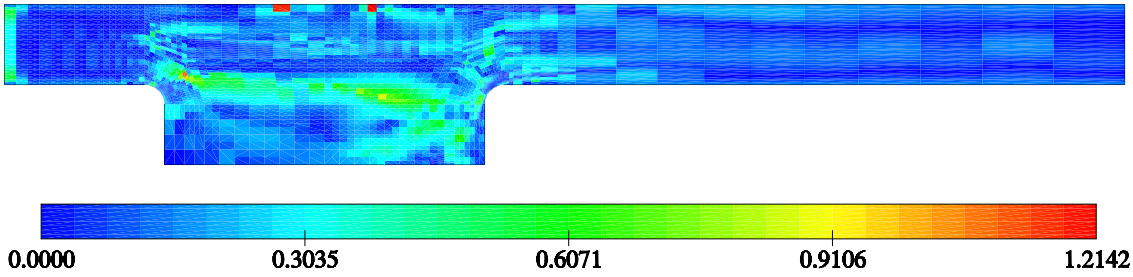


Figure 6.5: Mesh after 8 cycles of refinement for the flow over a cavity (3257 cells)

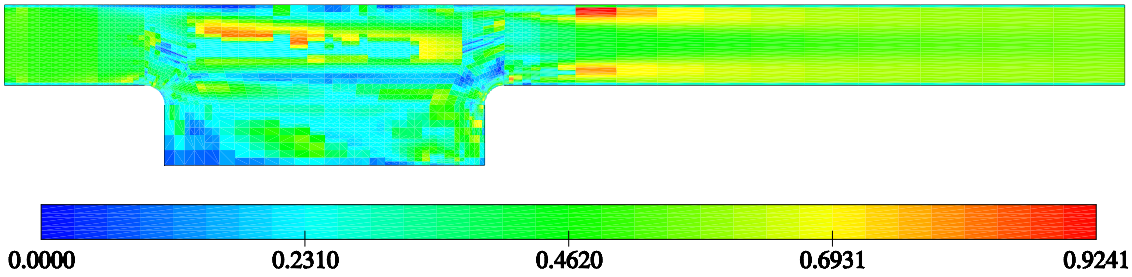
point, as would be expected in a region of high gradients. Some refinement occurred within the cavity to resolve the impingement zone and the shear layer. The mesh is unexpectedly fine within the channel and the main reason for such behaviour is the fact that, for the uniform mesh as present here, the discretisation of the diffusion term can resolve the parabolic profile within the channel exactly, Eqn. (3.56), while the error estimator is sensitive to the second gradient and reports non-zero error for such conditions, Eqn. (4.45). Fig. 6.6(a) shows that the exact error after 8 cycles of adaptive refinement is negligible within the channel while the estimated error shown in Fig. 6.6(b) is non-zero there. This undesired behaviour has caused poor agreement of the estimated error with the exact one.

Figs. 6.7(a) and 6.7(b) show the reduction of maximum and mean-velocity errors with mesh refinement. The maximum error exhibits steep reduction during the final three cycles when the refinement is localised to a small number of cells with high error. The estimated mean-velocity error does not tend to the exact error because of the over-estimation within the channel.

The distribution of estimated errors on faces is shown in Table 6.1.

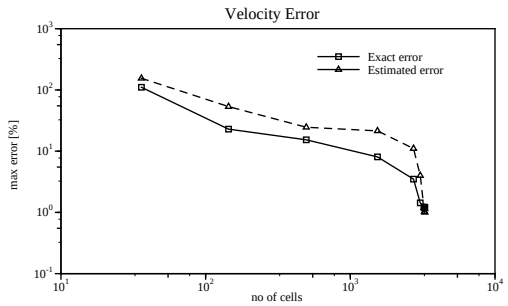


(a) Exact error

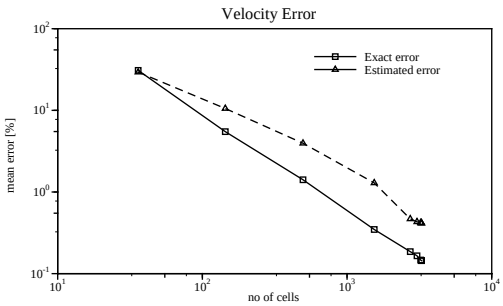


(b) Estimated error (cell values are the weighted average of errors on cell faces)

Figure 6.6: Errors for the flow over a cavity on the final adapted mesh with 3257 cells (given as percentage of U_{avg})



(a) Maximum error



(b) Mean error

Figure 6.7: Variation of velocity errors with adaptive mesh refinement for the flow over a cavity (errors given as percentage of U_{avg})

	% of maximum estimated error on a given mesh									
Mesh	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
Starting	58.2	8.8	4.4	11	2.2	6.5	6.5	1.09	0	1.1
Final	37.1	9.9	13.1	10.2	8.5	12.4	4.2	2.2	1.5	0.8

Table 6.1: Distribution of face errors for the cavity case

The refinement procedure tends to reduce the number of faces with low estimated error and increase the number of faces with nearly average estimated error thus improves error distribution. Fig. 6.8 shows the estimated velocity error on the faces

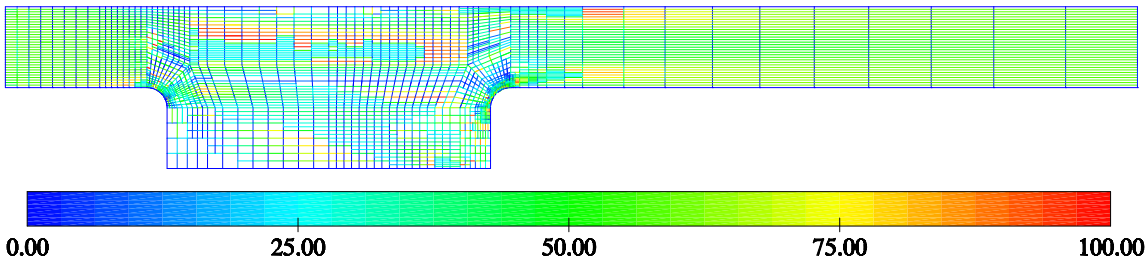


Figure 6.8: Estimated-velocity error on the faces of the final mesh with 3257 cells (given as percentage of the maximum estimated error on that mesh)

of the final mesh. The main reason for this unfavourable error distribution is the existence of many faces perpendicular to the flow which have low error due to slow variation of velocity field in that direction.

The pressure drop, defined as the difference between the average pressures at the inlet P_{inlet} and outlet boundaries P_{outlet}, is also used as an indicator of accuracy. The pressure drop coefficient (C_p) defined as:

$$C_p = \frac{P_{inlet} - P_{outlet}}{\frac{1}{2}\rho U_{avg}^2}$$

can be found in Table 6.2. The refinement makes the pressure drop reach its final value quickly. It does not change during the last four cycles of refinement suggesting that the remaining faces with estimated error above the required do not influence the global accuracy of the pressure field. The pressure drop on the final adapted mesh is in good agreement (0.2% error) with the benchmark pressure drop. This suggests that the adaptive refinement is capable of achieving accurate solutions with savings

Adaptive refinement	
Cells	C_p
36	1.241
144	1.351
496	1.468
1541	1.508
2734	1.518
3044	1.518
3227	1.518
3257	1.518

Uniform refinement	
Cells	C_p
172	1.438
1548	1.502
13932	1.518
125388	1.52
501552	1.52

Table 6.2: Pressure-drop coefficients for the flow over a cavity

in the number of cells because the final adapted mesh has 0.65% of the number of cells in the benchmark mesh.

6.3 S-shaped Pipe Bend

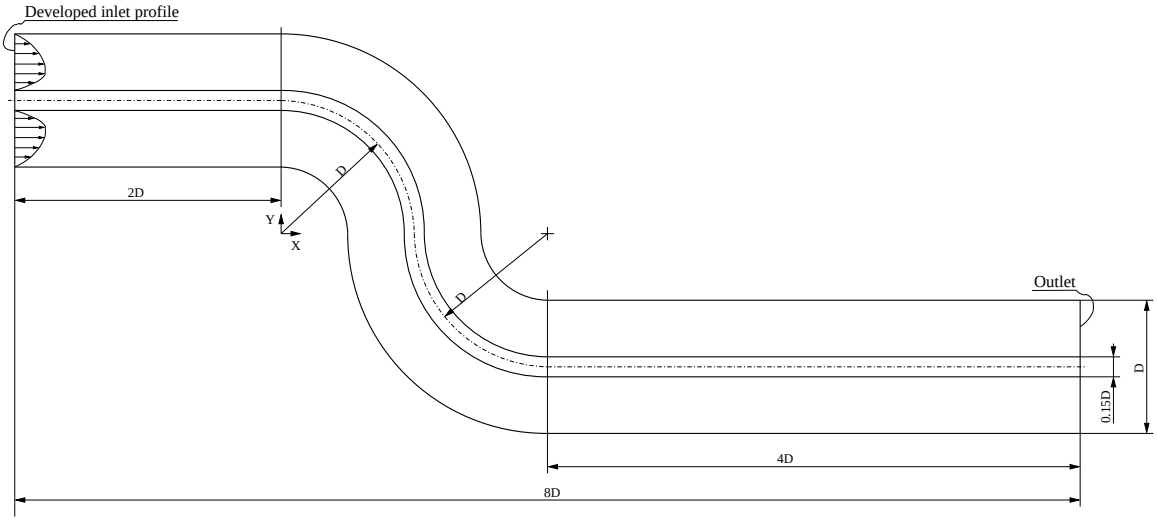
The second example is a 3D laminar flow in an annular pipe with a S-shaped bend which serves as a fluid mixer. The geometry and the boundary conditions for this case are shown in Figs. 6.9(a) and 6.9(b).

The Reynolds number of the flow is $Re = 750$ based on the outer pipe diameter D and the average inlet velocity U_{avg}. The velocity profile applied at the inlet boundary corresponds to a fully-developed flow in an annulus, thus:

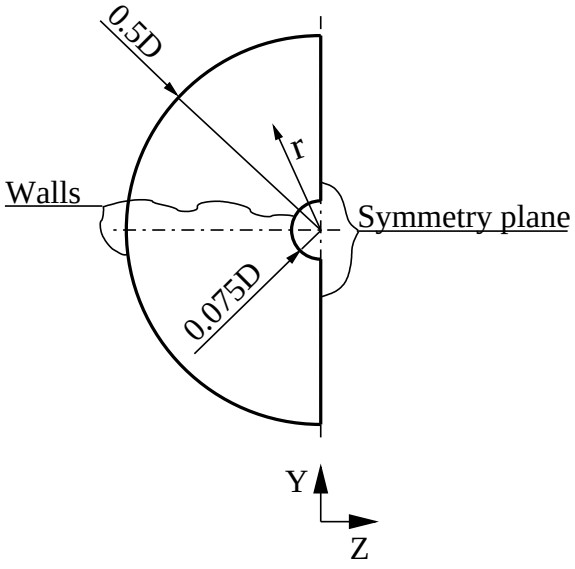
$$\mathbf{U}_x = \frac{-2\,U_{avg}}{R_2^2 + \frac{R_2^2 - R_1^2}{ln\frac{R_1}{R_2}} + R_1^2}\left[\frac{R_2^2 - R_1^2}{ln\frac{R_1}{R_2}}\,ln\frac{r}{R_1} + r^2 - R_1^2\right]$$

where r is the radial coordinate, $R_2 = 0.5D$ and $R_1 = 0.075D$.

The adaptive calculation was started from the 270 cell mesh, depicted in Fig. 6.10.



(a) Side view



(b) Front view

Figure 6.9: Case setup for the S-bend case

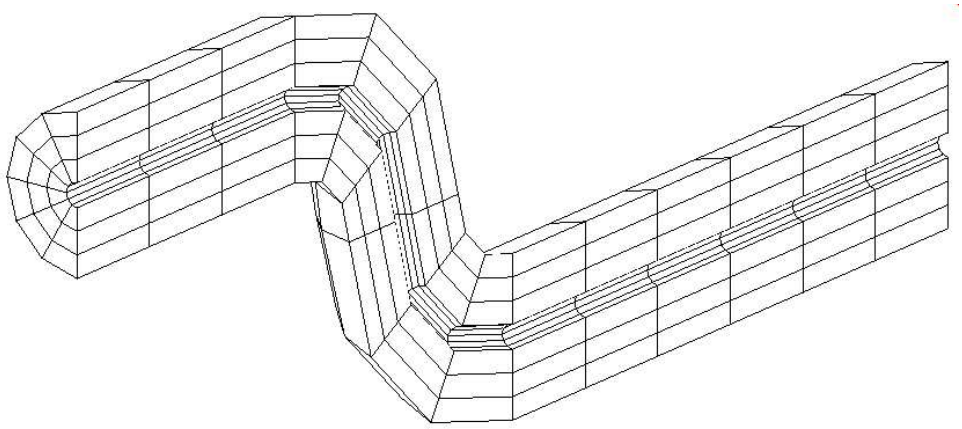


Figure 6.10: Starting mesh for the S-bend case (270 cells)

The convection term is approximated by using the second-order CD. Gradients of all fields are evaluated by using the Gauss-divergence Theorem, Eqn. (3.26).

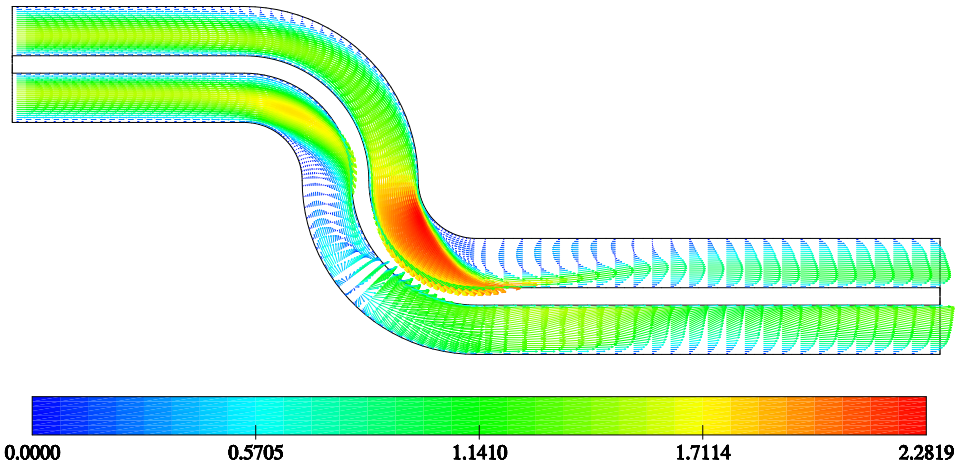
Figs. 6.11(a), 6.11(b) and 6.11(c) show the flow field for this problem. The largest velocity can be found in the bend and is a consequence of reduced area due to flow separation within the bend. There is also a separation in the straight part of the annulus downstream of the bend. Fig. 6.11(a) shows that boundary layers are very thin in the region of maximum velocity, which require a fine mesh to resolve it accurately. The pressure field, depicted in Fig. 6.12, is smooth with locally large gradients near the zone of high shear. Velocity and pressure in the section after the bend are shown in Figs. 6.13(a) and 6.13(b). The flow has cross-stream vortices whose strength will also be used to assess the numerical accuracy.

The required maximum error was:

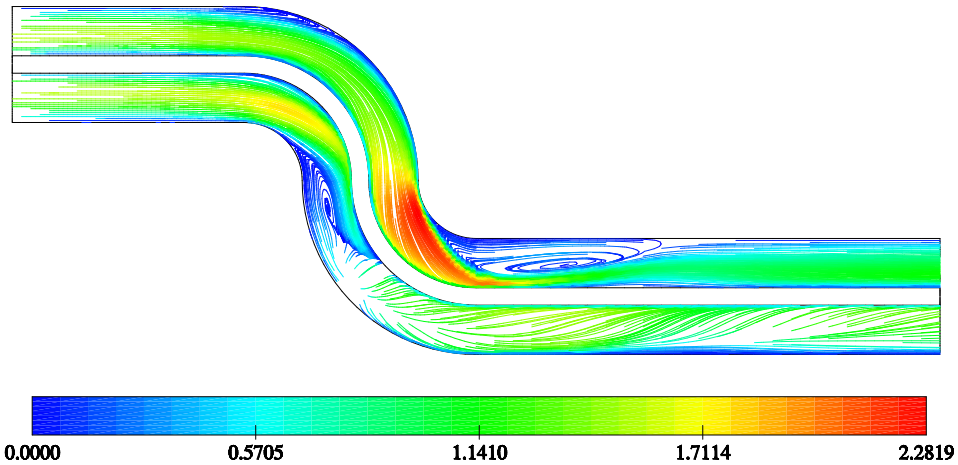
$$E = 1\,\%$$

based on the maximum velocity $U_{max} = 2.28\,U_{avg}$. The accuracy of the solution obtained by adaptive refinement will be compared to the benchmark obtained on a uniform mesh with 5.12 million cells. The maximum error in the benchmark solution, estimated using the FREE, is 5% of U_{max}.

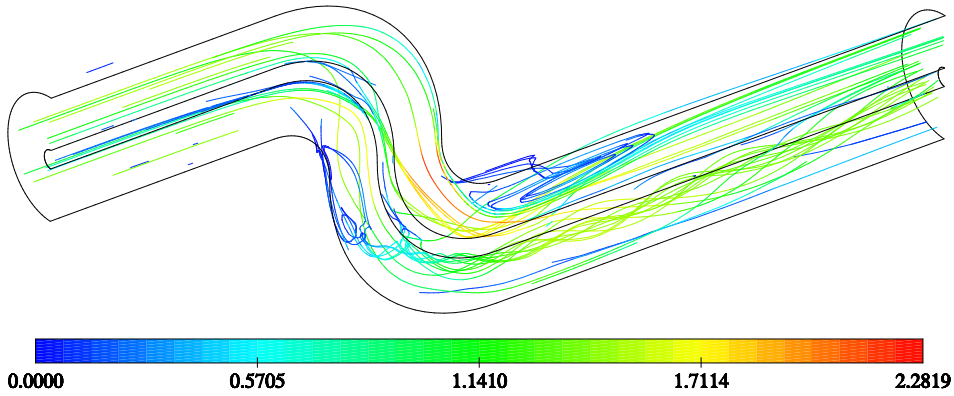
The calculation stopped after 9 cycles of adaptive refinement after reaching the target maximum error, as shown in Fig. 6.14(a). The mesh after 9 cycles of refinement consists of 390787 cells and is highly refined near the walls, Fig. 6.15(a). The refinement is primarily driven by the steep velocity gradients normal to the wall. The mesh is also highly refined within the shear layers near the separation zones



(a) Velocity field in the symmetry plane



(b) Streamlines in the symmetry plane



(c) Particle traces

Figure 6.11: Velocity field in for the S-bend case obtained on the final adapted mesh with 390787 cells (normalised by U_{avg})

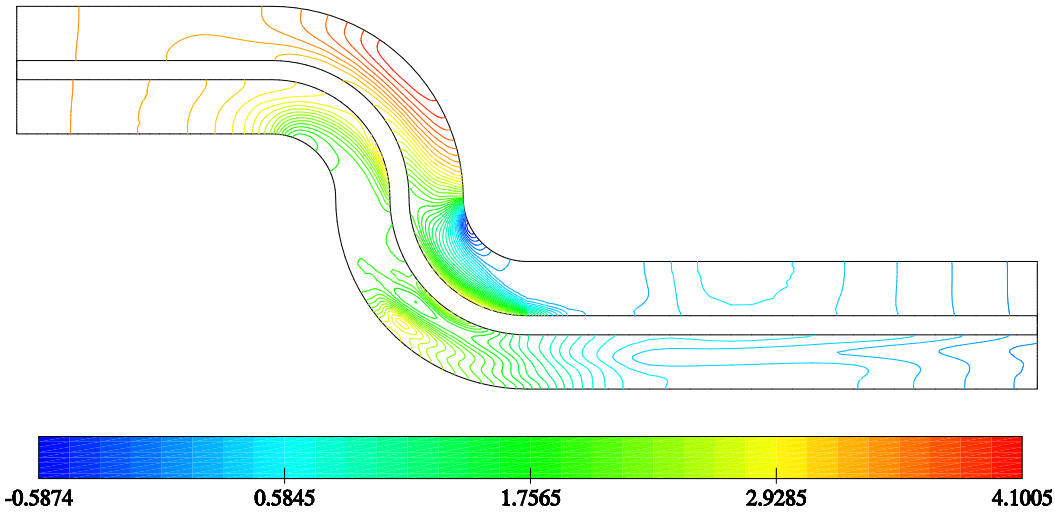
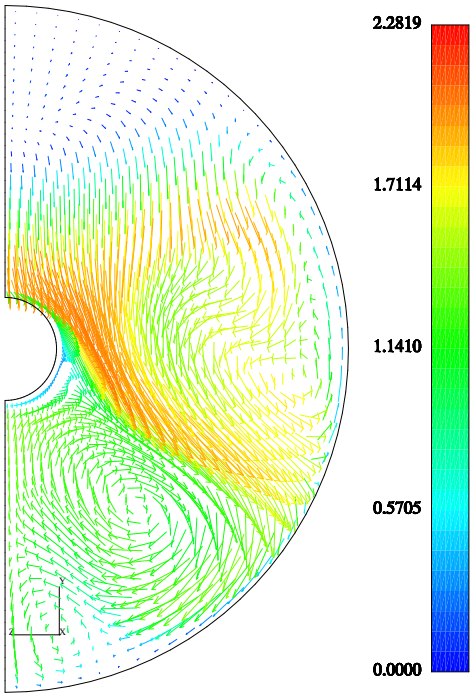


Figure 6.12: Pressure coefficient in the symmetry plane for the S-bend case obtained on the final adapted mesh with 390787 cells

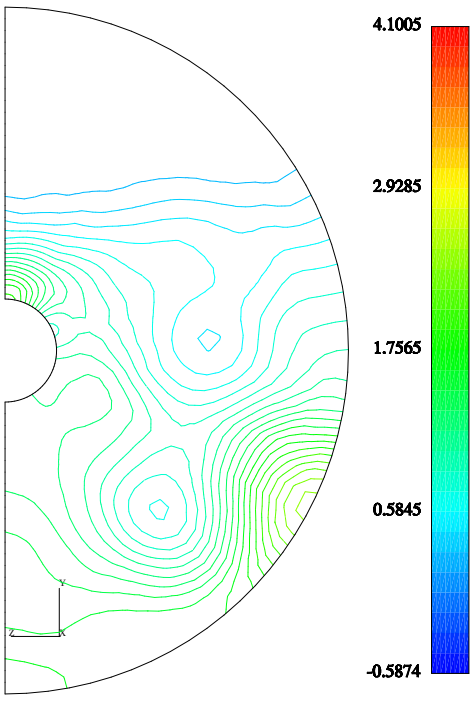
located downstream of the bend which are also regions of steep gradients.

The maximum estimated and exact velocity errors are in good agreement during the first few cycles, see Figs. 6.14(a) and 6.14(b). The unexpected behaviour of maximum and mean exact error during the last three cycles is due to the fact that the adapted mesh, Fig. 6.15(a), becomes finer in the regions of high error than the mesh for the benchmark solution. Fig. 6.14(a) shows that the maximum estimated error reduces more quickly than in case of uniform refinement suggesting that the adaptation procedure removes high estimated errors efficiently. This is not the case for the average estimated errors, shown in Fig. 6.14(b), which show that there is no saving in the number of cells between the adaptive procedure and uniform refinement. Fig. 6.16(b) shows the difference between the benchmark solution and the final adapted solution which is highest near the inner pipe wall where most refinement occurred. The agreement between the estimated and "exact" error fields is poor due to the limited accuracy of the benchmark solution. The areas of the remaining error are captured by the error estimator *i.e.* the shear layers near the downstream separation.

The accuracy was also judged by comparing the overall pressure drop, the magnitude of the force acting on the annulus walls and the average vorticity magnitude within the domain. The pressure drop is defined as the difference between the average pressure at inlet (P_{inlet}) and outlet (P_{outlet}) boundaries, and is given as a pressure

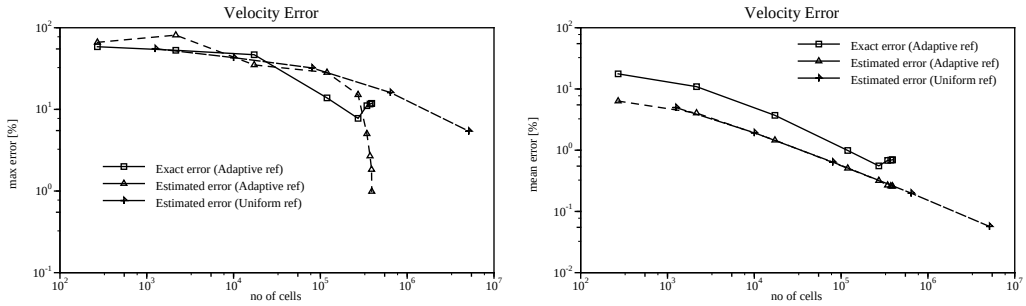


(a) Velocity field (normalised by U_{avg})



(b) Pressure coefficient

Figure 6.13: Section at $X = 2D$ (390787 cells)



(a) Maximum error (b) Mean error

Figure 6.14: Variation of velocity errors with adaptive mesh refinement for the S-bend case (given as percentage of U_{max})

coefficient defined as:

$$C_p = \frac{P_{inlet} - P_{outlet}}{\frac{1}{2}\rho U_{avg}^2}.$$

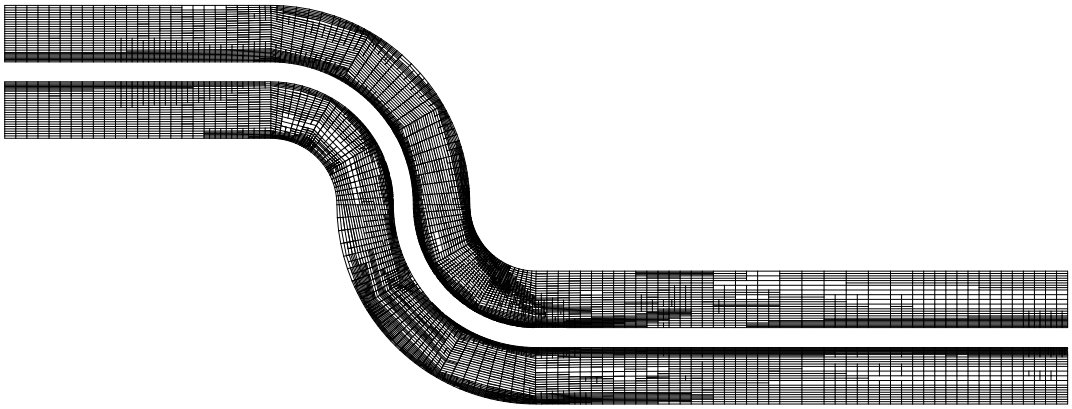
The force coefficient is given as:

$$C_F = \frac{|\mathbf{F}|}{\frac{1}{2}\rho A_{ref} U_{avg}^2}$$

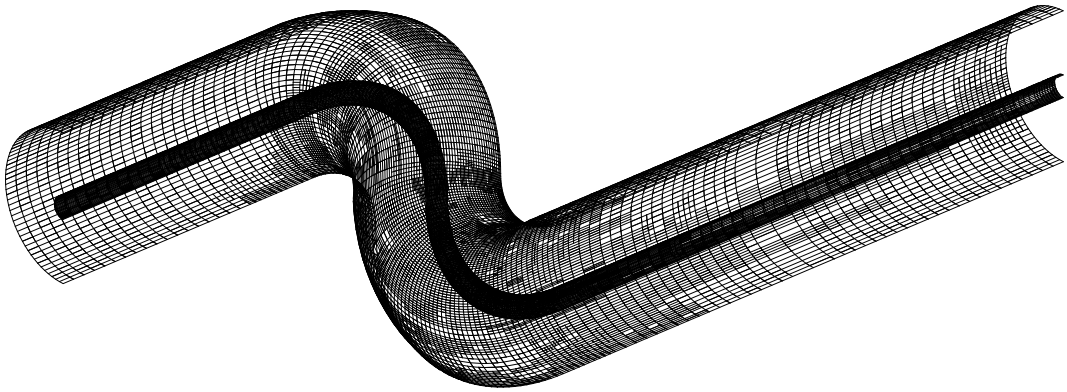
where $|\mathbf{F}|$ is the magnitude of the total force and $A_{ref} = \frac{\left((0.5D)^2 - (0.075D)^2\right)\Pi}{2}$, see Fig. 6.9(b). The total force is calculated by integrating the pressure over all boundaries and the shear force over the walls. The vorticity $\omega = curl\,\mathbf{U}$ is given in a normalised form:

$$C_\omega = \frac{|\omega| D}{U_{avg}}.$$

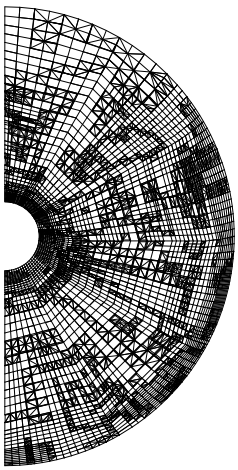
The values of C_p, C_F and C_ω are given in Table 6.3. The values on the final adapted mesh are in good agreement with the benchmark solution, the pressure drop differs less than 0.2 % from the benchmark solution, the force magnitude matches within 0.3% and the vorticity magnitude matches within 1.6 % with the benchmark solution but the final adapted mesh has 8% of the number of cells in the benchmark mesh. They have not changed much during the last five cycles because the mean estimated error exhibits slow reduction, Fig. 6.14(b), suggesting that the faces with highest estimated errors do not have a large influence on the overall accuracy.



(a) Mesh in the symmetry plane

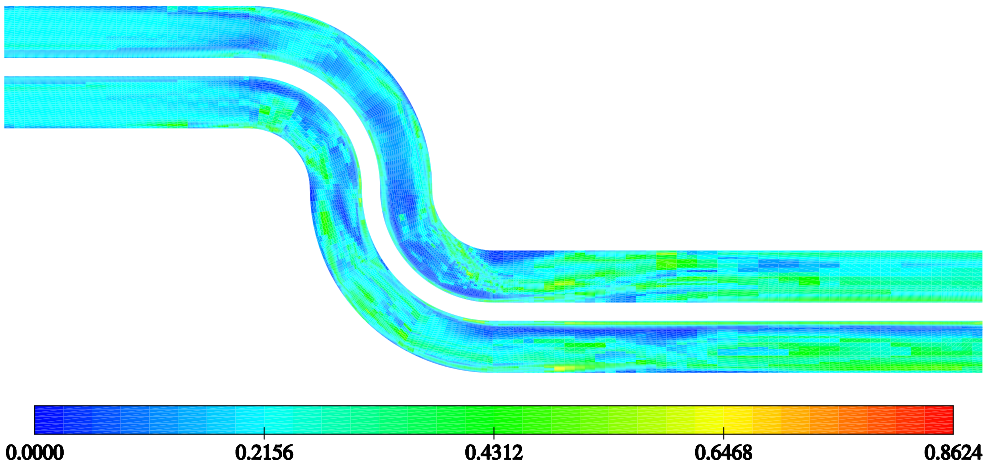


(b) Mesh at wall boundaries

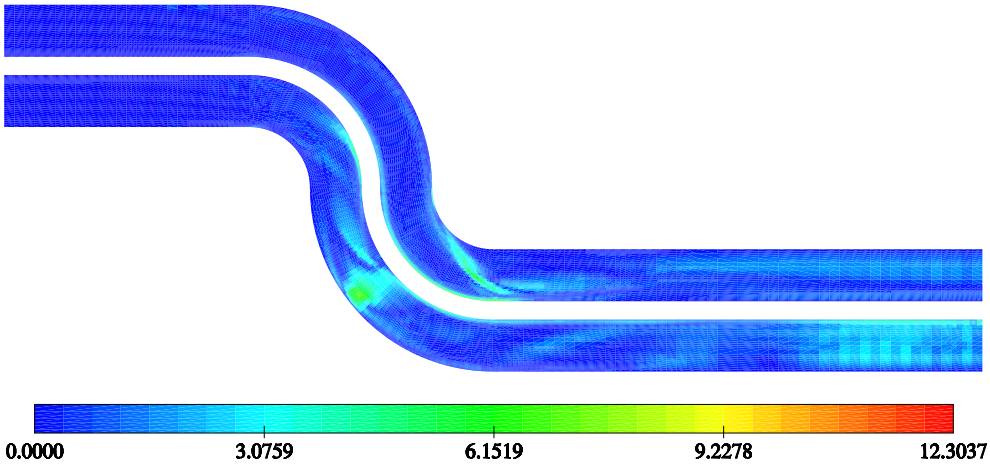


(c) Mesh at section $X = 2D$ (EnSight visualisation package decomposes intersections between split-hexahedra and the plane into triangles)

Figure 6.15: Mesh after 9 cycles of refinement for the S-bend case (390787 cells)



(a) Estimated error



(b) "Exact error"

Figure 6.16: Velocity error in the symmetry plane after 9 cycles of refinement for the S-bend case (390787 cells) (given as percentage of U_{max})

Adaptive refinement			
Cells	C_p	C_F	C_ω
270	1.861	2.187	5.37
2160	3.058	3.583	7.33
17280	3.504	3.967	8.24
119174	3.547	4.042	8.77
271810	3.536	4.048	8.91
343374	3.532	4.045	8.93
373203	3.533	4.045	8.93
388673	3.533	4.046	8.93
390787	3.533	4.046	8.94
Uniform refinement			
Cells	C_p	C_F	C_ω
1250	2.885	3.259	7.26
10000	3.47	3.948	8.09
80000	3.55	4.035	8.74
640000	3.54	4.052	8.97
5.12e+06	3.539	4.057	9.08

Table 6.3: Pressure-drop coefficients C_p, force coefficients C_F and average vorticity coefficient C_ω for the S-bend case

6.4 Tube Bundle

The flow across a bank of staggered tubes is an example of turbulent flow calculated using the $q-\zeta$ turbulence model developed by Gibson *et al.* [54]. This case is a part of the ERCOFTAC database [44]. The geometry of the computational domain and the boundary conditions employed here are shown in Fig. 6.17. Even though the flow is periodic, see Fig. 6.19(a), periodic boundary conditions were not employed for the inlet and the outlet boundary. This is because the refinement procedure cannot guarantee that the adapted mesh will have same distribution of faces at the left and the right boundary which is required by the flow solver.

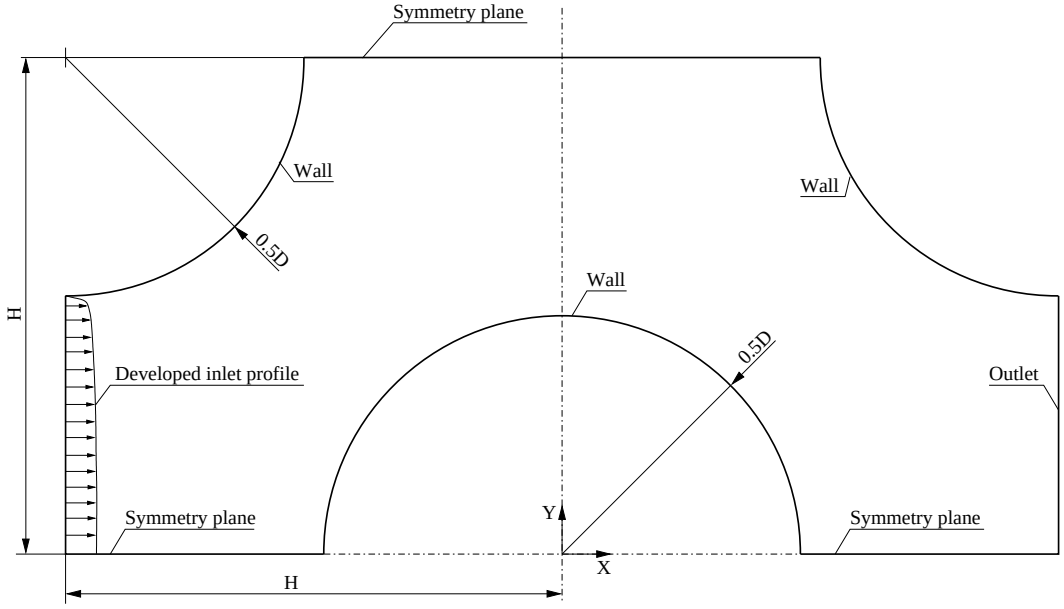


Figure 6.17: Geometry and boundary conditions for the tube bundle case

The ratio between H and D is:

$$\frac{H}{D} = 1.0369.$$

The Reynolds number of the flow is $Re = 18000$ based on the tube diameter D and the average inlet velocity U_{avg}. A developed inlet profile is obtained by fitting a cubic spline through the data obtained by a separate calculation using periodic boundary conditions. The data sets for velocity, q and ζ are taken at the location of the inlet boundary while the data set for the pressure field is taken at the location of the outlet boundary.

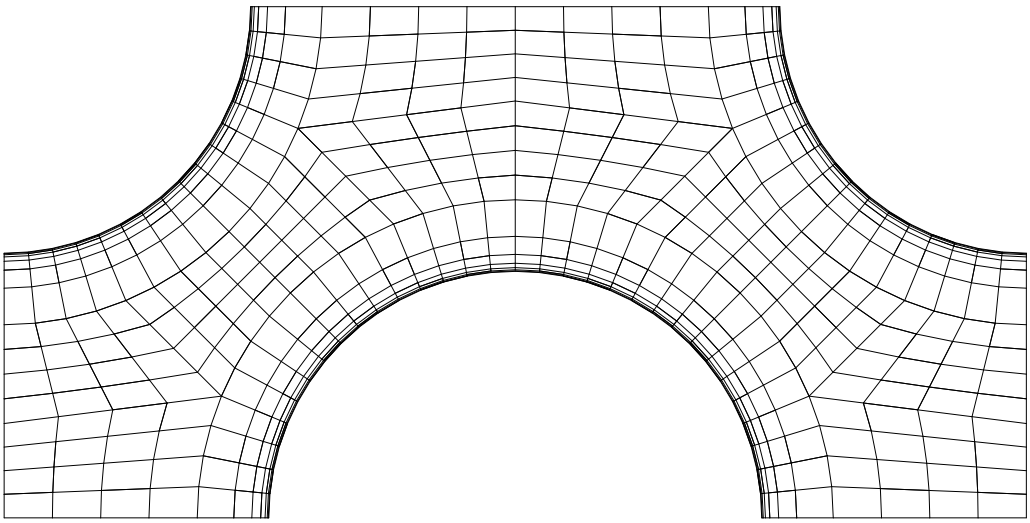


Figure 6.18: Starting mesh for the tube bundle case (640 cells)

The calculation was started on the coarse mesh with 640 cells, shown in Fig. 6.18. Second-order Central Differencing was used for approximations of convective terms for all fields. Gradients were evaluated by using the Gauss Theorem, Eqn. (3.26).

Figs. 6.19(a) and 6.19(b) show the flow field for this problem. Fig. 6.20 shows the pressure field which is given as a pressure coefficient defined as:

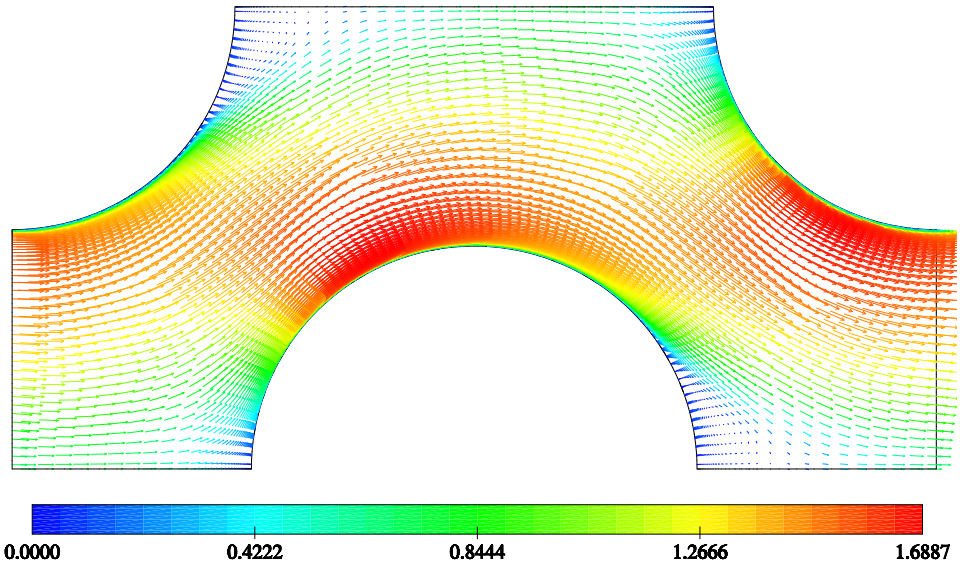
$$C_p = \frac{P - P_{atm}}{0.5\,\rho\,U_{avg}^2} \tag{6.1}$$

where P_{atm} is the atmospheric pressure. The flow is periodic with high gradients of velocity, q and ζ present near the tube walls, as shown in Figs. 6.19(a), 6.21 and 6.22. Separations on the downstream sides of the tubes are a consequence of the adverse pressure gradients on the leeward sides which slow the fluid down and eventually cause separation, Fig. 6.20. The turbulence variables q and ζ vary strongly in the near-wall region, while their variation in most parts of the domain is slow, see Figs. 6.21 and 6.22.

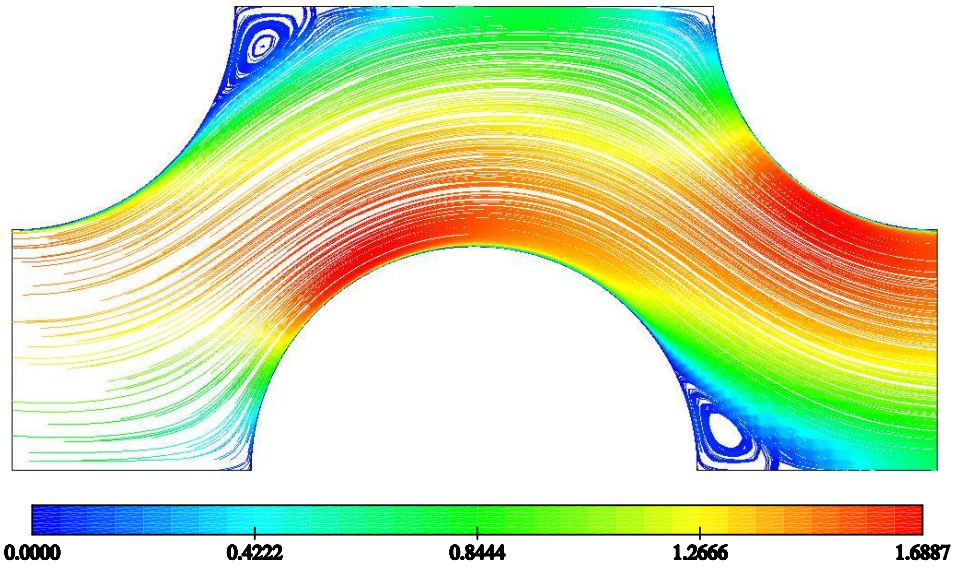
The required accuracy of the adapted solution was:

$$E = 1\,\%$$

for velocity, q and ζ fields based on the maximums $U_{max} = 1.689U_{avg}$, $q_{max} = 0.599U_{avg}$ and $\zeta_{max} = 5.83\frac{U_{avg}^2}{D}$, respectively. The accuracy of the adapted solution is compared with the solution on a mesh with 486000 cells considered to be the benchmark. The error in the benchmark is estimated to 0.2%, 1.3% and 0.51%



(a) Velocity field



(b) Streamlines

Figure 6.19: Velocity field for the tube bundle case (17505 cells)(given as $\frac{\mathbf{U}}{U_{avg}}$)

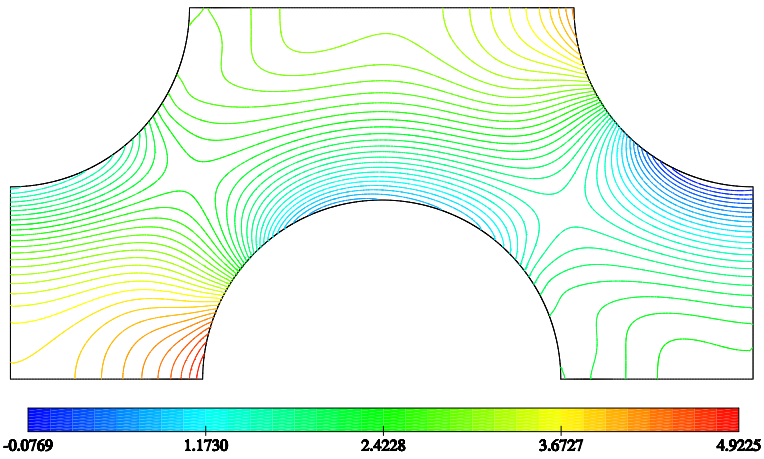


Figure 6.20: Pressure coefficient for the tube bundle case (17505 cells)

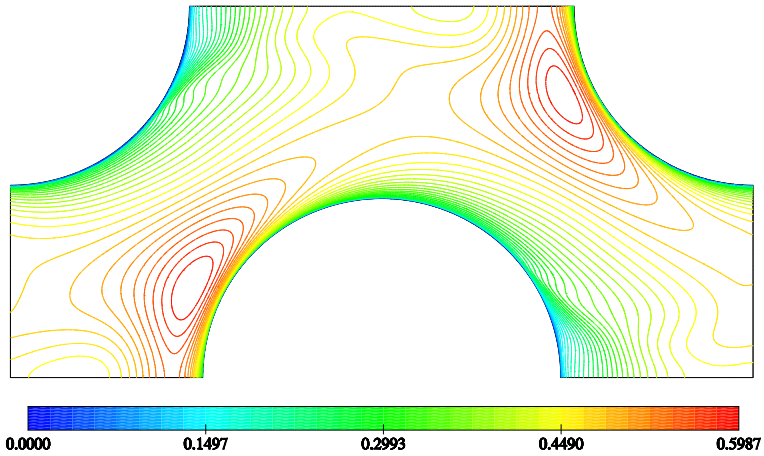


Figure 6.21: q field for the tube bundle case (17505 cells)(given as $\frac{q}{U_{avg}}$)

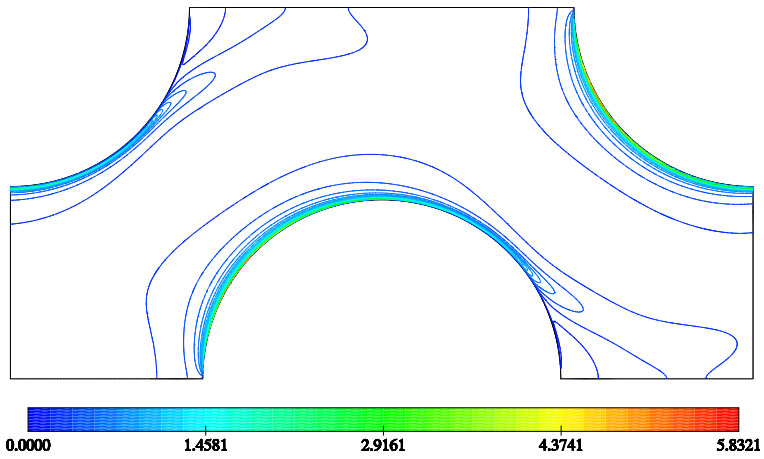


Figure 6.22: ζ field for the tube bundle case (17505 cells)(given as $\frac{\zeta D}{Uavg^2}$)

for the velocity, q and ζ fields, respectively, using the FREE. The solution is also compared with the measured data of Simonin and Barcouda [44].

The final adapted mesh has 17505 cells and is highly refined near the walls in the direction parallel to the wall, see Figs. 6.23(a) and 6.23(b), which is the direction of steepest variations of the velocity, q and ζ fields. The regions of fine mesh downstream of the tubes were needed to resolve the shear layers near the separation zones. The refined mesh regions also show periodic behaviour of the flow, except near the outlet because of the zero gradient boundary condition imposed on the velocity, q and ζ fields is not accurate on the coarse mesh.

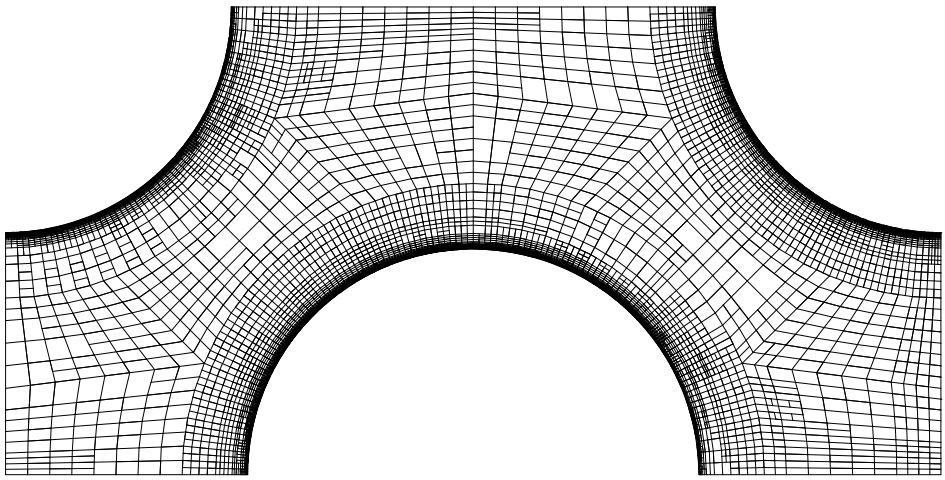
Figs. 6.24(a), 6.24(c) and 6.24(e) show good agreement between the estimated and exact maximum errors for the first five cycles of refinement. It is believed the under-estimation of these errors during the last cycles of refinement, see Figs. 6.24(a), 6.24(c), 6.24(e), 6.24(b) and 6.24(d), is caused by the fact that the errors on the adapted mesh become comparable to the error in the benchmark solution. The ζ error is over-estimated in the near-wall region dominated by diffusion for which the estimator may predict higher errors because its truncation error, Eqn. (4.45), is not consistent with the truncation error for the diffusion term Eqn. (3.56).

The agreement between the exact-error fields shown in Figs. 6.25(a), 6.25(c) and 6.25(e) and the estimated-error fields presented in Figs. 6.25(b), 6.25(d) and 6.25(f) is poor and the reasons for that are threefold:

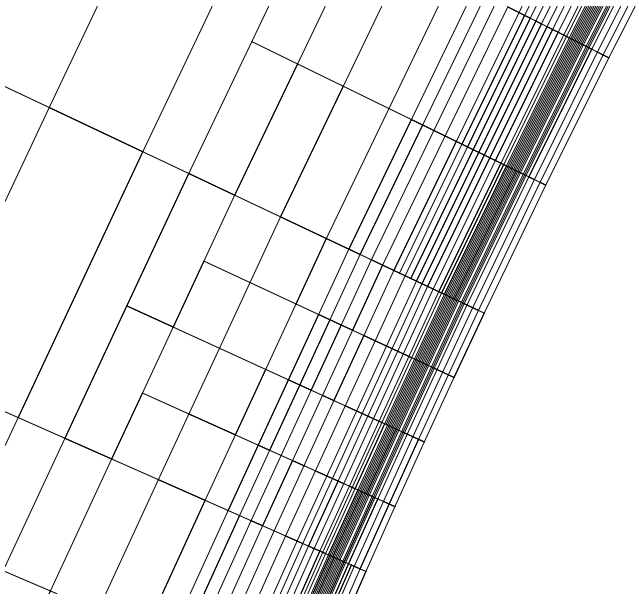
1. **The FREE** which tends to over-estimate the error on faces at which diffusion is dominant. This is the reason why the estimated error in velocity, Fig. 6.25(b), is higher in the bulk of the domain than the exact error, Fig. 6.25(a).

2. **Face error averaging**, Eqn. (4.48), used for plotting the errors. For example, the maximum of the estimated velocity-error field is located in the boundary layer, as expected, but the Fig. 6.25(b) shows that the error is highest in the stagnation zones. This happens because the size of the cell face determines its weighting factor in Eqn. (4.48).

3. **Benchmark solution** which is not exact. It is believed that the main reason for under-estimation of the error in the q field, Figs. 6.25(c) and 6.25(d), is the remaining error in the benchmark solution because the estimated error in the benchmark solution is highest there.

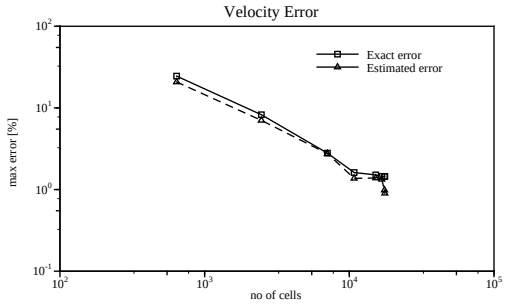


(a) Final adapted mesh

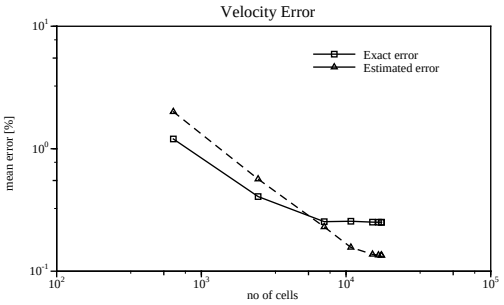


(b) Detail near the bottom tube wall

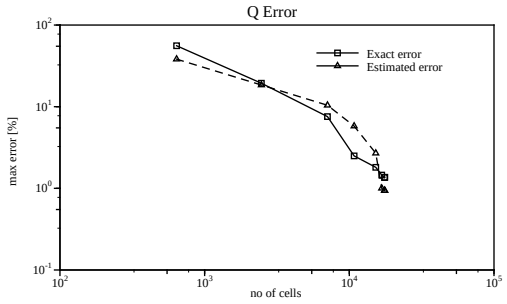
Figure 6.23: Mesh after 7 cycles of refinement for the tube bundle case (17505 cells)



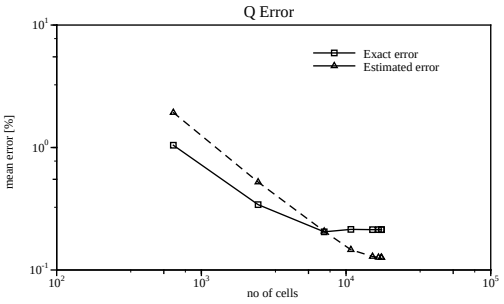
(a) Maximum-velocity errors



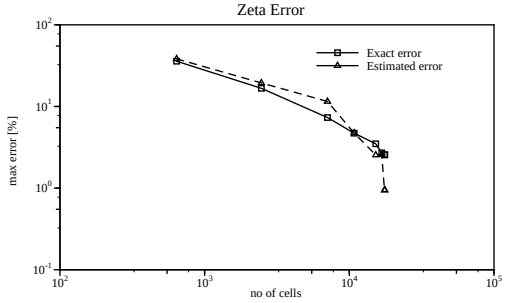
(b) Mean-velocity errors



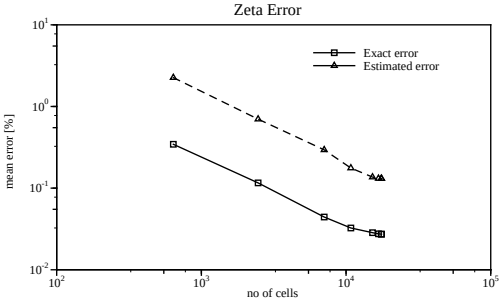
(c) Maximum-q errors



(d) Mean-q errors

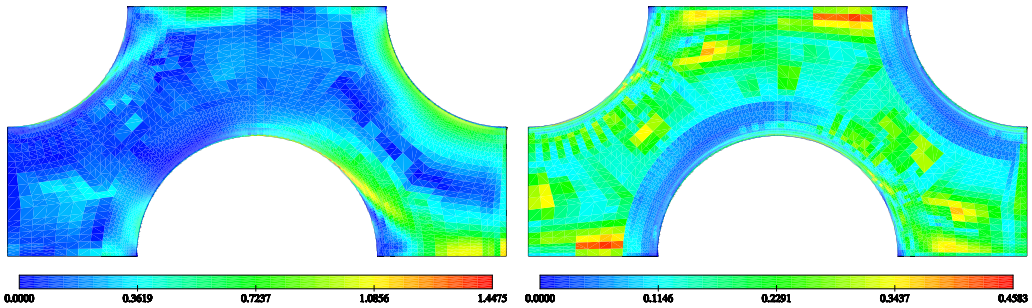


(e) Maximum-ζ errors

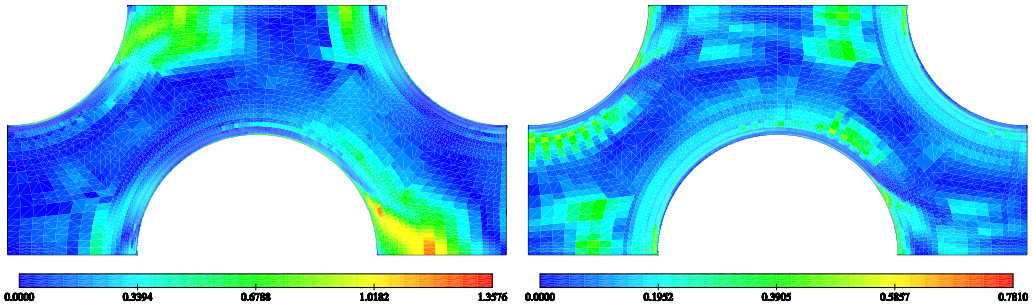


(f) Mean-ζ errors

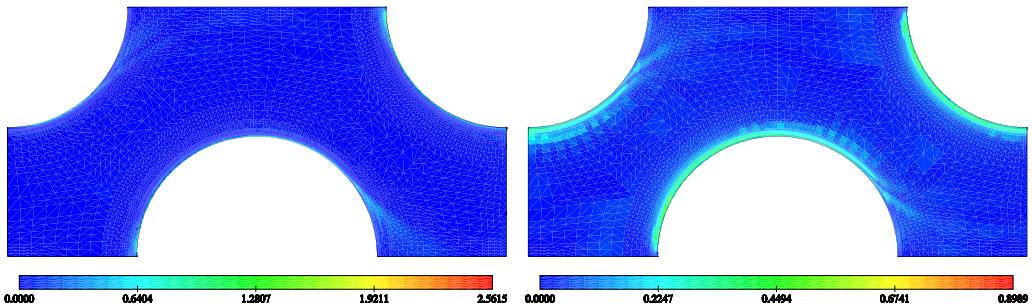
Figure 6.24: Variation of errors with adaptive mesh refinement for the tube bundle case (errors given as percentage of U_{max}, q_{max} and ζ_{max} respectively)



(a) Exact-velocity error (b) Estimated-velocity error

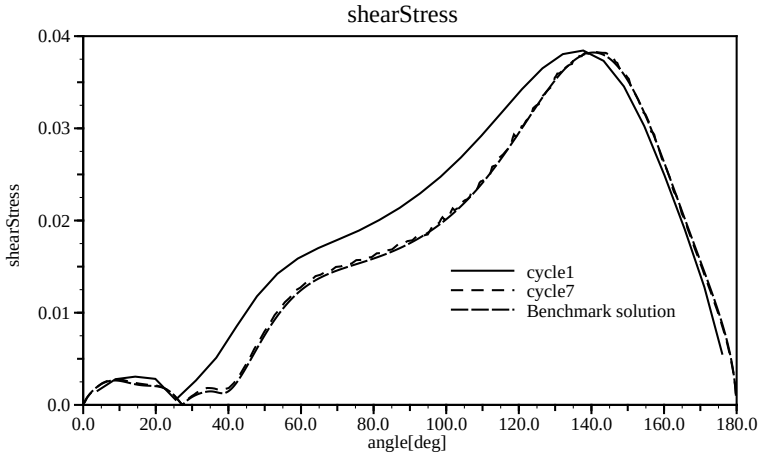


(c) Exact-q error (d) Estimated-q error

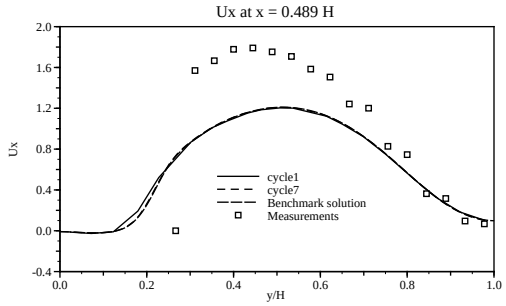


(e) Exact-ζ error (f) Estimated-ζ error

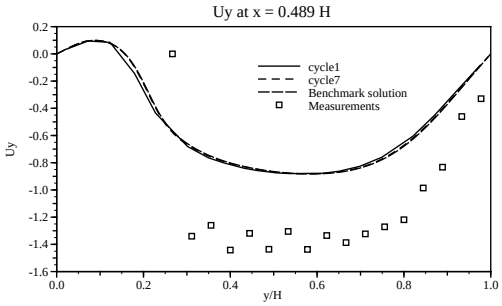
Figure 6.25: Exact and estimated-error fields after 7 cycles of refinement (17505 cells)(errors are given as percentage of U_{max}, q_{max} and ζ_{max} respectively). Exact errors are calculated as the difference from the benchmark solution. Estimated errors are plotted as a weighted average of face errors.



(a) Shear stress at bottom tube wall (given as $\frac{\tau_w}{\rho U_{avg}^2}$)



(b) U_x profiles at $x = 0.489H$ (U_x normalised by U_{avg})



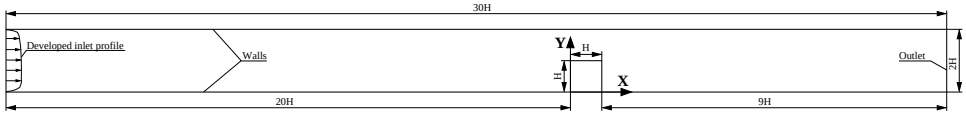
(c) U_y profiles at $x = 0.489H$ (U_y normalised by U_{avg})

Figure 6.26: A comparison of profiles for the tube bundle case

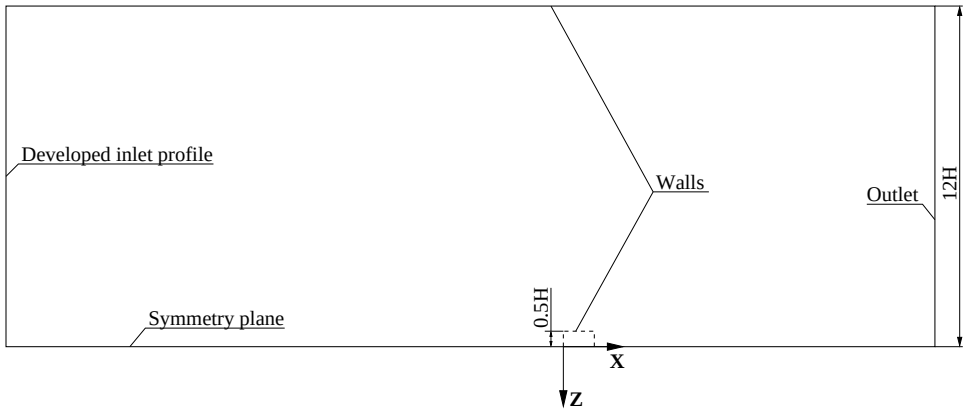
Fig. 6.26(a) shows the shear stress at the bottom tube wall which is in a very good agreement with the benchmark solution even though the adapted mesh has only 3.6% of the number of cells of the benchmark mesh. Figs. 6.26(b) and 6.26(c) show that velocity profiles after 7 cycles of refinement match very well with the benchmark solution but the agreement with the measured data by Simonin and Barcouda [44] is poor. The main cause of this is the $q - \zeta$ turbulence model, which tends, in common with most two-equation models, to over-predict k and/or q before stagnation zones.

6.5 Wall-Mounted Cube

The turbulent flow around a cubical obstacle mounted on a wall in a channel was chosen for testing the mesh-refinement procedure for calculations using the Standard k-ϵ turbulence model with Wall-functions [124]. Even though the geometry is simple, see Figs. 6.27(a) and 6.27(b), the flow around the cube is very complex, consisting of many vortices, a stagnation point, a recirculation zone, a re-attachment point, *etc.* [97], see Figs. 6.29(b), 6.30(b) and 6.31(b). It therefore imposes a serious challenge for the turbulence model and the refinement procedure to detect and predict all important flow features correctly. The objective of this exercise was to examine if the mesh-refinement procedure is capable of producing a mesh-independent solution for such a case.



(a) Side view



(b) Top view

Figure 6.27: Geometry and boundary conditions for the wall mounted cube case

A computational study involving tests of different turbulence models on this flow was performed by Lakehal *et al.* [83]. Muzaferija [102] performed a mesh-refinement study for a different channel geometry than in the present case. His final mesh consists of 186000 cells, which was not enough to produce a mesh-independent solution. The majority of refinement occurred near the cube and in the boundary

layers.

The geometry used for the present calculation is the one presented by Martinuzzi *et al.* [97] which is a part of the ERCOFTAC test-case library [44].

The computational domain and the boundary conditions are shown in Figs. 6.27(a) and 6.27(b). The geometry presented in [97] is symmetric so only one half is used for calculations. A fully developed channel flow was simulated first in a separate calculation and then interpolated onto the inlet using a cubic interpolation [105], which guarantees smooth profiles of velocity, k and ϵ at the inlet. The profile was interpolated after every cycle of refinement if the mesh at the inlet changed.

The Reynolds number is $Re = 40000$ based on the cube height H and the average inlet velocity U_{avg}.

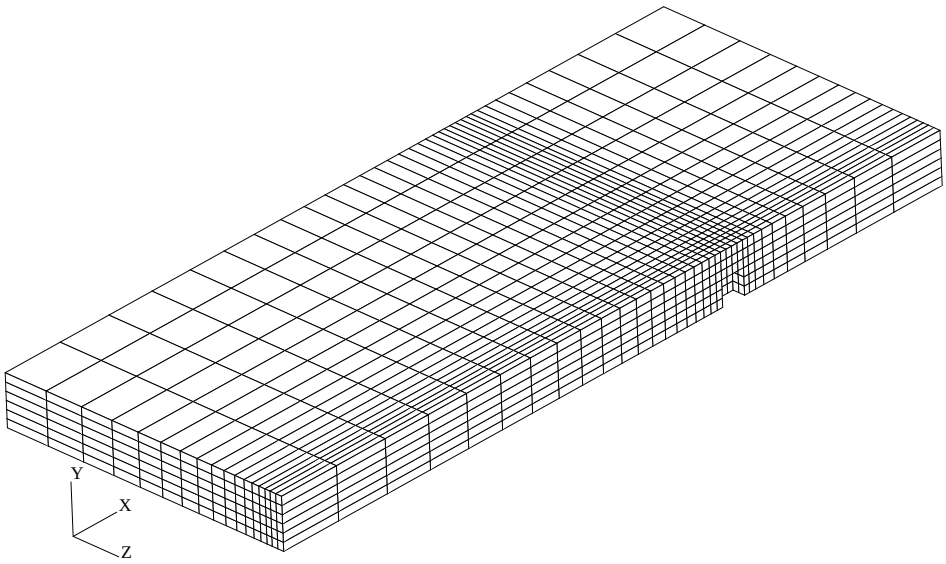
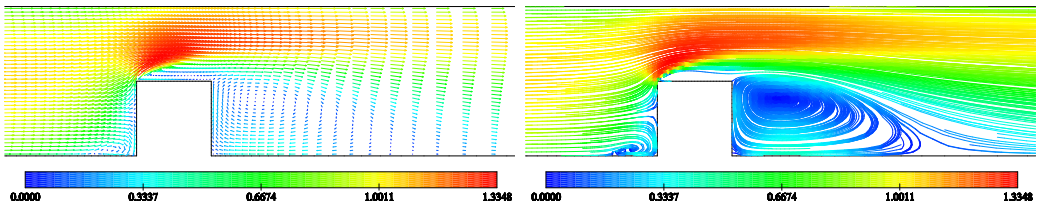


Figure 6.28: Starting mesh for the wall-mounted cube case (3444 cells)

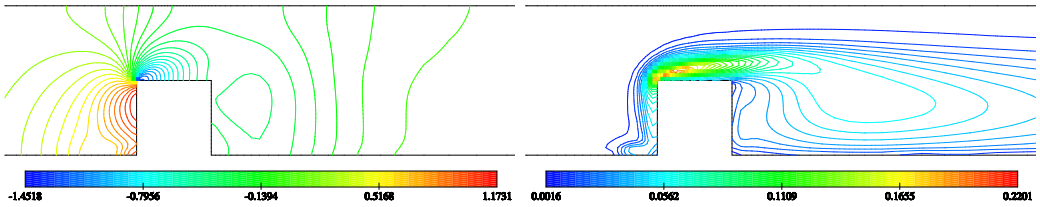
The starting mesh for the adaptive calculation consists of 3444 cells and is shown in Fig. 6.28. The Gamma scheme, described in Section 3.3.1, with the factor $\gamma = 0.5$ was used for the discretisation of convective terms for velocity, turbulent kinetic energy k and turbulent dissipation ϵ. All gradients are calculated using the Gauss-divergence Theorem.

The velocity field in the symmetry plane is shown in Figs. 6.29(a) and 6.29(b) where three vortices can be seen. The vortex in front of the cube is the start of a horse-shoe vortex. Those on the top and rear are caused by the adverse pressure



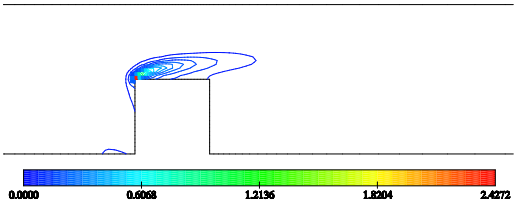
(a) Velocity field (given as $\frac{\mathbf{U}}{U_{avg}}$)

(b) Streamlines



(c) Pressure coefficient

(d) Turbulent kinetic energy (given as $\frac{k}{U_{avg}^2}$)



(e) Turbulent dissipation (given as $\frac{\epsilon H}{U_{avg}^3}$)

Figure 6.29: Distribution of flow variables in the symmetry plane for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells)

gradient. The pressure field is shown in Fig. 6.29(c) and is given as:

$$C_p = \frac{P - P_{atm}}{0.5\rho U_{avg}^2} \tag{6.2}$$

where P_{atm} is the atmospheric pressure applied at the outlet boundary. The velocity gradient is large near the top leading edge and it is the reason why the turbulent kinetic energy, Fig. 6.29(d), and turbulent dissipation, Fig. 6.29(e), are high there.

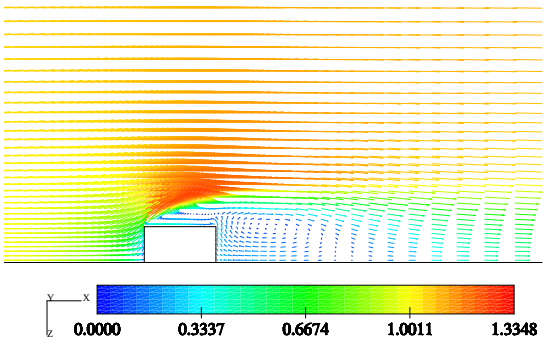
The velocity field in the plane $y/H = 0.5$ is presented in Figs. 6.30(a) and 6.30(b) which show a vortex beside the cube and the arch vortex downstream of the cube. The side vortex is again a consequence of the adverse pressure gradient, Fig. 6.30(c) while the high turbulent kinetic energy, Fig. 6.30(d), and turbulent dissipation, Fig. 6.30(e), are a result of sharp velocity gradients near the leading edge.

The section at $x/H = 0.5$ shows the separations on top and beside the cube, Figs. 6.31(a) and 6.31(b). An additional feature which can be seen near the bottom wall is the horse-shoe vortex. The pressure field does not exhibit high gradients there and it turns the fluid towards the cube. The turbulent energy, Fig. 6.31(d), and the turbulent dissipation, Fig. 6.31(e), are highest in the regions of highest shear stresses, Fig. 6.31(a).

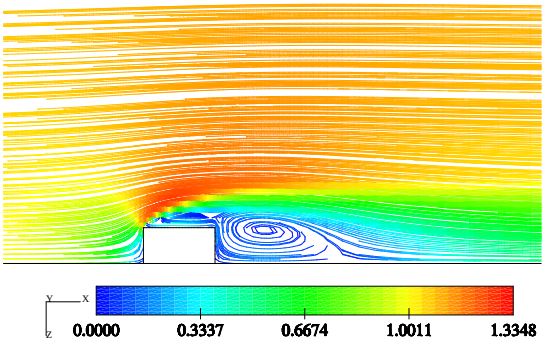
Fig. 6.32 shows the fully-developed horse-shoe vortex in the plane $x/H = 2$. The structure left of the horse-shoe vortex is the arch vortex behind the cube where the velocity gradient is high in the z direction.

The adaptive procedure was set to reduce the error below 1% for velocity based on the average inlet velocity U_{avg}. The tolerances for k and ϵ fields were 5% and 10%, respectively, based on its maximum values found at the inlet. The accuracy of the solution on the adapted mesh will be assessed by comparing it with experiments performed by Martinuzzi *et al.* [97] and with the CFD results obtained by Lakehal *et al.* [83].

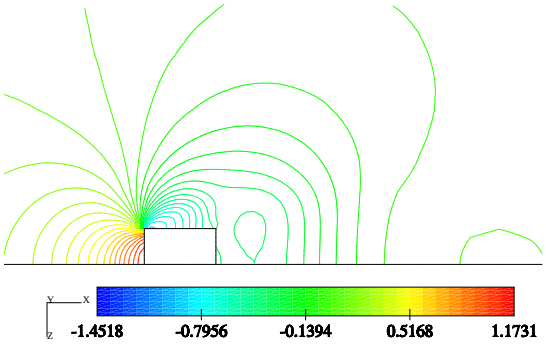
Fig. 6.33(a) shows the mesh after 7 cycles of refinement (1161490 cells) where three main regions of refinement can be recognised. Most refinement occurred near the cube (region A) where the flow is most complex and where the highest gradients reside. Figs. 6.33(b), 6.33(c) and 6.33(d) show that the finest mesh is present near the leading edges of the cube which are the regions of highest shear. The refinement in front of the cube occurred to resolve the separation. The refinement behind the cube (region C) occurred to resolve the arch vortex. Fig. 6.33(d) shows that the mesh is also refined to resolve the horse-shoe vortex downstream of the cube. Significant



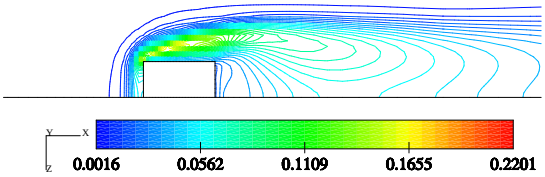
(a) Velocity field (given as $\frac{\mathbf{U}}{U_{avg}}$)



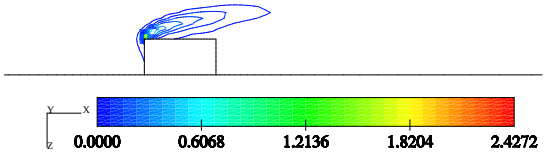
(b) Streamlines



(c) Pressure coefficient

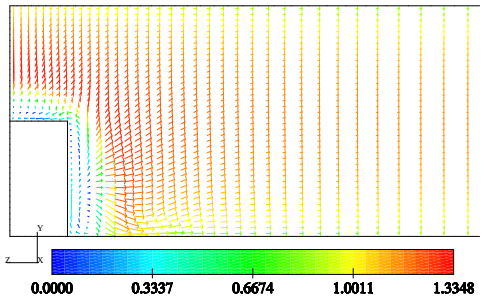


(d) Turbulent kinetic energy (given as $\frac{k}{U_{avg}^2}$)

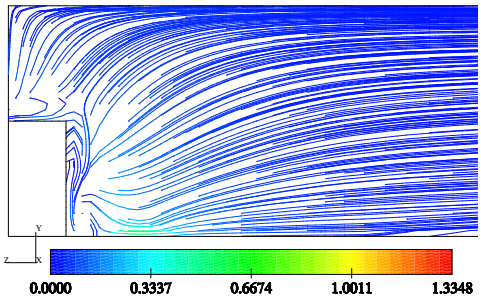


(e) Turbulent dissipation (given as $\frac{\epsilon H}{U_{avg}^3}$)

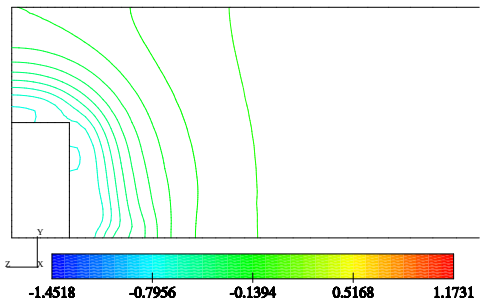
Figure 6.30: Distribution of flow variables in the plane $y/H = 0.5$ for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells)



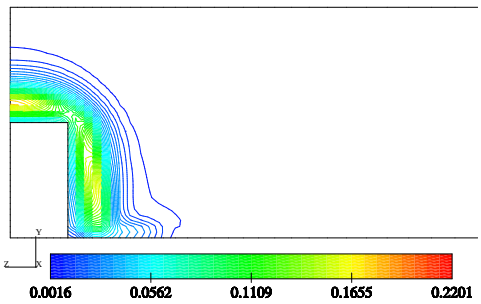
(a) Velocity field (given as $\frac{\mathbf{U}}{U_{avg}}$)



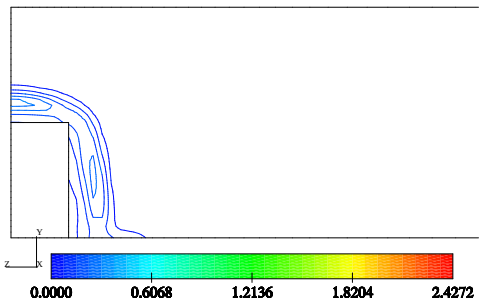
(b) Streamlines



(c) Pressure coefficient



(d) Turbulent kinetic energy (given as $\frac{k}{U_{avg}^2}$)



(e) Turbulent dissipation (given as $\frac{\epsilon H}{U_{avg}^3}$)

Figure 6.31: Distribution of flow variables in the plane $x/H = 0.5$ for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells)

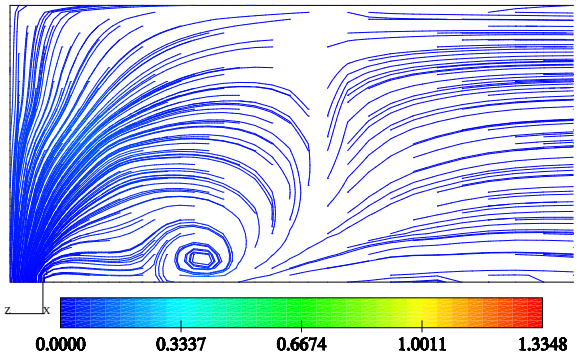


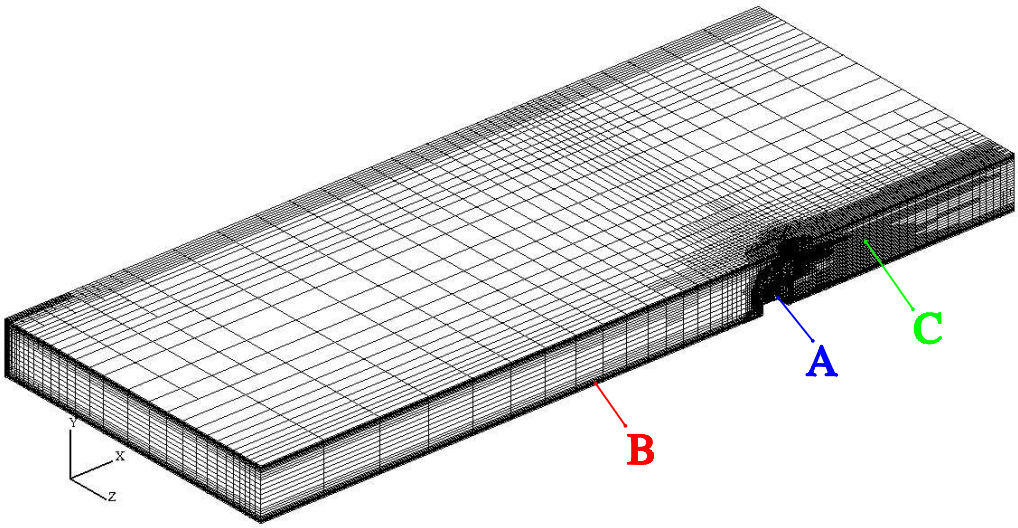
Figure 6.32: Streamlines in the plane $x/H = 2$ for the wall-mounted cube case, obtained on the final adapted mesh (1161490 cells)

refinement also occurred in the near-wall region (region B) to resolve the boundary layers properly. Refinement in the near-wall region was stopped when the value of Y^+ dropped below 50, see Fig. 6.33(b), but the cells on which that restriction does not apply were still refined.

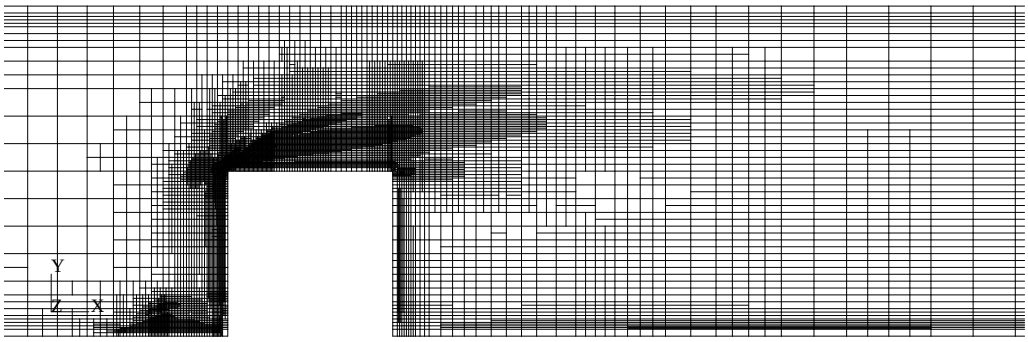
No of cells	$\frac{\epsilon H}{U_{avg}^3}$	$\frac{\lvert\nabla \mathbf{U}\rvert H}{U_{avg}}$	$\frac{\lvert\nabla k\rvert H}{U_{avg}^2}$	$\frac{\lvert\nabla p\rvert H}{0.5\rho U_{avg}^2}$
3444	0.0195506	11.1664	0.0911675	3.11114
12588	0.0936529	22.6795	0.383472	7.58207
45284	0.354801	43.5974	0.882812	16.1717
150095	1.06293	86.7659	1.78578	32.6592
433616	2.86874	176.786	4.35404	62.4194
850243	6.82605	352.721	10.2637	129.418
1161490	7.65171	437.867	18.2246	231.578

Table 6.4: Maximum of ϵ, velocity gradient, k field gradient and pressure gradient fields

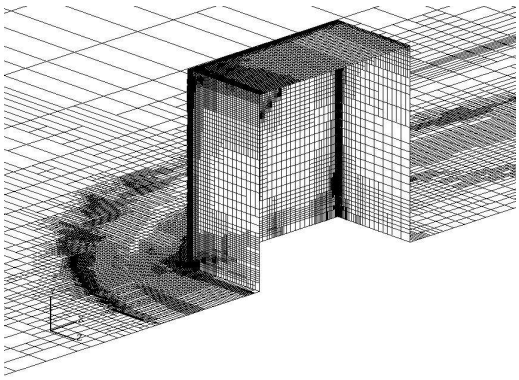
Figs. 6.34(a), 6.34(c) and 6.34(e) show that the maximum estimated errors do not reduce with mesh refinement. They are located near the leading cube edges, Figs. 6.35(a), 6.35(b) and 6.35(c), which are the regions where the gradients of all fields are highest, Figs. 6.36(a) and 6.36(b), and exhibit singular behaviour, see Table 6.4, Figs. 6.37(a) and 6.37(b). This causes high over-estimation of the error for the velocity and k fields because singular gradients prevent the diffusion part of the face residual, Eqn. (4.43), from tending to zero. This happens because the



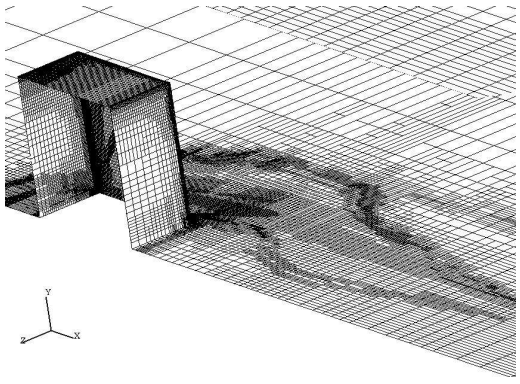
(a) Final mesh



(b) Symmetry plane

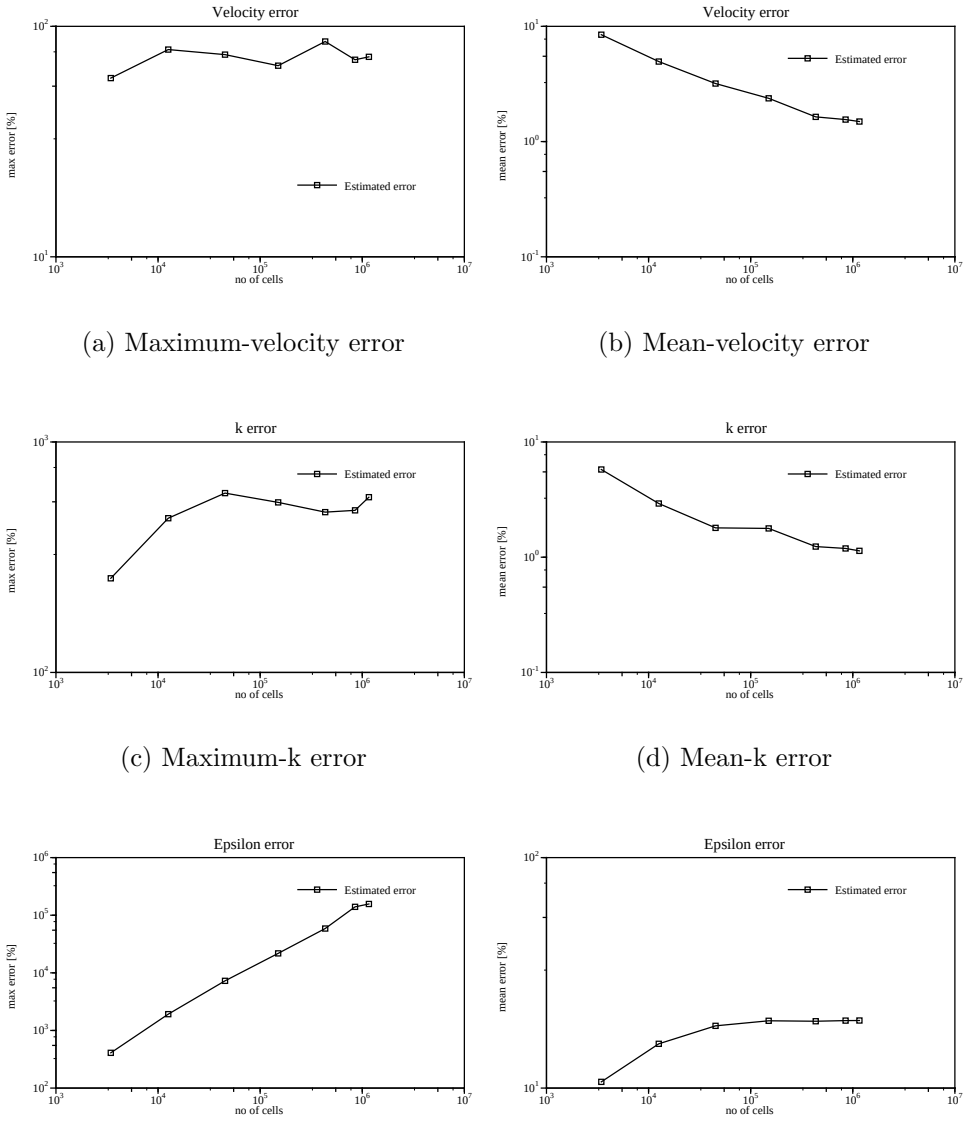


(c) Isometric view of mesh at the wall upstream of the cube



(d) View of the mesh at the wall downstream of the cube

Figure 6.33: Mesh after 7 cycles of refinement for the wall-mounted cube case (1161490 cells)



(a) Maximum-velocity error

(b) Mean-velocity error

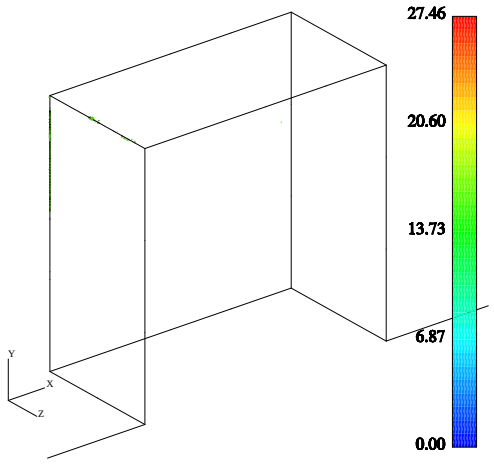
(c) Maximum-k error

(d) Mean-k error

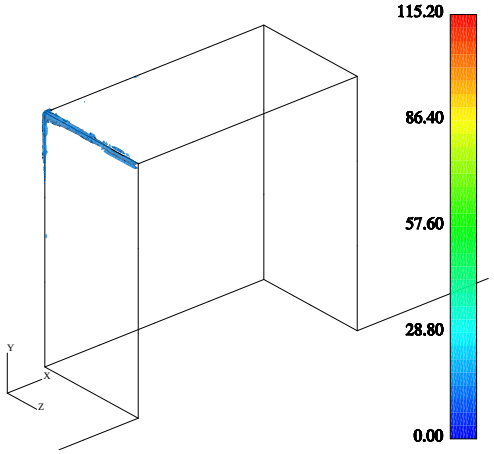
(e) Maximum-ϵ error

(f) Mean-ϵ error

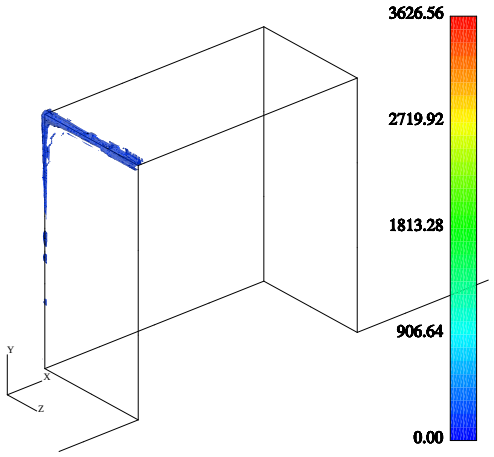
Figure 6.34: Variation of estimated errors with adaptive mesh refinement for the wall-mounted cube case (errors are given as percentage of U_{avg}, k_{max} at inlet and ϵ_{max} at inlet)



(a) Velocity error

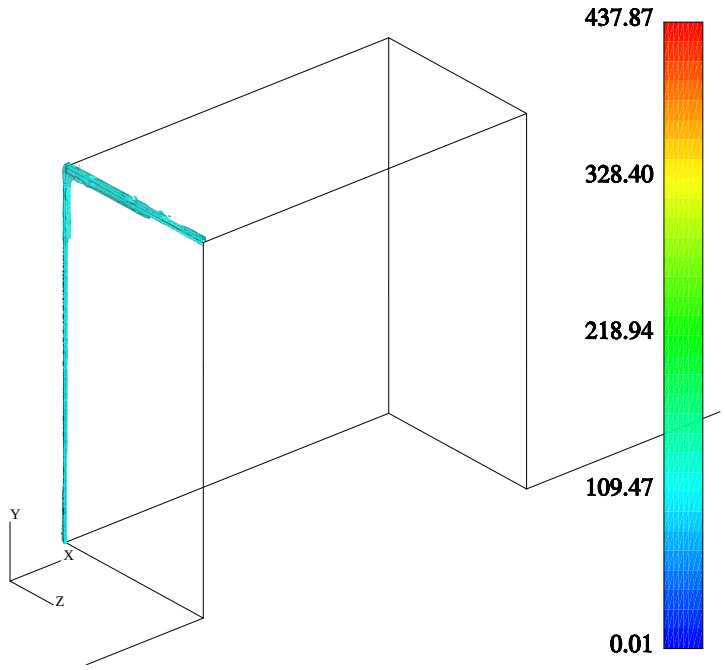


(b) k error

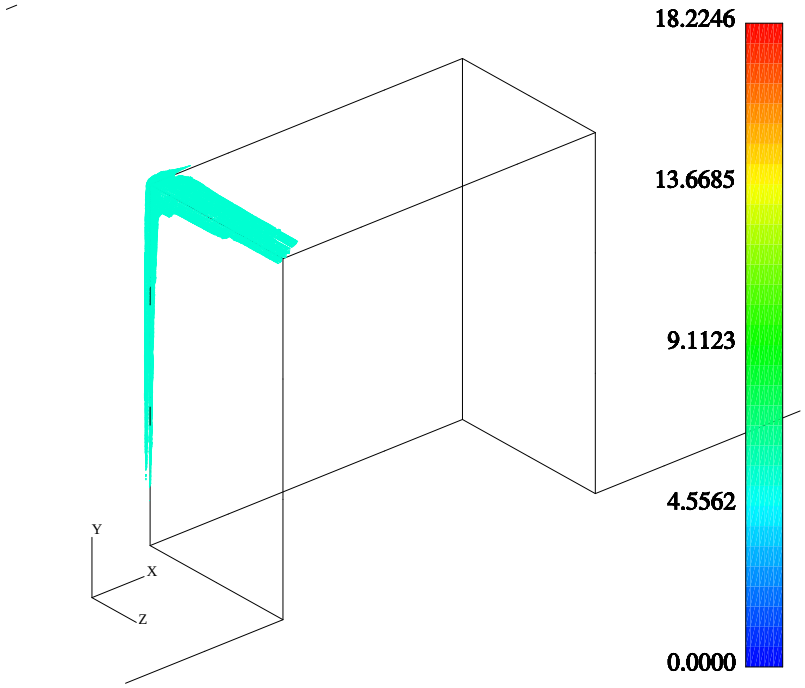


(c) ϵ error

Figure 6.35: Remaining estimated errors after 7 cycles of refinement (errors are given as percentage of U_{avg}, k_{max} at inlet and ϵ_{max} at inlet, respectively) Estimated errors are plotted as a weighted average of face errors.

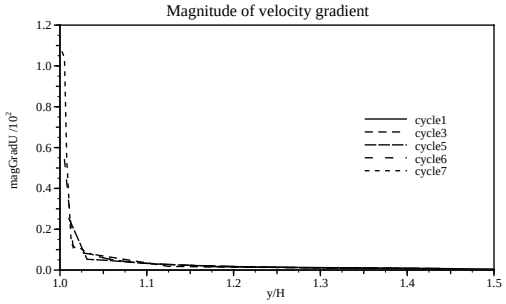


(a) Velocity gradient $\frac{|\nabla \mathbf{U}|H}{U_{avg}}$

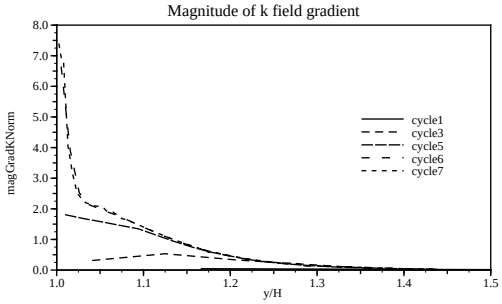


(b) k field gradient $\frac{|\nabla k|H}{U_{avg}^2}$

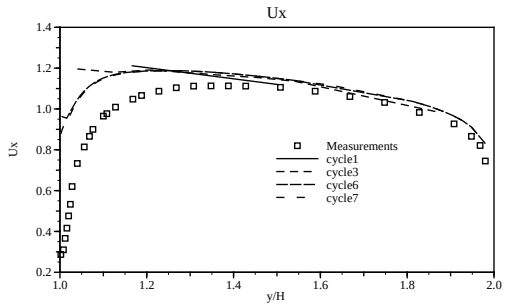
Figure 6.36: Regions of highest gradients after 7 cycles of refinement



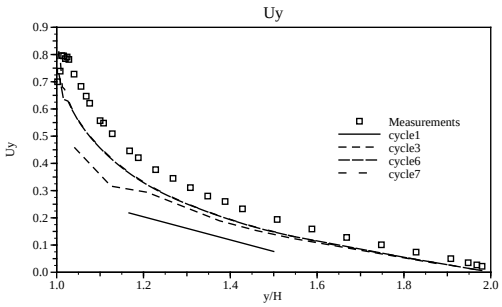
(a) Velocity gradient (given as $\frac{|\nabla \mathbf{U}|H}{U_{avg}}$)



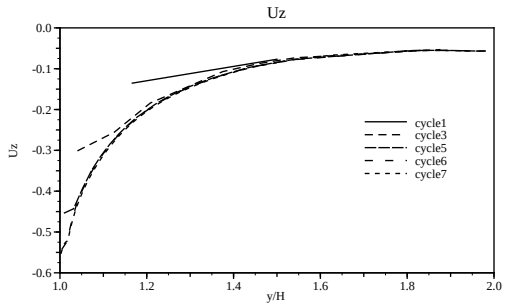
(b) k-field gradient (given as $\frac{|\nabla k|H}{U_{avg}^2}$)



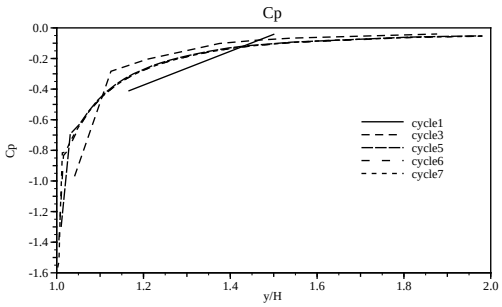
(c) U_x (given as $\frac{U_x}{U_{avg}}$)



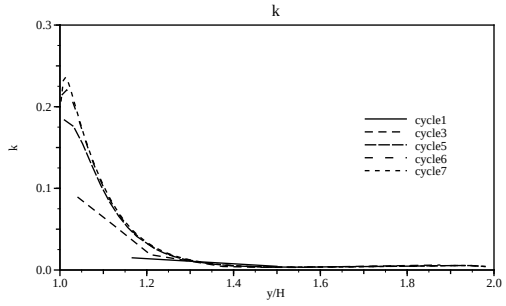
(d) U_y (given as $\frac{U_y}{U_{avg}}$)



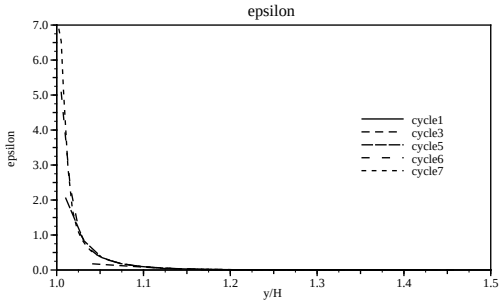
(e) U_z (given as $\frac{U_z}{U_{avg}}$)



(f) C_p (given as $\frac{P-P_{atm}}{0.5\rho U_{avg}^2}$)



(g) k (given as $\frac{k}{U_{avg}^2}$)



(h) ϵ (given as $\frac{\epsilon H}{U_{avg}^3}$)

Figure 6.37: Profiles of flow variables taken at $\frac{x}{H} = 0$, $\frac{z}{H} = -0.5$ (above the corner at which the leading edges meet)

difference between gradients in Eqn. (4.43) increases, while the $|\mathbf{x}_N - \mathbf{x}_P|$ term in Eqn. (4.35) decreases by almost the same factor thus making the estimated error not tend to zero. Figs. 6.37(c), 6.37(d) and 6.37(e) show that the velocity field tends to its mesh independent value near the leading corner of the cube even though the estimated error is highest there. The k field also tends to its mesh independent value there, see Fig. 6.37(g), even though the estimated error does not reduce with mesh refinement. The maximum estimated ϵ error, Fig. 6.34(e), increases with mesh refinement because the ϵ field exhibits singular behaviour near the leading cube edges, see Table 6.4 and Fig. 6.37(h). It is a consequence of Eqn. (3.80) which determines ϵ in the near wall cells. It seems that the production term P, see Eqn. (3.81), tends to infinity near the leading edges, because the velocity gradient, shown in Fig. 6.37(a), tends to infinity there. The singularity of P makes k reduce slower than the distance from the wall, which makes ϵ tend to infinity. Average estimated errors for the velocity and k fields reduce slowly during the last three cycles, see Figs. 6.34(b) and 6.34(d), because there still exist high errors in the boundary layers which cannot be further refined because of the Y^+ limit. Fig. 6.34(f) shows that the average-estimated-ϵ error increases with mesh refinement, and it is believed that the reason for this behaviour is the remaining error in the boundary layers and the singular behaviour of the ϵ field near the leading cube edges.

No of cells	C_F	$\frac{X_R}{H}$	$\frac{X_F}{H}$
3444	1.49	2.35	0.391815
12588	1.64	3.00	0.32
45284	1.83	2.65	0.5
150095	1.931	2.65	0.57
433616	1.95	2.65	0.67
850243	1.96	2.65	0.63
1.161490	1.97	2.65	0.63
Lakehal *et al.*		2.18	0.65
Measurements		1.61	1.04

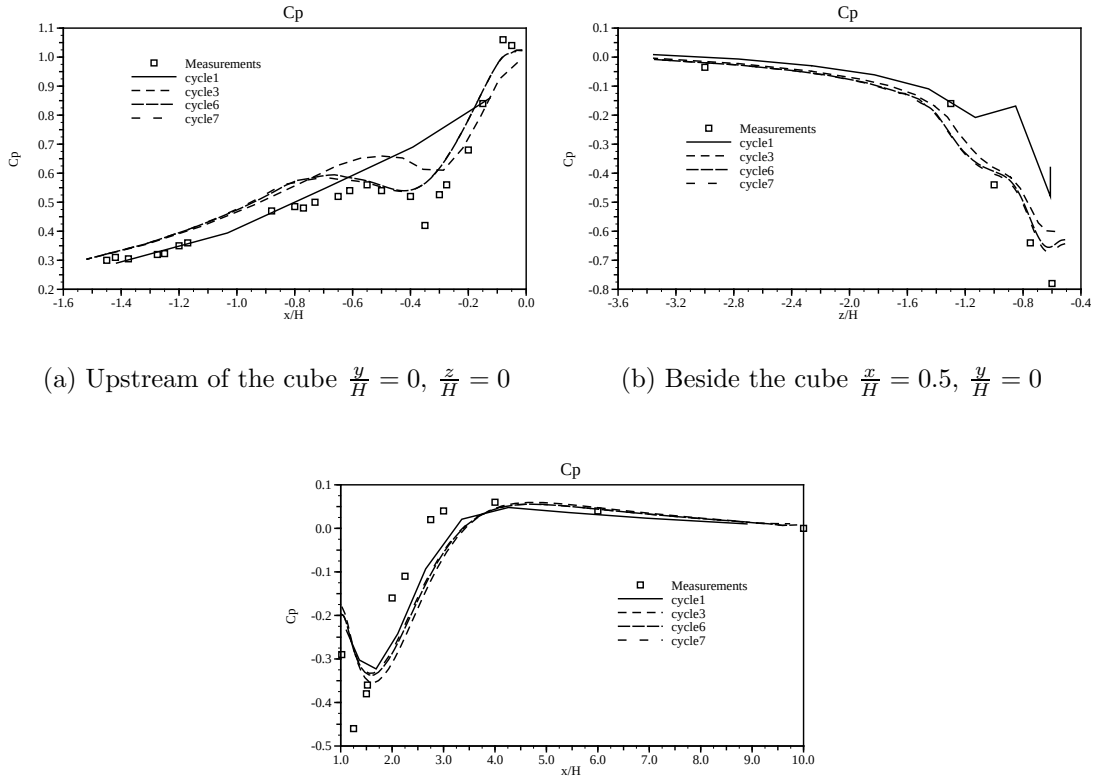
Table 6.5: Magnitude of the force acting on the cube and the lengths of vortices downstream and upstream of the cube

The magnitude of the force acting on the cube is another indicator of the achieved

accuracy which is monitored. Unfortunately, it is not given by Lakehal [83] nor Martinuzzi *et al.* [97]. The magnitude of the force, given in a form:

$$C_F = \frac{|\mathbf{F}|}{0.5\rho U_{avg}^2 \frac{H^2}{2}}$$

is shown in Table 6.5. It changes rapidly during the first four cycles of refinement while the mesh is still not fine enough to capture the separations around the cube accurately.



(a) Upstream of the cube $\frac{y}{H} = 0$, $\frac{z}{H} = 0$

(b) Beside the cube $\frac{x}{H} = 0.5$, $\frac{y}{H} = 0$

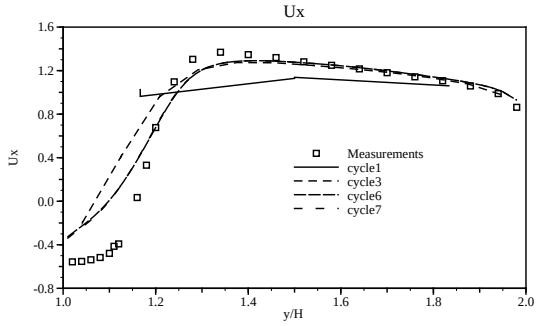
(c) Downstream of the cube $\frac{y}{H} = 0$, $\frac{z}{H} = 0$

Figure 6.38: C_p on the bottom wall

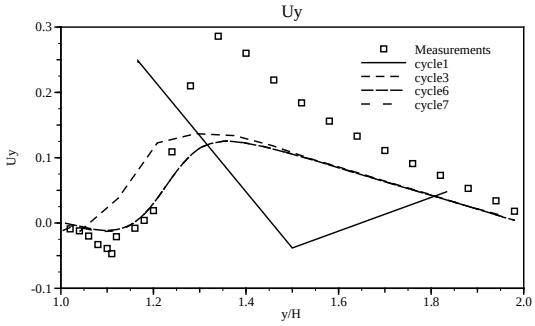
The predicted length of the separation region downstream of the cube, $\frac{X_R}{H}$ in Table 6.5, is significantly larger than the measured value by Martinuzzi *et al.* [97] which is $\frac{X_R}{H} = 1.61$. It is also larger than $\frac{X_R}{H} = 2.18$ as predicted by Lakehal *et al.* [83] by using the same turbulence model. This discrepancy suggests that the maximum error tolerances set for the adaptive calculation are too high, which results in the mesh not being fine enough to capture the separation accurately. Lakehal *et al.* [83] have not proven that their results are mesh-independent, which may also

be the cause of the above discrepancy. The length of the upstream separation, $\frac{X_F}{H}$ in Table 6.5, is in good agreement with the results of Lakehal *et al.* The final adapted mesh is finer than the mesh used by Lakehal and is the most probable cause of the small discrepancy in predicting the separation upstream of the cube.

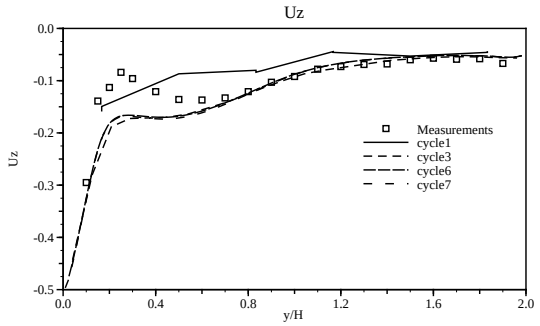
Figs. 6.38(a), 6.38(b) and 6.38(c) show the pressure profiles on the bottom wall near the cube. The differences between the profiles for the sixth and seventh cycle of adaptive refinement are small. The agreement between the CFD and the measured profiles improves with mesh refinement, especially beside the cube as shown in Fig. 6.38(b). It is poor in the regions where separations are wrongly predicted.



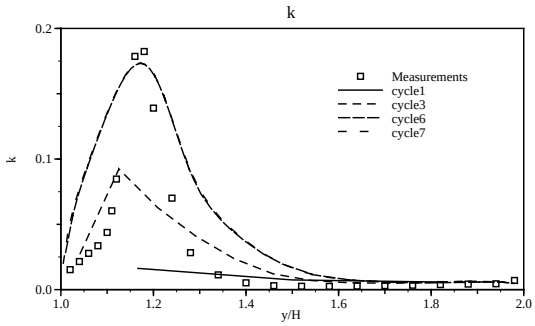
(a) x velocity component $\frac{x}{H} = 0.5$, $\frac{z}{H} = 0$ (above the centre of the top face) (given as $\frac{U_x}{U_{avg}}$)



(b) y velocity component $\frac{x}{H} = 0.5$, $\frac{z}{H} = 0$ (above the centre of the top face) (given as $\frac{U_y}{U_{avg}}$)



(c) z velocity component $\frac{x}{H} = 0.5$, $\frac{z}{H} = 1$ (given as $\frac{U_z}{U_{avg}}$)



(d) Turbulent energy k $\frac{x}{H} = 0.5$, $\frac{z}{H} = 0$ (above the centre of the top face) (given as $\frac{k}{U^2_{avg}}$)

Figure 6.39: Velocity and turbulent energy k profiles

Figs. 6.39(a) and 6.39(b) show that the velocity profiles above the centre of the top cube face do not change between the last two cycles of refinement. The velocity in the separation zone is under-predicted by the CFD calculation, which was also

the case in the calculations by Lakehal *et al.* [83]. The velocity in the z direction beside the cube, depicted in Fig. 6.39(c), is predicted well away from the cube but the horse-shoe vortex is under-predicted. Turbulent kinetic energy has undergone rapid changes near the cube, Fig. 6.39(d), and its maximum is in good agreement with the measured profile but it is more diffused.

The above evidence shows that the velocity, pressure and k fields are near mesh-independence. This is not the case for the ϵ field which exhibits singular behaviour near the leading edges of the cube. The mesh is highly refined in the regions of high gradients and has remained coarse elsewhere. Even though the refinement

	% of maximum error									
Mesh	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
Starting	86.71	1.16	0.58	0.32	0.26	0.41	0.77	0.095	0.32	0.018
Final	96.12	0.33	0.21	0.22	0.085	0.024	0.0072	0.0047	5e-4	8.3e-5

Table 6.6: Distribution of estimated velocity error on the faces

is highly localised, the mesh has not become better than the starting mesh, see Table 6.6, because the percentage of faces with very low error has increased after the refinement. This happens because the maximum estimated velocity error near the leading cube edges does not reduce as expected.

6.6 Summary and Conclusions

The mesh-refinement procedure described in Chapter 5 was tested on a set of cases of engineering interest.

The procedure was found capable of producing solutions which were in good agreement with the benchmark solutions for laminar flows, and the final adapted meshes had much smaller numbers of cells than the benchmark meshes.

The calculation using a low-Re turbulence model has produced a mesh which is very fine near the walls, and mostly refined in the direction of high gradients. This has resulted in a highly-directional refinement and high savings on the number of cells when compared to the benchmark solution.

Similar patterns of refinement in the near-wall regions were observed in the calculation using a high-Re turbulence model, but the refinement of near-wall cells

was blocked in the direction normal to the wall for $Y^+ < 50$ even though the errors there were still above the required upper limit.

The fact that the FREE over-estimates the error in the diffusion-dominated parts of the flow makes the procedure produce over-refined meshes there. It was also found that the error estimator severely over-estimates the error near sharp corners where solution gradients tend to infinity and prevent the estimated error from tending to zero. Analyses of errors on the faces have shown that the refinement procedure also generates faces with low errors which could only be removed by some coarsening procedure.

Chapter 7

Adaptive-Polyhedral-Mesh Generation

7.1 Introduction

An error-driven adaptation procedure for CFD calculations on hexahedral meshes was described and tested in the preceeding two chapters. The procedure is dependent on the use of auxiliary criteria which are introduced in order not to generate split-hexahedron cells in the regions where they can reduce accuracy. In this chapter an adaptive-mesh-generation technique for polyhedral meshes will be presented.

Polyhedral meshes consist of cells of arbitrary shapes, for which the distribution of nodes is adjusted to achieve the required accuracy. After the distribution of nodes is known, it is supplied to the mesh generator to generate a new mesh. In order to achieve this, a polyhedral-mesh generator based on Delaunay triangulation was developed in the present study and will be presented in this chapter.

The chapter is organised as follows. A literature survey of mesh-generation methods is given in Section 7.2. The theory behind the Voronoi polygons and the corresponding Delaunay Triangulation will be given in Section 7.3. The algorithm used to calculate the Delaunay Triangulation will be described in Section 7.3.1 along with the possible degenerate situations and the remedies for them. Polyhedral-mesh generation will be described in Section 7.5.

A comparison of accuracies and computational times obtained on different mesh types will be discussed in Section 7.5.1.

A mesh-adaptation procedure for polyhedral meshes will be presented in Sec-

tion 7.6. The main parts of the procedure are: estimation of the error on the faces of the polyhedral mesh and generation of a new polyhedral mesh. The performance of the procedure will be tested in Section 7.6.1.

Conclusions and some closing remarks will be given in Section 7.7.

7.2 Literature survey

As numerical methods become applicable to more complicated problems and computer resources advance quickly, mesh generation has emerged as the major bottleneck [156]. Tetrahedral-mesh generators have reached a high degree of automation, but tetrahedral meshes are not a preferable type of mesh because of low accuracy. Hexahedral-mesh generators still require substantial user intervention to generate a mesh of good quality. The aim to achieve good accuracy and automate the mesh-generation process has motivated researchers into developing polyhedral-mesh generators.

Considerable research has been done on mesh generation and it will be reviewed here considering tetrahedral-mesh generation first, followed by hexahedral-mesh generation. Polyhedral-mesh generation and applications of Voronoi meshes will be discussed at the end of this section.

Tetrahedral-mesh-generation methods can be divided into two main classes:

- **Advancing front methods (AFT)** consist of element additions into the mesh by marching into a yet ungridded space and creating elements one by one. The region between the gridded and the ungridded regions is called the front. At the early stage of development AFT methods were designed to generate meshes from a set of vertices created before the mesh-generation process has started [32, 121], which limits the quality of generated meshes. Over the years, more advanced generators have appeared, which generate vertices during the process [78, 93, 119]. This improves the quality of generated meshes. Optimal data structures have also been implemented which reduce the complexity of mesh generation to O(N log(N)), where N is the number of tetrahedra, and the time required to generate the mesh [26, 78]. AFT has also been ported to parallel computers [93, 158].

  AFT is a reliable and robust method for generating good-quality tetrahedral

meshes with pre-determined cell sizes. It is limited to tetrahedral meshes only which are not a preferable type of mesh from the aspect of accuracy of the FV method.

- **Delaunay methods** are based on the property that a set of mesh points in space can be connected into a triangulation in 2D or tetrahedrisation in 3D, with the properties defined in Section 7.3, by using an automatic procedure. At the early stages of development Delaunay methods were designed to generate meshes from a pre-defined set of points, and by removing the tetrahedra with centroids outside the domain from the mesh [33]. Such methods work well for convex geometries, while on concave ones they may result in incorrect boundary description. The fact that the original boundary triangulation must be recovered after mesh generation also has an adverse impact on mesh quality. The quality and reliability of Delaunay mesh generators was further improved by enabling automatic creation of internal vertices and by implementing procedures which can recover the original boundary surface from the Delaunay structure for general geometries [51–53, 155]. One of the adverse effects on mesh quality associated with Delaunay methods is the appearance of flat tetrahedra called *slivers* which appear in 3D applications. Joe [79] has replaced the circumsphere criterion with a Max-Min solid angle criterion, ensuring that all solid angles are maximised, which improved the quality of generated meshes at the expense of time required to complete the process. He has shown that for 2D problems the Max-Min angle triangulations and the Delaunay triangulations are identical, thus producing well-shaped triangles. Avoidance of slivers has also been discussed in [33].

  An advantage of Delaunay methods is that points which form a mesh can be connected into a valid triangulation automatically. The main disadvantages of the method is the generation of flat slivers and the recovery of the boundary from Delaunay structure.

- **Octree methods** are an approach where the computational domain is represented as an octree structure which is obtained by recursive subdivision of a cube which encompasses the entire domain into smaller cubes called octants. The octants are subdivided until the required mesh resolution is achieved. When the octree structure is finished, the mesh is generated by decompos-

ing the octants into tetrahedra. Examples of this approach can be found in [137, 159]. This methodology can also be modified to generate hybrid meshes consisting of hexahedra and tetrahedra. The advantage of octree methods is that they reduce the number of searching operations because they are localised to octants. Because the tetrahedra are generated by decomposing octants, it mostly produces tetrahedra with right angles which may locally be of low quality.

An extensive review of tetrahedral-mesh-generation methods was presented by Löhner in [92].

Even though hexahedral-mesh generators have been developed, they are still not as reliable as required. Hexahedral-mesh-generation methods can be classified into inside-out, outside-in and domain-decomposition methods.

- **Inside-out methods** generate meshes by filling the interior of the domain with squares or hexahedra of the desired size which are all contained within the domain. The mesh is closed by projecting points on the boundary to form boundary cells. After the mesh is connected, a smoothing procedure is applied to improve its quality. Examples can be found in [136, 153]. A difficulty associated with this approach is the control of mesh quality near the boundary where it may generate small and skewed cells. This may cause problems for viscous-flow computations because the mesh may be of low quality in the boundary layers which is not desirable from the aspect of accuracy.

- **Outside-in methods.** Paving is an example of an outside-in method, which generates quadrilateral meshes by layering the geometry with rows of cells starting from the boundary towards the interior [25, 160]. A 3D method based on this approach, so-called plastering, was developed by Cass *et al.* [31]. It does not require any subdivision of the domain and it generates good-quality meshes near the boundary but their quality may deteriorate within the domain.

  Indirect methods which generate quadrilateral meshes from triangular meshes by merging triangles into quadrilaterals were also developed [111, 162]. In order to be able to produce an all-quadrilateral mesh, the boundary surface must be discretised in an even number of segments.

Outside-in methods based on dual graphs, which represent the connections between the hexahedra in the mesh, have become popular recently [30, 101, 144]. The mesh is created by advancing from the surface mesh into the interior forming dual entities whose intersections form a hexahedron. These methods are still not capable of producing meshes without any user intervention. The problem is associated with the creation of dual entities which may intersect in an inappropriate manner, thus resulting in an invalid mesh.

In general, Outside-in methods generate high-quality meshes near the boundary and the rotation or translation of the given geometry results in the same mesh topology. The complexity of meshing procedures is the main drawback of the method and the reason why this type of generators are still at early stages of their development.

- **Domain-decomposition methods** are another important group of methods based on decomposing the domain into primitive polyhedra which can be filled with hexahedra [89, 90, 125, 126]. A method commonly used for meshing sub-domains is midpoint subdivision. In order to make the meshes in subregions match at their boundaries, an integer-programming technique can be used for determining the number of divisions of each edge [90]. Decomposition of the domain is facilitated by medial axes positioned in the middle between the opposite surfaces. Some authors have used copying of meshes from a template, smoothing and sweeping of 2D meshes through a 3D space to generate hexahedral meshes in the sub-domains [82, 156].

  Domain-decomposition methods produce meshes with discontinuities in regions where the sub-domains meet which has an adverse effect on the accuracy at these regions.

In [143] Tautges gives a review of the state of art of the hexahedral meshing. Mapping techniques along with an extensive overview of the unstructured mesh-generation techniques can be found in Thompson *et al.* [145].

The development of Finite Volume solvers has enabled the use of cells of arbitrary topology, *i.e.* [37, 69]. Such polyhedral meshes can consist of cells of different shape depending on the mesh-generation method and the complexity of the domain. Sampl [132] presented a mesh generation method using the Delaunay method to determine

the medial surface within the domain. Regions between the boundary and the medial surface are filled with cells by creating layers of cells starting from the boundary and going towards the interior. The method may produce low-quality cells near the medial surface which must be topologically cleansed to provide a valid mesh. The method is considered very time-consuming due to large number of steps required to generate the mesh. An example of a method similar to paving [31] can be found in $H-$morph developed by Owen *et al.* [110], which generates a hexahedra-dominant meshes from background tetrahedral meshes. The mesh-generation process starts by generating a surface mesh consisting of quadrilaterals, which are then decomposed into triangles which then serve for tetrahedral-mesh generation; the tetrahedra are then merged into hexahedra. The process is started at the boundary and proceeds towards the interior until no further hexahedra can be created. The method is capable of creating meshes of good quality but at the expense of time required to perform all necessary tests before creating a hexahedron. A method starting from a background tetrahedral mesh, which generates meshes consisting of hexahedra, pentahedra and tetrahedra, by employing dual graphs, was discussed by Meshkat *et al.* [99]. The method relies on the existence of a dual graph of the tetrahedral mesh in which different topologies can be found, *eg.* dual graphs of hexahedra decomposed into tetrahedra or pentahedra decomposed into tetrahedra. By searching through the tetrahedral mesh it is possible to recognise these topologies and merge tetrahedra into hexahedra or pentahedra. The examples show variable success depending on the size of the tetrahedral mesh and the geometric complexity.

Meshes of arbitrary topology can also be generated using the Delaunay technique where the dual of the triangulation, so-called Voronoi Polyhedra, are used for computation, Fig. 7.1. A FVM method for Voronoi meshes was presented by Taniguchi *et al.* [140]. In [141] the methodology was tested on a turbulent flow around a car. The performance of the method with regard to the calculation time was found similar to that for structured hexahedral mesh. The advantage of polyhedral meshes was found in mesh generation as it was less time consuming then to generate a hexahedral mesh. A FV method using polyhedral meshes for simulating moving fronts during mould injection and casting was presented by Maliska *et al.* [95]. Another application of Voronoi meshes for multi-material problems was developed by Dukowicz *et al.* [41]. The polyhedral mesh was used due to its ability to adapt to the Lagrangian interface without reducing the quality of the mesh. Mishev

[100] has analysed properties of the second-order-cell-centred Finite Volume Method on Voronoi meshes and has proven that the expected convergence is between first and second-order which is in agreement with findings from Chapter 3.

Although modern mesh-generation tools can generate tetrahedral meshes reliably and quickly, tetrahedral meshes are not the desirable type of mesh from the computational viewpoint because of their low accuracy. In contrast, good quality hexahedral meshes provide better accuracy but reliable methods for their automatic generation are still nonexistent for complicated geometries appearing in engineering applications. Polyhedral meshes based on Voronoi cells can be generated automatically without much intervention from the analyst and should be of a good quality even for complex geometries. Polyhedral meshes also offer a good basis for adaptive mesh generation because the quality of polyhedral cells should remain high even for meshes with highly-localised refinement.

7.3 Voronoi Polygons and Delaunay Triangulation

For a given set of points P_n in a plane, a Dirichlet tessellation is a partition of the plane into a number of convex polygons, one per point, Fig. 7.1. Each point is assigned a territory that is the area of the plane closer to that point than to any other point in the set. This tessellation of non-overlapping convex polygons, called Voronoi Polygons, covers the whole domain [28]. The Voronoi Polygon V_i around the point P_i is defined as [154]:

$$V_i = \{P : \|P - P_i\| < \|P - P_j\|, \forall j \neq i\} \tag{7.1}$$

where P is the set of all points within the Voronoi Polygon V_i and $\|.\|$ represents the Euclidian distance between the two points. The Voronoi Polygon is formed by points in P which are closer to point P_i than to any other point in the set P_n. Voronoi Polygons may have finite and infinite territories, Fig. 7.1. Polygons with infinite territories are formed around points which are on the so-called convex hull. A side of the Voronoi Polygon lies midway between the two points forming the two neighbouring regions. When all the points which form two neighbouring regions are connected, the result is a triangulation of the convex hull. This triangulation is called a Delaunay Triangulation, and for each triangle in the triangulation there exists an associated vertex of the Voronoi diagram, Fig. 7.1, which is its circumcentre.

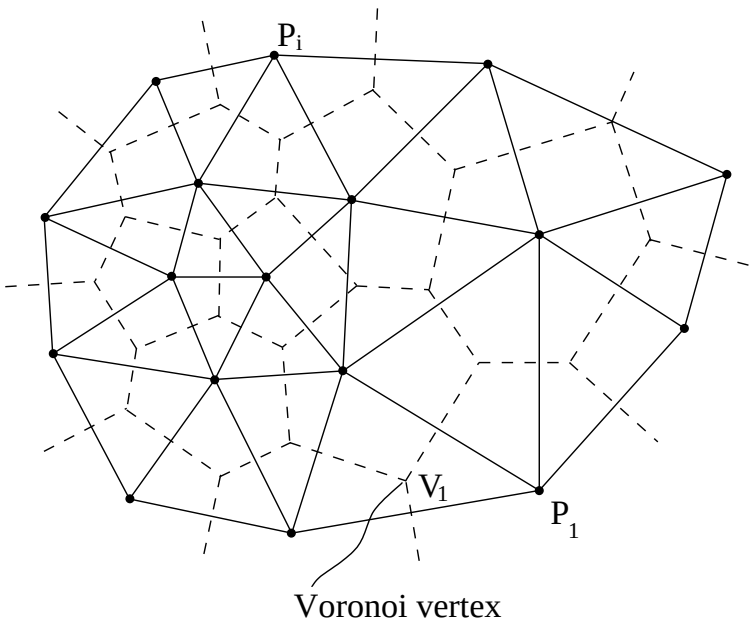
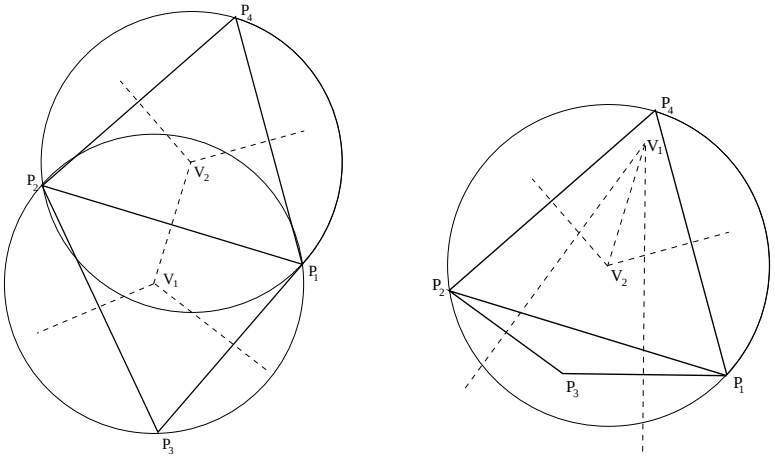


Figure 7.1: Delaunay Triangulation (solid lines) and Voronoi Polygons (dashed lines) for a set of points P_n (dots)

Therefore, each vertex of the Voronoi diagram corresponds to only one triangle in the Delaunay Triangulation. An important property of the Delaunay Triangulation is that no other points in P_n, apart from the ones forming a triangle, lie within the circle centred at this vertex, Fig. 7.2 a). If a point in P_n was within the circumcircle



a) Delaunay Triangulation b) Non-Delaunay Triangulation

Figure 7.2: Delaunay vs other triangulations

of the triangle, Fig. 7.2 b), this would result in the non-convex Voronoi Polygons which overlap yielding an invalid Dirichlet Tessellation.

In 3D, the Voronoi diagram is formed of convex polyhedra, one around each

point, and the points form Delaunay tetrahedra.

7.3.1 Algorithm for calculation of the Dirichlet Tessellation

The algorithm for creation of the Dirichlet Tessellation used here is based on the work of Bowyer [28] and is applicable to both 2D and 3D spaces. It is a sequential process in which the points in the set P_n are inserted into the tessellation one after another. Every point is inserted into the existing tessellation such that the tessellation is broken first and then reconnected with the point included in it. The algorithm for 2D space consists of the following steps [28]:

1. Start the calculation by defining four triangles from the first three points in P_n and an additional point called point at infinity, Fig. 7.3. The triangle containing points P_1, P_2 and P_3 forms the convex hull while other three triangles (P_1P_2N, P_2P_3N and P_1P_3N) which contain the point at infinity have infinite areas and they span the whole plane. The Voronoi vertex of the triangle which forms the convex hull is located at its circumcentre while other triangles have their Voronoi vertices located at the infinity. The data associated with a triangle is shown in Table 7.1. Every triangle holds information about:
   (a) Points forming it
   (b) Neighbouring triangles
   (c) Voronoi vertex
   (d) Information if the triangle is within the convex hull

Triangle	Forming points	Neighbouring triangles	In the hull?	Circumcentre
1	P1 P2 P3	2 3 4	yes	V1
2	N P1 P2	1 3 4	no	∞
3	N P2 P3	2 4 1	no	∞
4	N P1 P3	1 3 2	no	∞

Table 7.1: Data structure for the triangulation

2. Introduce a new point within the domain.
3. Determine which Delaunay triangles have to be changed due to the addition of the point. If the point to be inserted lies within a circumscribed circle of a

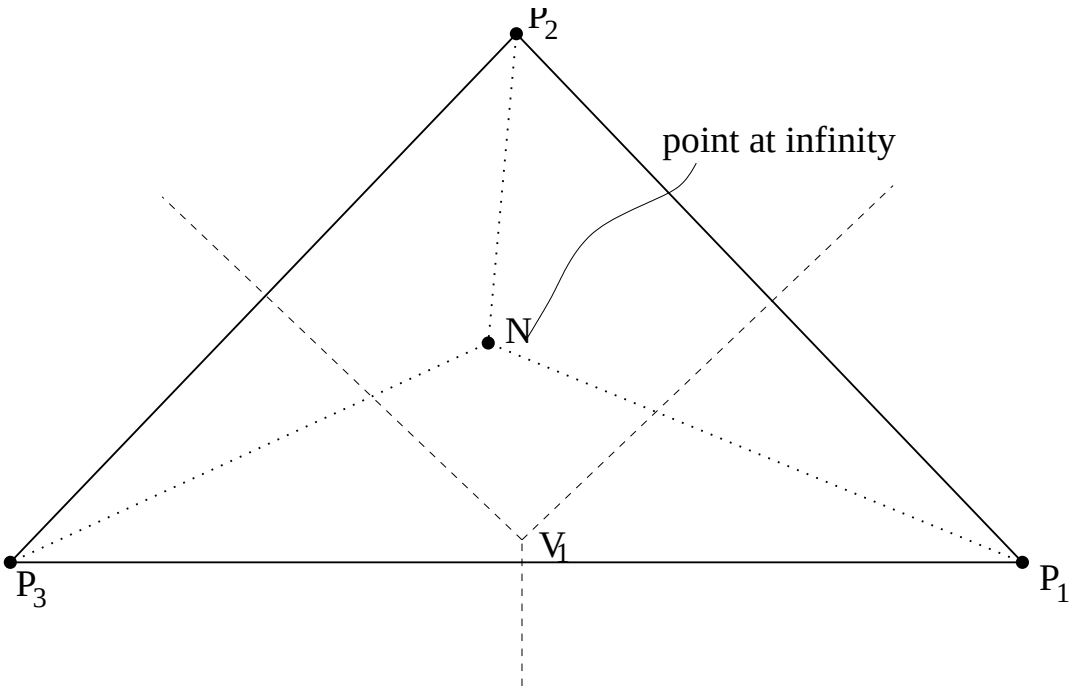


Figure 7.3: Initial hull for the Delaunay triangulation

triangle, the circle centred at the Voronoi vertex corresponding to the triangle, the triangle is marked for deletion.

4. Find all the points forming the triangles marked for deletion as these are the contiguous points to the point which will be inserted.

5. Determine the Delaunay triangles which are the neighbours of the deleted Delaunay triangles and are not themselves deleted. This information will be used for reconnection of the tessellation.

6. Determine which points form the new Delaunay triangles. Each new triangle is formed from the new point and the two points contiguous to that point.

7. Set the information about neighbours both into the newly created triangles and the already existing ones.

8. Store the new Delaunay triangles and delete the ones removed from the tessellation from computer's memory.

9. Repeat steps 2-8 for every new point.

Step 3 is the most time-consuming part of the above algorithm. It requires operations which are not local in nature and cannot be carried out in an operation

count independent of the total number of points in the tessellation. Therefore, a naive search would require a $O(N^2)$ workload. The workload can be minimised using the following search:

- Find a triangle which will be deleted.
- Search through its neighbours and check whether they should be deleted as well. This step is recursive as further neighbours should be searched until no triangles for deletion can be found.

Finding the initial triangle for the above search does not require much workload if the points are ordered in a systematic manner. Then the last inserted triangle is a reasonable starting triangle. To speed up the process, the following search is used:

1. Start the search from the last inserted triangle. Check if the point lies inside the circumcircle of that triangle. If the point is inside the circle the search is stopped.
2. Find the distances from the point to the Voronoi vertices corresponding to the neighbouring triangles.
3. Determine the smallest distance of all and set the searching triangle to the one corresponding to the smallest distance.
4. Check if the point is inside the circle of the new searching triangle. If it is inside the circle stop the search or else go to Step 2.

It is important to note that problems can arise in the triangulation procedure when certain degeneracies occur in the data. The most common degeneracies in 2D are:

1. Two points are coincident.
2. Three points of a potential triangle lie on a straight line.
3. Four or more points lie on a circle.

The first degeneracy is easily avoided by rejecting points which are very close to any of the points which are already in the tessellation. The second degeneracy is easily detected as it implies that the radius of the formed triangle tends to infinity. It can be alleviated by moving the point to be inserted by a small distance. The

third degeneracy is of a different nature than the previous two as it produces a triangulation which is not degenerate but the Voronoi vertices corresponding to two or more triangles are coincident. This degeneracy can manifest itself at the latter stage of the process when the insertion of another point may result in the deletion of one triangle but not the others which share the same Voronoi vertex. It can cause breakdown of the algorithm or cause the appearance of Voronoi Polygons which overlap and are not convex. The problem is caused by the round-off error. In order to avoid cyclic points from being inserted it is advisable to check whether the point lies on the circumcircle of any triangles selected for deletion. If it does, it is moved by a small distance. Therefore, it is advisable to insert first the points which must have certain positions (*i.e.* boundary points, corner points), as it may happen that they have to be moved. When calculating the tessellation in 3D space, an additional degeneracy appears when the four points forming a potential tetrahedron lie in a plane.

7.4 Computational mesh

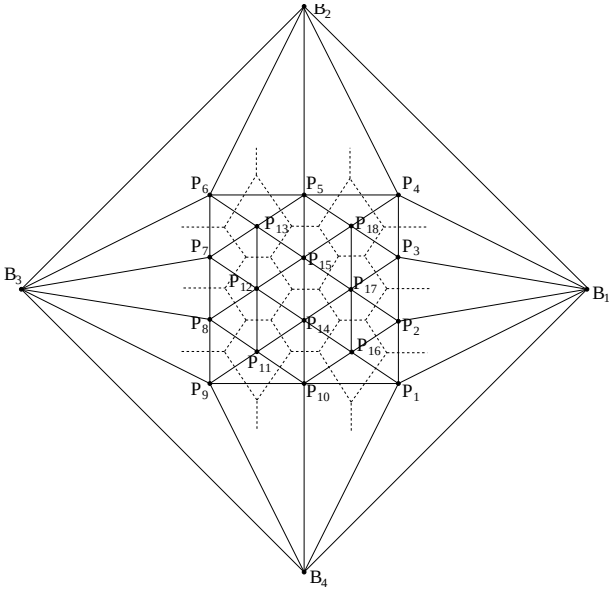
The information provided by the Delaunay method presented in the previous section is not in a form required for CFD analysis. The mesh for CFD calculations is defined as follows [94]:

- A list of coordinates of mesh vertices.
- A list of mesh faces. A face is defined as a list vertex numbers forming it.
- A cell is formed from faces in the face list and is given as a list of face numbers in the face list. Cells are stored in a list.
- The boundary is decomposed into a list of patches where the patch corresponds to a region of the boundary. Each patch contains information about the faces forming it.

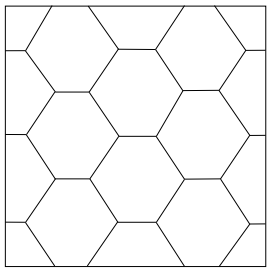
Note that the definition of the mesh does not impose any limitations on the shape of faces and cells.

7.5 Polyhedral-Mesh Generation

A polyhedral mesh can be constructed from convex Voronoi Polyhedra. The polyhedra which intersect with the boundary surface are cut by the boundary and only the parts which are within the domain are taken, Fig. 7.4, to form a closed mesh suitable for the FV discretisation practice presented in Chapter 3. Because the polyhedra must be cut near the boundary, the algorithm is dependent on the dimension of the space considered. Both 2D and 3D cases will be presented here.



a) Dirichlet Tessellation



b) Polyhedral Mesh

Figure 7.4: Dirichlet Tessellation and polyhedral mesh

The generation of the polyhedral mesh is started when the Delaunay Triangulation is already constructed. There are two main issues which have to be resolved here:

- How to construct Voronoi Polyhedra from the available data, Table 7.1, which is stored as Delaunay triangles instead of Voronoi Polyhedra?

Delaunay Triangulation	point	edge	face	tetrahedron
Dirichlet Tessellation	polyhedron	face	edge	Voronoi Vertex

Table 7.2: Relations between objects forming Delaunay Triangulation and Dirichlet Tessellation

- How to cut the polyhedra near the boundary?

It will be recalled that the Voronoi Polyhedra form the Dirichlet tessellation which is a dual of Delaunay triangulation. This property will be used for generation of polyhedra. Relations between objects of Delaunay Triangulation and Dirichlet Tessellation are given in Table 7.2. Fig. 7.4 shows these relations in 2D space, where the faces of both Delaunay Triangulation and Dirichlet Tessellation are contracted into edges of Delaunay Triangles and Voronoi Polygons.

Table 7.2 shows that a point in the Delaunay Triangulation corresponds to a polyhedron in the Dirichlet Tessellation. Therefore, the number of cells in the polyhedral mesh is equal to the number of points in the set P inserted into the Delaunay triangulation. The Voronoi Polyhedra originating from the set of points B forming the bounding box around the domain are excluded Fig. 7.4.

Another property of the Dirichlet Tessellation which is used for mesh generation is the fact that there should be a polygonal face around every edge of its corresponding Delaunay Triangulation, see Table 7.2. The number of vertices and faces of the polyhedral mesh cannot be calculated before the cuts of polyhedra with the boundary are resolved. Due to this, vertices and faces of the polyhedral mesh are created during the mesh-generation process and stored into their corresponding lists every time a new vertex or face is created. The process of creating mesh vertices and faces in the 3D space consists of the following steps:

1. **Generate internal vertices.** Vertices of the polyhedral mesh which are within the domain correspond to Voronoi Vertices which are the centres of circumscribed spheres around tetrahedral cells, Fig. 7.2 a). Only the Voronoi Vertices which are inside the domain are stored into the list of mesh vertices, Fig. 7.4. Vertices located at the boundary are created during the generation of internal and boundary faces.

2. **Generate internal faces.** Table 7.2 shows that faces of Voronoi Polyhedra

correspond to edges of Delaunay Triangulation. This property will be used for generation of internal faces. In order to simplify and speed up the process, the following information is created first:

(a) A set of edges L in the Delaunay Triangulation which are formed by points forming the domain (set P in Fig. 7.4 a)).

(b) For every edge in the set L find which tetrahedra share that edge and order them such that every tetrahedron shares a face with the previous and the next one.

Internal faces of the polyhedral mesh are created from Voronoi Vertices of the tetrahedra around every edge of the set L, Fig. 7.5. If there exist Voronoi Vertices which are outside the domain, only the part of the face within the domain is retained by calculating coordinates of the boundary vertices at which the edges connecting two neighbouring Voronoi vertices intersect the boundary, Fig. 7.5. Every boundary vertex also holds information about the boundary region it is in. This will be used later to create boundary faces of the mesh. A

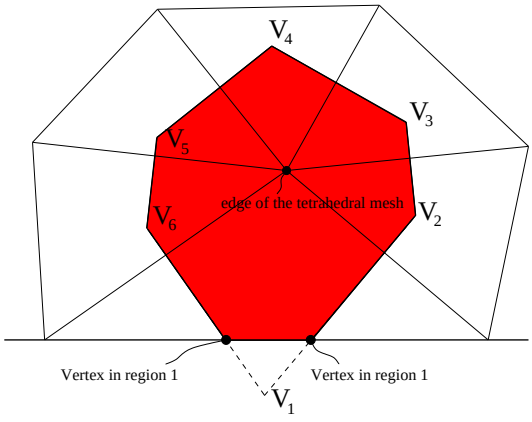


Figure 7.5: Generation of an internal face of the polyhedral mesh (section in the plane of the face)

word of caution is necessary here. Table 7.2 shows that an edge of the polyhedral mesh corresponds to a triangular face in the Delaunay triangulation. From this it follows that an edge of the polyhedral mesh appears in three different polygonal faces created from the edges of a triangular face. In order not to generate and store the same vertex more than once, a number of the vertex in the list of vertices is made available to the triangular face corresponding to

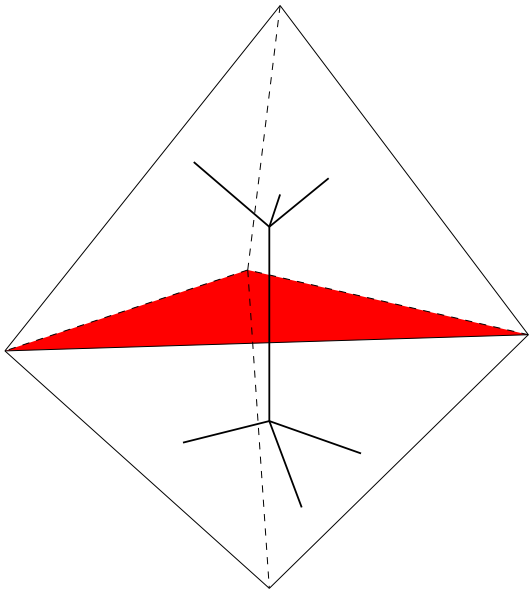


Figure 7.6: Tetrahedra sharing a face. The edge of the polyhedral mesh (thick lines) is perpendicular to the shared triangular face (red). There exist a polygonal face for every edge of the triangular face.

that edge of the polyhedral mesh, Fig. 7.6. This information can be read next time that edge of the polyhedral mesh is checked to store the vertex into the face of the polyhedral mesh without storing it into the list of vertices again.

Internal faces of the polyhedral mesh can also intersect edges which define borders between two boundary regions of the domain, Fig. 7.7. If this happens, the intersection is also stored into the face and into the list of vertices. Such vertices cannot appear more than once and they do not require any special treatment.

Polyhedral cells sharing a created internal face are known as they correspond to the points of the set P which form the edge. This is a consequence of the definition of the Dirichlet Tessellation where there exists one convex polygon around every point in the Delaunay Triangulation, Table 7.2.

3. **Generation of boundary faces** is started after the internal faces are known. It was mentioned before that every boundary vertex holds information about boundary regions it is in, Fig. 7.8. By sweeping through the list of cells, it is possible to create boundary faces for every cell such that all boundary vertices found in the polyhedron and which are in the same boundary region form a

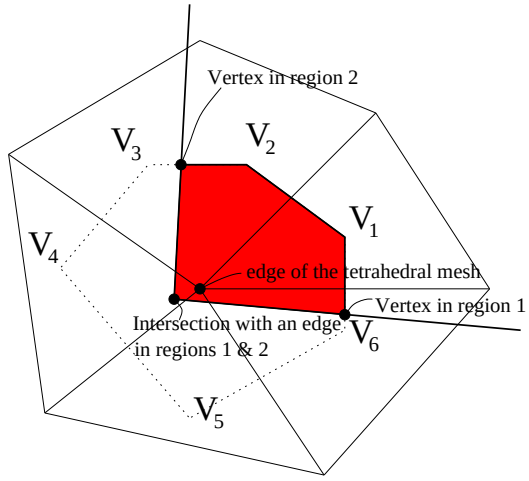


Figure 7.7: Generation of an internal face including the intersection of a boundary edge (section in the plane of the face)

boundary face. The number of boundary faces per polyhedron is dependent on the number of boundaries which cut it.

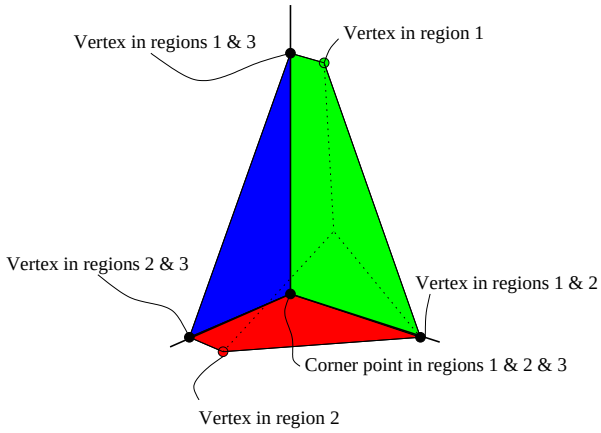


Figure 7.8: Generation of boundary faces (coloured) including intersections with boundary edges (thick lines) and a corner point

Points which define corners at which three or more boundary regions meet require more attention. Such points are not yet stored into the list of vertices but they must be kept for the sake of geometry definition. After the vertex is stored into the list of vertices, boundary faces can be created such that the new vertex is added into boundary faces created from boundary points in the same boundary region. An example can be found in Fig. 7.8.

Degeneracies discussed in Section 7.3 have an impact on the quality of Voronoi Polyhedra. If five or more points in the set P are nearly Co-spherical, the Voronoi Vertices of tetrahedra formed by those points are nearly coincident. If that happens the edges and/or faces of the polyhedral mesh have negligible size and are removed from the mesh.

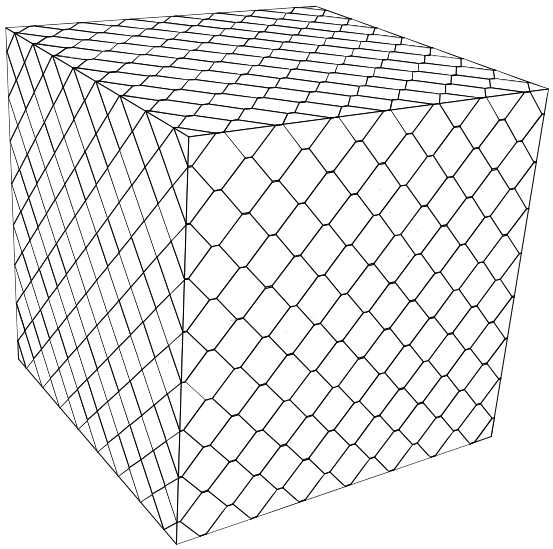


Figure 7.9: An example of a 3D mesh for a cube

The algorithm for generation of meshes in 3D space is currently limited to box-like domain, see Fig. 7.9, because the support for arbitrary surfaces is not implemented. The procedure would not change for arbitrary geometries and the only difference would be in calculating the intersections between the surface and an edge of the polyhedral mesh.

The algorithm for mesh generation in 2D space differs from the 3D one as follows:

1. **Internal faces.** Faces and edges are the same structure in 2D space for both Voronoi Polygons and Delaunay Triangles. An internal face of the polyhedral mesh is therefore a perpendicular bisector of an edge in L connecting Voronoi Vertices of the triangles adjacent to the edge, Fig. 7.4. If there exist Voronoi Vertices outside the domain only the part of the face which is within the domain is retained.

2. **Boundary faces** are constructed using the same procedure as in 3D case but with a difference that there should exist only two vertices which are in the same boundary region, Fig. 7.4.

7.5.1 A comparison of accuracy on Quadrilateral, Polygonal and Triangular Meshes

The accuracy of the FV discretisation practice on quadrilateral, polygonal and triangular meshes will be tested on two cases, namely the Planar Jet case described in Chapter 4 and the Flow Over a Cavity from Chapter 6. Variation of the exact discretisation error will monitored for all types of meshes.

Planar Jet

This case has already been introduced in Section 4.5.1 and used for testing of the FREE in Chapter 4 and the mesh adaption procedure presented in Chapter 5. The same discretisation procedures are applied here.

The purpose of this test is to compare the accuracy achievable on meshes of different cell types, namely; quadrilateral meshes, polyhedral meshes mainly consisting of hexagons and triangular meshes. The time required to obtain the solution on different mesh types will also be compared.

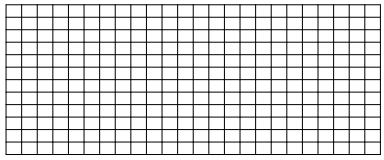
The comparison was conducted on five meshes for every cell type and their sizes are given in Table 7.3. They were created such that the first quadrilateral and polygonal meshes have a similar number of cells. The triangular meshes are duals of the polygonal mesh and are expected to have twice as many cells, see Fig. 7.4, as present for finer meshes. The meshes corresponding to the third row of Table 7.3

No	Quadrilaterals	Polygons	Triangles
1	18	19	20
2	72	59	86
3	288	211	362
4	1152	782	1448
5	4608	3063	5898

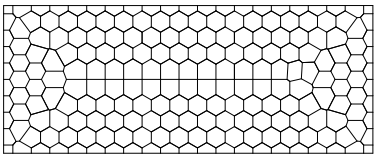
Table 7.3: Number of cells for different types of meshes (Jet case)

are shown in Figs. 7.10(a), 7.10(b) and 7.10(c).

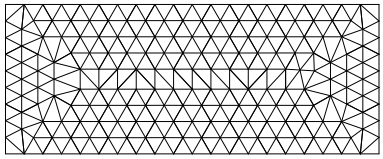
The exact-velocity errors obtained on the meshes in Figs. 7.10(a), 7.10(b) and 7.10(c) are shown in Figs. 7.11(a), 7.11(b) and 7.11(c). The error field on the quadrilateral mesh, Fig. 7.11(a), is smoother than on other types of meshes. The



(a) Quadrilateral mesh (288 cells)

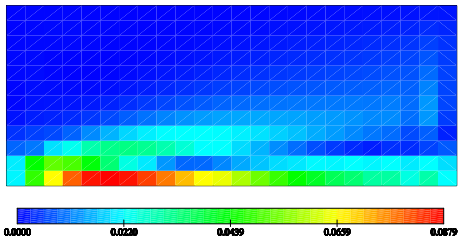


(b) Polygonal mesh (211 cells)

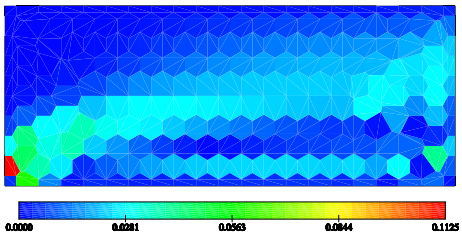


(c) Triangular mesh (362 cells)

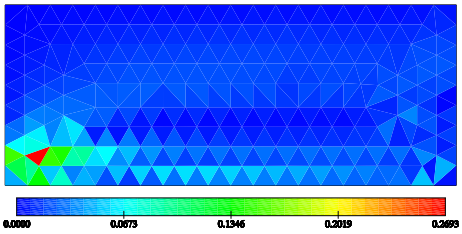
Figure 7.10: Meshes used for the jet case



(a) Quadrilateral mesh (288 cells)



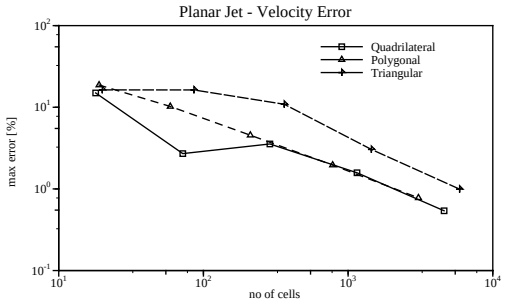
(b) Polygonal mesh (211 cells)



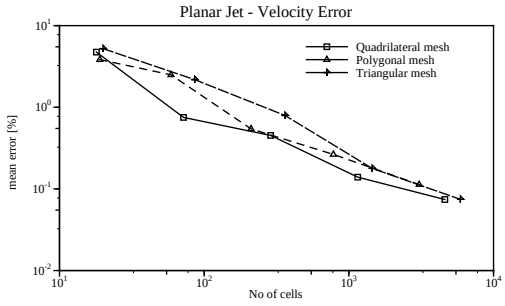
(c) Triangular mesh (362 cells)

Figure 7.11: Exact-velocity errors for the jet case [m/s]

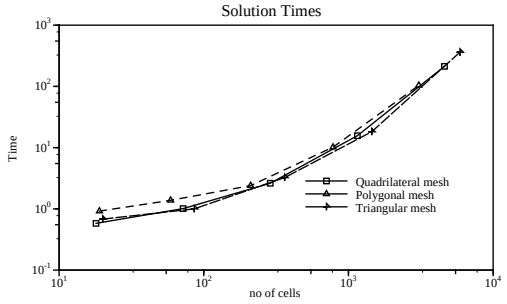
error fields on the polygonal mesh, Fig. 7.11(b), and triangular mesh, Fig. 7.11(c), have peaks in the regions where the regularity of the mesh is impaired. It is believed that this is a consequence of the fact that the approximation of the surface-normal gradient required for the discretisation of the diffusion term, Section 3.3.2, is of first-order on non-uniform meshes resulting in increased errors in the regions where the mesh is not uniform.



(a) Variation of the maximum exact error in the velocity field ($|\mathbf{U}_{norm}| = 2.474m/s$)



(b) Variation of the mean exact error in the velocity field ($|\mathbf{U}_{norm}| = 2.474m/s$)



(c) Time required to obtain the solution on different mesh types (s)

Figure 7.12: Variation of the exact-velocity error and solution times on different types of meshes for the jet case.

Fig. 7.12(a) shows that for the same number of cells the maximum exact error is largest on triangular meshes while it is at the same level for polygonal and quadrilateral meshes when the mesh becomes finer. The mean exact error depicted in Fig. 7.12(b) also behaves according to the findings from Section 3.6. The mean error is the smallest on quadrilateral meshes and the highest on triangular ones. On finer meshes all mesh types tend to the same curve which is the influence of the approximations used for deriving the analytical solution for this case, Section 4.5.

The calculations were terminated when the residuals of both pressure and velocity were below the 10^{-8}. The pressure equation is solved using the ICCG solver [68], the momentum equation is solved using Bi-CGSTAB solver [148] and the pressure-velocity coupling is treated using the SIMPLE algorithm described in Section 3.8. Fig. 7.12(c) shows the times obtained for all mesh types. Triangular and quadrilateral meshes require similar time while polygonal meshes require more time for the same number of cells. The times required on different mesh types tend to the same curve as the mesh becomes finer from which it follows that only the number of cells but not their type influence the performance of the solvers of linear equations.

Flow Over a Cavity

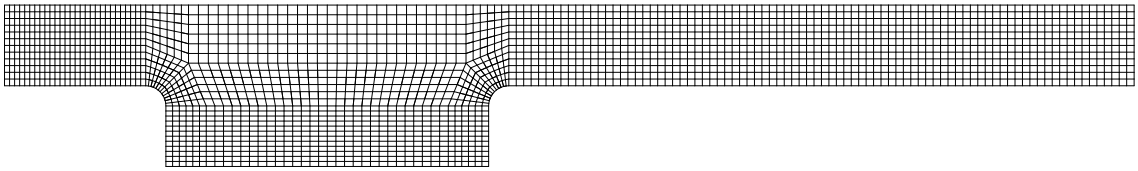
This problem has been introduced in Section 6.2 and is used to test the accuracy of different mesh types on a more complex example than the previous one. Four meshes of each type, summarised in Table 7.4, were used to determine which gives the best accuracy and to monitor the reduction of error with mesh refinement. The convergence of the predicted pressure-drop coefficient will also be monitored.

Quadrilateral and polygonal meshes in each row of the Table 7.4 were generated to have a similar number of cells while the triangular mesh is a dual of the polygonal mesh and is expected to have twice as many cells as the polygonal mesh. The quadrilateral, polygonal and triangular meshes corresponding to the second row in the Table 7.4 are shown in Fig. 7.13.

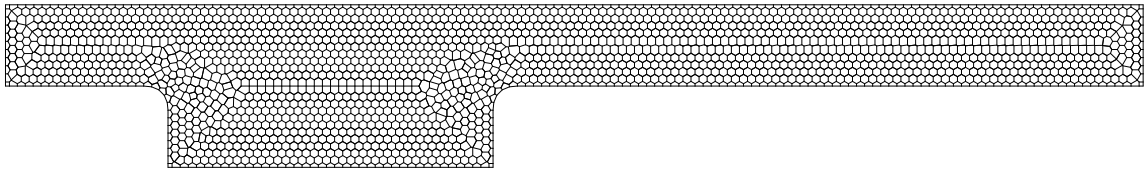
No	Quadrilaterals	Polygons	Triangles
1	594	556	950
2	2376	2159	3996
3	9504	8487	16336
4	38016	33352	65430

Table 7.4: Number of cells for different types of meshes

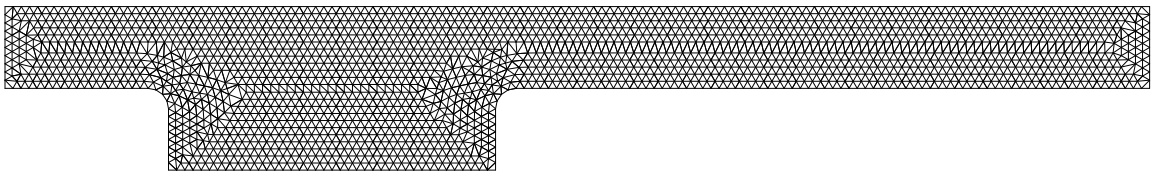
Variations of maximum and mean exact errors in the velocity field for different types of meshes are given in Figs. 7.14(a) and 7.14(b), respectively. Triangular meshes have the largest exact error for the same number of cells, as expected. The maximum exact errors found on quadrilateral meshes are almost the same as on the polygonal mesh. These findings are in agreement with the analysis given in Sec-



(a) Quadrilateral mesh (2376 cells)



(b) Polygonal mesh (2159 cells)

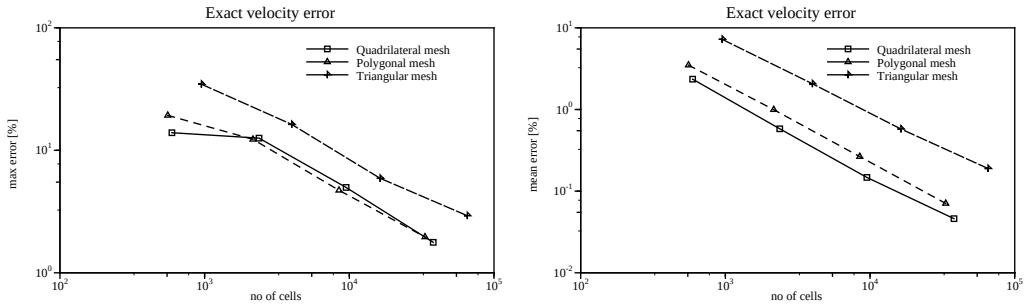


(c) Triangular mesh (3996 cells)

Figure 7.13: Quadrilateral, polygonal and triangular meshes used for comparison

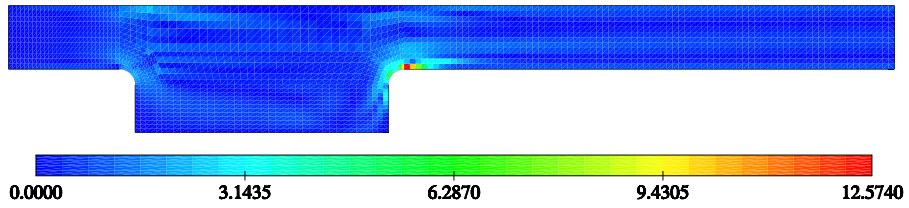
tion 3.6. The unexpected behaviour of the maximum exact error on the quadrilateral mesh between the first and the second cycles is attributed to inadequate resolution and non-uniformity of the mesh near the stagnation zone, see Figs. 7.13(a) and 7.15(a). The triangular mesh has the largest average exact error for a given number of cells and requires almost 10 times more cells to achieve the same accuracy as on the quadrilateral mesh. The average error on polygonal meshes is much closer to the quadrilateral mesh result, requiring approximately twice as many cells to achieve the same accuracy. The reduction of the average exact error is very close to the theoretical second-order reduction rate for all types of meshes and it deteriorates when the mesh becomes finer. This is probably caused by the remaining error in benchmark solution as its influence is larger on fine meshes.

A comparison of the pressure-drop coefficients (C_p) given in Table 7.5 and Fig. 7.17 shows that on the quadrilateral and polygonal meshes C_p tends to the mesh-independent

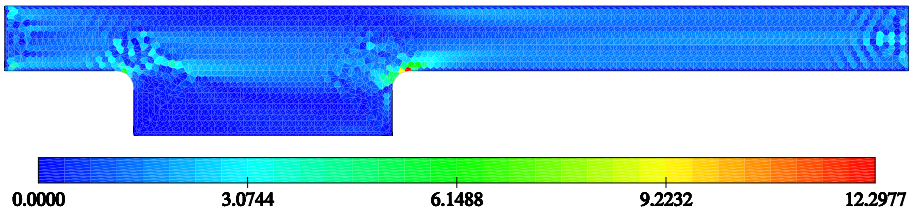


(a) Maximum error (b) Mean error

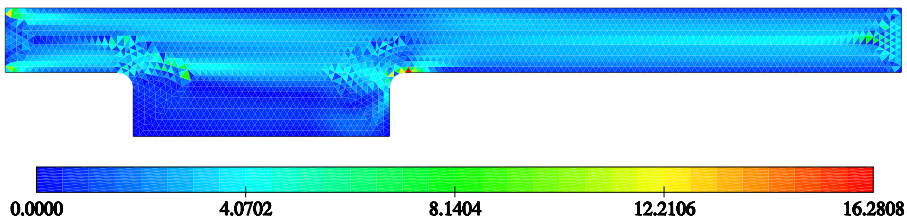
Figure 7.14: Variation of the exact-velocity error on different types of meshes for the cavity case. Errors are given as percentage of average inlet velocity U_{avg}.



(a) Quadrilateral mesh (2376 cells)

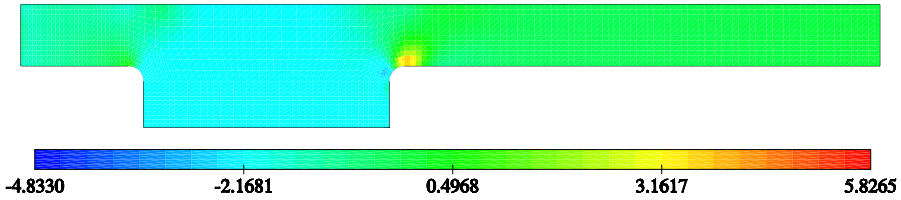


(b) Polygonal mesh (2159 cells)

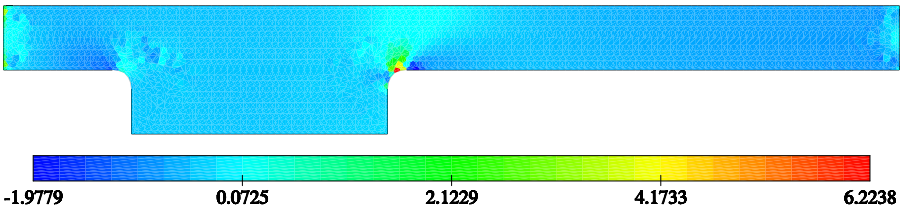


(c) Triangular mesh (3996 cells)

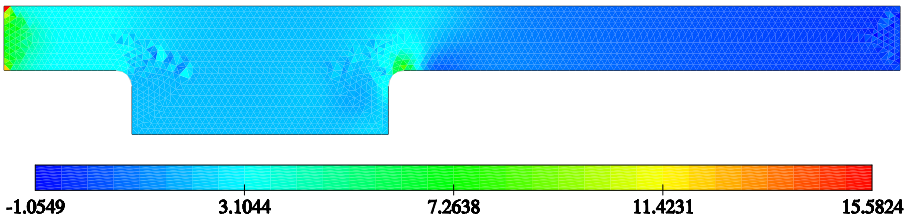
Figure 7.15: Magnitude of the exact-velocity error (given as percentage of average inlet velocity U_{avg})



(a) Quadrilateral mesh (2376 cells)



(b) Polygonal mesh (2159 cells)



(c) Triangular mesh (3996 cells)

Figure 7.16: The exact-pressure error (given as percentage of $p_{max} = \frac{1.64}{2}\,\rho\,U^2_{avg}$)

Pressure-drop coefficients (C_p)					
Quadrilateral		Polygonal		Triangular	
Cells	C_p	Cells	C_p	Cells	C_p
594	1.43128	556	1.59121	950	1.96345
2376	1.49723	2159	1.53935	3996	1.67216
9504	1.51544	8487	1.52665	16336	1.56432
38016	1.51982	33352	1.52333	65430	1.53555
Mesh-independent value $C_p = 1.5203$					

Table 7.5: A comparison of pressure drop for different types of meshes

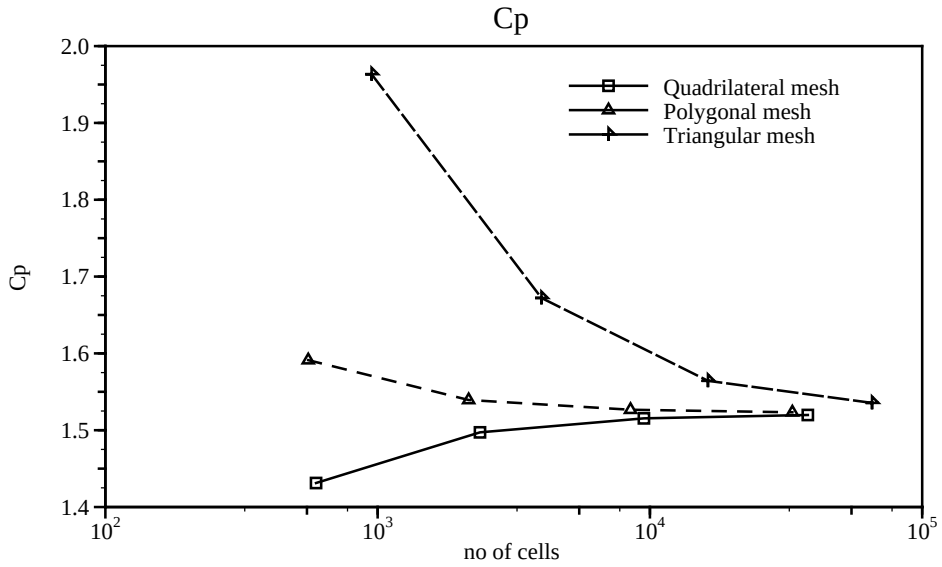
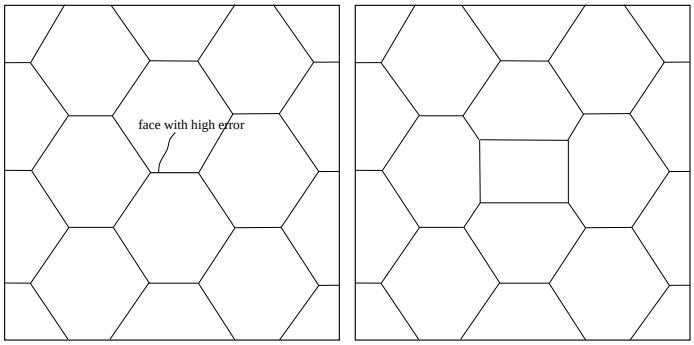


Figure 7.17: Scaling of the pressure drop for different types of meshes

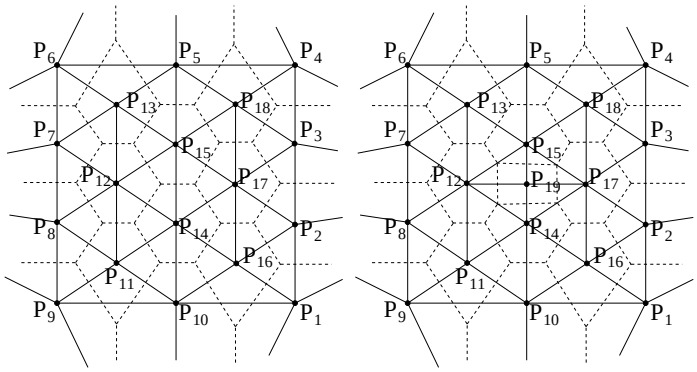
value more quickly than its triangular counterpart. The magnitude of the error is almost the same for quadrilateral and polygonal meshes for all mesh sizes, but their trends to the mesh independence are different. It is believed that different trends are caused by different distributions of the pressure error within the cavity and the channel, Figs. 7.16(a), 7.16(b) and 7.16(c). The error on the quadrilateral mesh is close to zero within the channel because the mesh is uniform and flow-aligned there, and consequently the variations of velocity and pressure fields can be captured exactly by the second-order FV approximation. It is negative within the cavity and thus causes the pressure drop to be smaller than required. On the polygonal and triangular meshes the error is negative within the part of the channel to the right of the cavity, which therefore causes the pressure drop to be higher than the exact pressure drop. The pressure error on the triangular mesh is caused by the error coming from the evaluation of mass fluxes through the faces, which is not zero, Eqn. (3.93), and this results in errors in velocity and pressure fields. The error on the polygonal meshes is concentrated in the regions where the mesh is non-uniform (the symmetry of the channel and the boundary layers), because the errors committed on mesh faces do not cancel there.

7.6 Mesh adaptation on Polyhedral Meshes

A mesh-adaption technique developed for polyhedral meshes is an adaptive-mesh-generation method because after the error estimation is performed a new mesh is generated with the density of the nodes increased in the regions of high error using the mesh generator described in Section 7.5. This is achieved by inserting points into the Delaunay Triangulation and then generating a new mesh after each computational cycle. As a point in the Delaunay Triangulation corresponds to a cell in the polyhedral mesh, see Figs. 7.18(c) and 7.18(d), the mesh-adaptation technique can be seen as the growing or insertion of cells within the mesh structure.



(a) Initial mesh (b) Refined mesh



(c) Delaunay Triangulation before refinement (d) Delaunay Triangulation after refinement

Figure 7.18: Adaptation of polyhedral meshes

The error is estimated using the FREE at all faces of the mesh. If the error on a face is larger than acceptable, a point located at the centroid of the face is inserted into the Delaunay Triangulation. An example is shown in Fig. 7.18(d) where the point P_{19} is inserted in the centroid of that face which has large error. Due to the

point insertion the parent mesh changes such that the inserted point generates a new cell in the refined polyhedral mesh, Figs. 7.18(a) and 7.18(b). The shape of the created cell is dependent on the distribution of points in the Delaunay Triangulation. The number of faces is equal to the number of edges in the Delaunay Triangulation sharing the inserted point.

This type of refinement should have a small impact on the mesh quality because the Delaunay method generates the best-shaped tetrahedra and polyhedra for a given set of points [79]. Figs. 7.18(c) and 7.18(d) show that points forming the Delaunay triangulation are very close to the nodes (centroids) of the polyhedral mesh. A face of the polyhedral mesh is almost perpendicular to the line connecting neighbouring nodes of the face. Hence, the non-orthogonality and skewness should be small. The mesh uniformity should also remain high because faces of the polyhedral mesh are midway the two neighbouring points in the Delaunay Triangulation.

The refinement procedure consists of the following steps:

1. Generate the starting mesh for the problem under consideration.

2. Solve the problem on the latest available mesh.

3. Estimate the discretisation error in the solution. If the error on every face is smaller than the required error E the calculation is stopped. If the error on a face is larger than E:

$$e_f > E,$$

   a point is inserted in the centroid of that face, Fig. 7.18(d).

4. Generate a new polyhedral mesh.

5. Map the solution from the previous mesh onto the new mesh.

6. Go to step 2.

7.6.1 Examples

Here, the adaptation procedure will be tested on a case already used in Chapter 5. The results obtained will also be compared with the results of the mesh-refinement procedure described in Chapter 5.

Convection and diffusion of heat

The description of this problem, including the numerical setup and the discretisation practices employed, can be found in Section 5.4.4. The calculation was started on a mesh shown in Fig. 7.19.

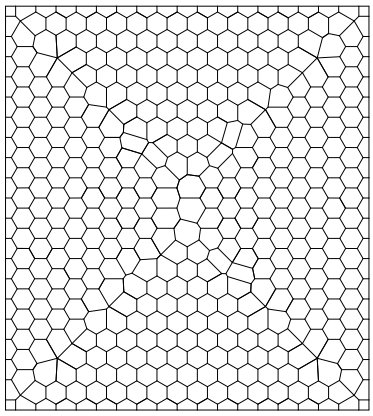


Figure 7.19: Starting polygonal mesh for the convection and diffusion of heat

The procedure was set to reduce the error below:

$$E = 0.5\,\% \tag{7.2}$$

of the maximum temperature $T_{max} = 1^oC$.

The required accuracy was achieved after nine cycles of adaptive refinement on the final mesh with 5925 cells shown in Fig. 7.20.

The refinement pattern in Fig. 7.20 is very similar to the one obtained by using the adaptive procedure described in Chapter 5, see Fig. 5.25(b). The mesh is highly refined in the regions of strong gradients. The region near the right boundary is predominately filled with quadrilaterals which appear from nearly degenerate Delaunay Triangulation, see Fig. 7.21 and are aligned with the direction of high gradients. Mesh-to-flow alignment is not present as in case when the adaptive mesh-refinement procedure of Chapter 5 was used, see Fig. 5.25(b). This is the main reason why the adaptive-mesh-generation procedure required 3.5 times more cells than the adaptive mesh-refinement procedure of Chapter 5 to achieve the required accuracy. The quality of the mesh has also remained good even though it is highly refined in some regions, Table 7.6.

Figs. 7.22(a) and 7.22(b) show the exact and the estimated error in the solution obtained after nine cycles of refinement. The error is over-estimated in the region

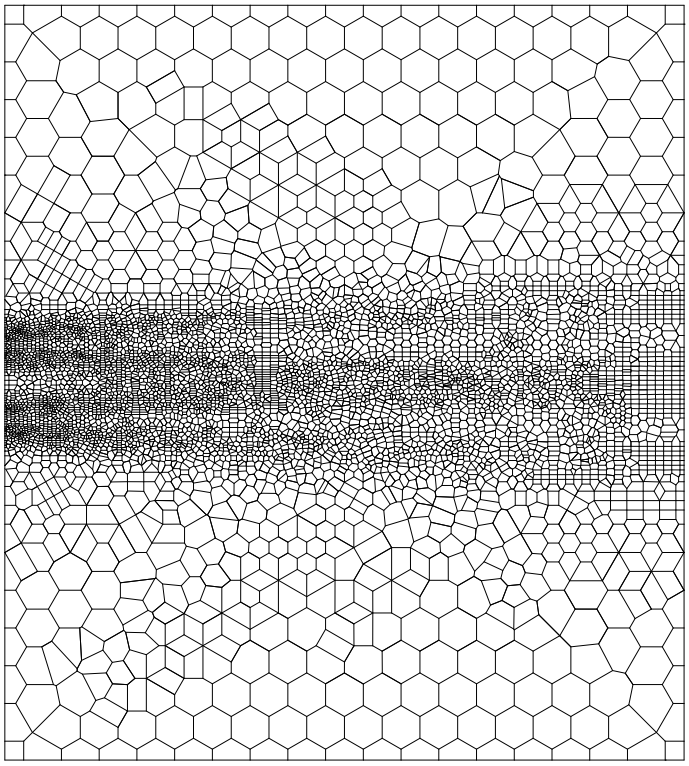


Figure 7.20: Mesh after 9 cycles for the convection and diffusion of heat (5925 cells)

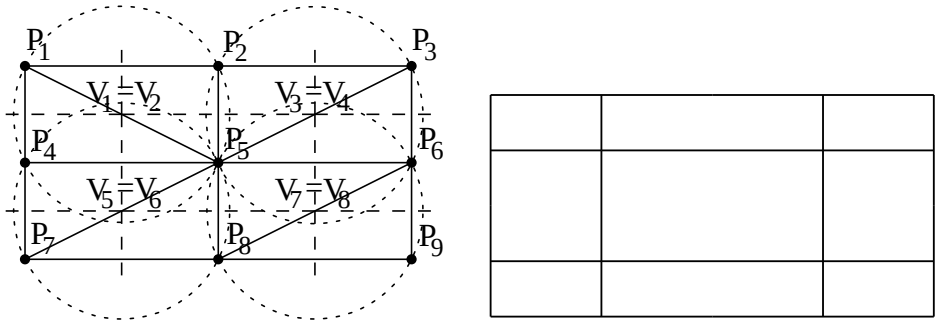
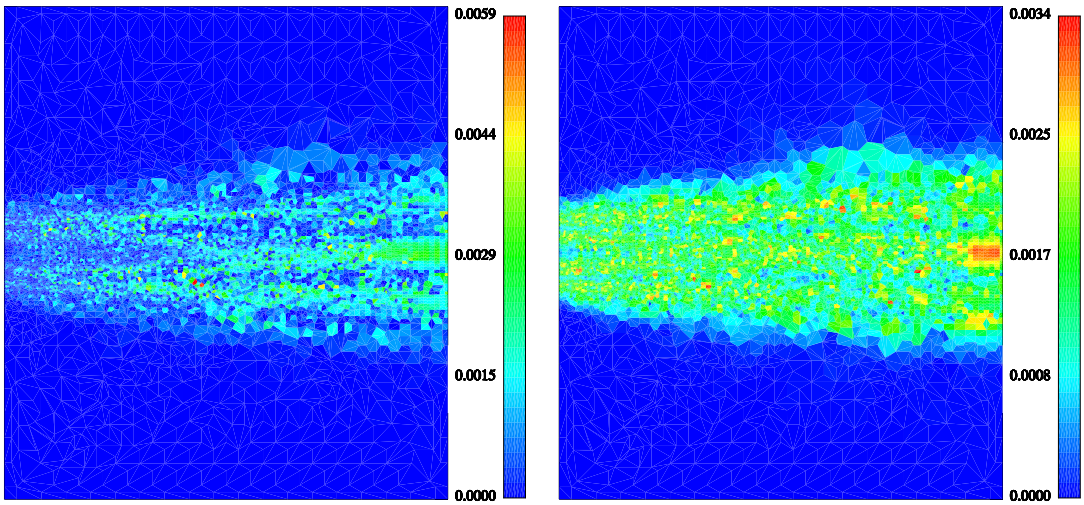


Figure 7.21: Polygonal mesh from nearly degenerate Delaunay Triangulation

	Non-orthogonality (deg)		Skewness	
	Maximum	Average	Maximum	Average
Starting mesh	13	1.74178	0.2	0.0295965
Refined mesh	39.5	5.04	0.78	0.08

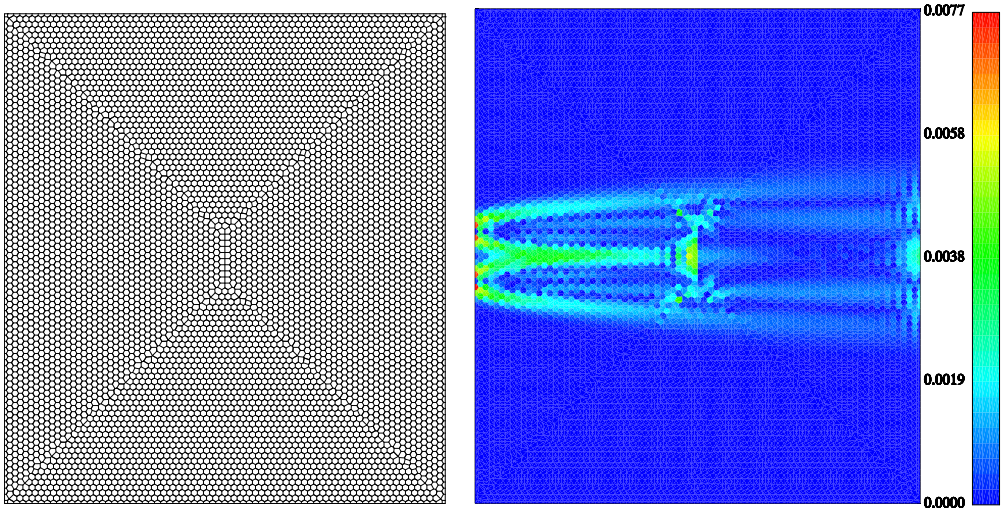
Table 7.6: Measures of mesh quality



(a) Exact error

(b) Estimated error (plotted as average of face errors)

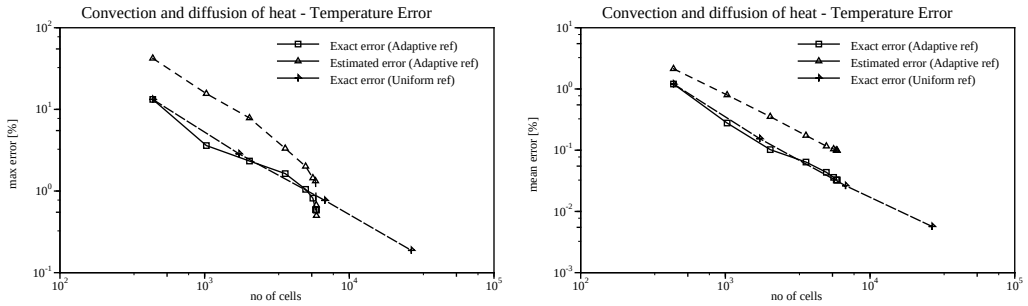
Figure 7.22: Errors after 9 cycles of refinement (5925 cells)



(a) Uniform mesh

(b) Exact error

Figure 7.23: Uniform mesh and the exact error (6769 cells)



(a) Maximum error (b) Mean error

Figure 7.24: Variations of errors with adaptive refinement for the convection and diffusion of heat ($T_{norm} = 1^oC$)

of high gradients. The reason for such behaviour has already been discussed in Section 5.4.4. The maximum exact error is slightly under-estimated, Fig. 7.24(a). There is no saving in the number of cells when the exact error is compared with the error on uniformly refined meshes. Fig. 7.23(a) shows a uniform mesh with similar number of cells as the adapted one. The error on the uniform mesh depicted in Fig. 7.23(b) is high in regions where the regularity of the mesh is impaired suggesting that the error is mainly influenced by mesh quality. The error on the adapted mesh, Fig. 7.22(a), is also high in the regions of high local refinement where the mesh quality is reduced. This is because most of the cells introduced by the refinement procedure do not have face pairs which would cancel face errors and therefore have higher error for the same cell size than the mesh consisting of hexagons.

7.7 Summary and Conclusions

A method for generation of polyhedral meshes suitable for the FV discretisation practice, and an approach to adaptive mesh generation on polyhedral meshes were presented in this chapter. Polyhedral meshes are generated by computing the Delaunay Triangulation and the corresponding Voronoi Diagram. The structure is then cut by the boundary surface in order to remove the parts of the Voronoi Polyhedra outside the boundary. The adaptive mesh generation is performed by inserting points into the Dirichlet Tessellation where a better resolution is needed, which is then followed by the generation of the new mesh.

The generator can produce meshes with smooth variation of cell sizes, and with low non-orthogonality and skewness without any intervention by the analyst. Unfortunately, it cannot generate 3D meshes for arbitrary geometry as the support for arbitrary geometries is not implemented.

Experiments conducted on triangular, polygonal and quadrilateral meshes confirm the findings from Section 3.6 that the quadrilateral/hexahedral meshes are the more accurate than other ones. The accuracy achieved on polygonal meshes is comparable with the one on quadrilateral meshes, while the triangular meshes consistently produced solutions with highest errors. The experiment conducted on the planar jet case suggests that the time required to converge the solution to the same residual is almost independent on the type of mesh used, but it only depends on the number of cells in the mesh.

The adaptive procedure has reduced the error below the required level and the reduction was monotonic. Unfortunately, it is not as effective as the procedure described in Chapter 5 as it produces meshes consisting of cell types which are not optimal from the aspect of accuracy.

Chapter 8

Conclusions and Future Work

8.1 Introduction

The aim of this study was to develop an accurate estimator of the discretisation errors for the FV solution method, and to develop an adaptive re-meshing procedure which is automatic and capable of producing solutions of required accuracy with savings in the number of required cells.

One of the important issues which had to be resolved was the influence of the mesh and the discretisation practice on the accuracy of the solution. The analysis of discretisation errors was performed for all terms appearing in the governing equations. It confirmed that the linear interpolation scheme for skewed meshes provides second-order accuracy of face integrals for any mesh. The approximation of the surface-normal gradient was of second-order on uniform meshes, but it becomes a first-order approximation on non-uniform ones. It was also shown that skewness and non-orthogonality have a negative impact on the accuracy. Gradient and divergence terms calculated using the Gauss-divergence theorem are confirmed to be of second-order accuracy. The FV approximation of the convective terms is dependent on the approximation procedure for the mass fluxes and on the interpolation procedure used for obtaining the value of the field under consideration at the mesh faces. The analysis of different interpolation schemes has shown that the Central Differencing scheme is the only second-order scheme of the considered ones. The error in the face flux obtained by the Rhie-Chow procedure, used for solving Navier-Stokes equations, is of second-order on uniform meshes, but it becomes a first-order method on non-uniform ones, which is a consequence of the discretisation procedure

for the surface-normal gradients required by the pressure equation. The accuracy of the diffusion term was found to be dependent on the linear interpolation scheme for skewed meshes, used for interpolation of the diffusion coefficient field onto the mesh faces, and on the discretisation procedure for the surface-normal gradients of the field under consideration. The accuracy of the source-term approximation is dependent on the linearisation procedure employed, and it is of second-order when the source term is linear.

8.2 Error Estimation

A novel approach to $a - posteriori$ error estimation for spatial terms has been proposed in Chapter 4. The Face Residual Error Estimate (FREE) estimates the error on the mesh faces, and thus provides information about the error magnitude in absolute terms, and which faces of a cell are the main contributors to the error. The analysis of the error on the faces has shown that the error may be transported in the opposite direction of the solution fluxes, and this is believed to be the reason why it is not possible to solve the error transport equation using the second-order FV procedure and achieve accurate estimation of the error.

The estimator was tested on a set of cases with analytical solutions. Test cases ranged from convection to diffusion-dominated ones. The accuracy of the estimator was satisfactory only for the convection-dominated problem. It over-estimates the error for problems where it is a consequence of diffusion-term disretisation because its truncation error is not consistent with that for the surface-normal-gradient term.

8.3 Mesh Adaptation

In this study two different adaptation techniques were proposed. A technique utilising the directional property of the FREE for driving a mesh-refinement procedure for hexahedral meshes was presented in Chapter 5. An adaptive-mesh-refinement technique working on unstructured meshes of completely arbitrary topology was described in Chapter 7.

The algorithm presented in Chapter 5 is a $h-$refinement procedure which is based on splitting the hexahedral cells parallel to the faces with large error. Additional safety criteria were introduced for preserving mesh quality and for making

the reduction of error monotonic. It is not allowed to insert split-hexahedron cells in the regions where the error may increase above the required upper limit; because the distance between the neighbouring nodes of the split-face is higher than it was before the refinement was performed, thus making the error higher there. This is of greatest merit for problems where high-aspect-ratio cells occur such that it preserves mesh quality and improves the rate of error reduction.

The procedure was tested in Chapter 5 with variable success. Quick error reduction and savings in the number of cells were achieved when the error was localised to some small part of the domain and highly directional. A convection-dominated case where the solution was generated by a source, given by an analytical expression, was found very sensitive to non-uniformity of the mesh, which causes problems with boundedness of the source-term-volume integral and the result of it is the increased error in the solution.

Further testing was performed in Chapter 6 on flows in complex geometries ranging from laminar to turbulent ones. It has produced solutions which are in good agreement with the benchmark solutions for laminar flows, but the final meshes had much smaller numbers of cells when compared to the benchmark meshes. The calculation using a low-Re turbulence model has produced a solution which is also in good agreement with the benchmark, but has only 3.6% of the number of the cells of the benchmark mesh. This happens because the refinement was highly directional near the walls. The calculation using a high-Re $k-\epsilon$ turbulence model produced a similar pattern of refinement in the near-wall regions, but the refinement of near-wall cells was blocked in the direction normal to the wall for $Y^+ < 50$ even though the errors there were still above the required upper limit. Analyses of face errors conducted for a few cases have shown that the refinement procedure also generates faces with low error which could be removed from the mesh using a coarsening procedure.

The method presented in Chapter 7 is based on point addition into the Delaunay structure. When the distribution of the discretisation error is known, points are introduced in the centroids of faces having large error. When the Delaunay structure is enriched with the new points, a new mesh is created. The method was found capable of reducing the estimated error below the required limit, but without any savings in the number of cells compared to uniform refinement. This is attributed to the fact that most cells which appear in the mesh are not of desirable shapes from

the aspect of accuracy.

8.4 Mesh Generation

The mesh generator developed during this study, and presented in Chapter 7, generates polyhedral meshes from Voronoi Polyhedra which are cut near the boundary to form a closed mesh. It can generate meshes for arbitrary 2D geometries, but it still cannot produce meshes for arbitrary 3D geometry.

The comparison of accuracy on polygonal, triangular and quadrilateral meshes was performed in Chapter 7. Those tests have shown that the maximum errors were at the same level for quadrilateral and polygonal meshes for the same number of cells, but the average error was smaller on quadrilateral meshes than on polygonal ones. Triangular meshes were the least accurate in all tests. These results are in agreement with the analysis performed in Section 3.6.

8.5 Future Work

In author's opinion, the results of this work can be further improved in the following areas:

- **Error Estimation.** The face residual for the diffusion term should be reformulated such that it becomes consistent with the truncation error for the diffusion term. This would remove the problem of error over-estimation for diffusion-dominated problems. Including the error coming from the source terms approximations into the face residual would improve the accuracy of error estimation but it would also require approximations of the moments of inertia for every face, which is not a trivial task when the cells are of arbitrary shape. The estimator should also be further tested on more cases to get a better insight into the behaviour of the face residual and the discretisation error, which could result in the improved normalisation factor.

- **Adaptive Mesh Refinement** can be improved further by adding the possibility of checking and controlling uniformity, which would not allow large differences between the sizes of neighbouring cells. This can be achieved by selecting more cells for refinement than required by the estimator until the

jump in cell sizes is smaller than some threshold value, or by incorporating r-refinement which would reposition mesh vertices such that the cell size varies smoothly. Coarsening would improve the cell count further, but it could only be allowed where the split-hexahedra generated by the coarsening procedure would not reduce the accuracy achieved by the refinement procedure. This is a very complicated task which could only be achieved if the coarsening could control the topology of the mesh.

Currently, the refinement procedure can only operate on hexahedra and split-hexahedra. Thus, it should be modified such that it is able to cope with all types of polyhedra which may appear in the mesh generated by a commercial hexahedra-dominant mesh generator. This would improve the usability of the procedure in the engineering environment.

It is also believed that the performance of the procedure would improve by implementing a Wall-function which would be valid for the whole range of Y^+ values, and would not impose a limit on the mesh resolution in the near-wall region.

- **Adaptive-Polyhedral-Mesh Generation** would become more efficient if the locations of the new points inserted into the Delaunay structure were chosen such that the resulting mesh is more regular. Coarsening can also be implemented by removing points from regions of low error, but care has to be taken that the grading between the fine and coarse mesh is smooth.

- **Polyhedral-Mesh Generator** should be improved such that it can generate meshes for arbitrary 3D geometries. This can be achieved by adding support for arbitrary surfaces, which would provide information about intersections of the Voronoi Polyhedra with the boundary surface. The ability to generate layers of prismatical cells near wall boundaries would enable users to perform calculation with low-Re turbulence models efficiently.

References

[1] Adjerid, S., Belguendouz, B., and Flaherty, J. E.: "A posteriori Finite Element Error Estimation for diffusion problems", *SIAM J. Sci. Comput.*, 21:728–746, 1999.

[2] Aftosmis, M. J.: "Upwind Method for Simulation of Viscous Flow on Adaptively Refined Meshes", *AIAA J.*, 32:268–277, 1994.

[3] Ainsworth, M.: "*A-posteriori* error estimators for second-order elliptic systems: Part 1. Theoretical foundations and *a-posteriori* error analysis", *Computers Math. Applic.*, 25(2):101–113, 1993.

[4] Ainsworth, M. and Oden, J. T.: "A procedure for *a-posteriori* error estimation for $h - p$ Finite Element methods", *Comput. Methods Appl. Mech. Engrg.*, 101:73–96, 1992.

[5] Ainsworth, M. and Oden, J. T.: "*A-posteriori* error estimators for second-order elliptic systems: Part 2. An optimal order process for calculating self-equilibrating fluxes", *Computers Math. Applic.*, 26(9):75–87, 1993.

[6] Ainsworth, M. and Oden, J. T.: "A posteriori error estimation in finite element analysis", *Comput. Methods Appl. Mech. Engrg.*, 142:1–88, 1997.

[7] Ainsworth, M. and Oden, J. T.: "A posteriori error estimators for the Stokes and Oseen equations", *SIAM J. Numer. Anal.*, 34:228–245, 1997.

[8] Ainsworth, M. and Oden, J.T.: "A unified approach to *a-posteriori* error estimation using Element Residual methods", *Numerische Mathematik*, 65:23–50, 1993.

[9] Ait-Ali-Yahia, D., Baruzzi, G., Habashi, W. G., Fortin, M., Dompierre, J., and Vallet, M. G.: "Anisotropic mesh adaptation: towards user-independent,

mesh-independent and solver-independent CFD. Part II: Structured grids", *Int. J. Numer. Meth. Fluids*, 39:657–673, 2002.

[10] Ait-Ali-Yahia, D., Habashi, W. G., Tam, A., Vallet, M. G., and Fortin, M.: "A directionally adaptive methodology using and Edge-Based Error Estimate on quadrilateral grids", *Int. J. Numer. Meth. Fluids*, 23:673–690, 1996.

[11] Almeida, R. C., Feijoo, R. A., Galeao, A. C., Padra, C., and Silva, R. S.: "Adaptive finite element computational fluid dynamics using and anisotropic error estimator", *Comput. Methods Appl. Mech. Engrg.*, 182:379–400, 2000.

[12] Angermann, L.: "The Finite Volume Method with Quadratic Basis Functions", *Computing*, 55:305–323, 1995.

[13] Angermann, L., Knabner, P., and Thiele, K.: "An error estimator for a finite volume discretization of density driven flow in porous media", *Applied Numerical Mathematics*, 26:179–191, 1998.

[14] Apel, T., Grosman, S., Jimack, P. K., and Meyer, A.: "A new methodology for anisotropic mesh refinement based upon error gradients", *Applied Numerical Mathematics*, 50:329–341, 2004.

[15] Babuška and C., Rheinboldt W.: "*A-posteriori* error estimates for the Finite Element Method", *Int. J. Numer. Meth. Engng.*, 12:1597–1615, 1978.

[16] Baker, T. J.: "Mesh adaptation strategies for problems in fluid dynamics", *Finite Elements in Analysis and Design*, 25:243–273, 1997.

[17] Bank, R.E. and Weiser, A.: "Some A Posteriori Error Estimators for Elliptic Partial differential Equations", *Math. Comp.*, 44:283–301, 1985.

[18] Bank, R.E. and Welfert, B. D.: "*A-posteriori* error estimates for the Stokes problem", *SIAM J. Numer. Anal.*, 28:591–623, 1991.

[19] Bardina, J. J., Ferziger, J. H., and Rogallo, R. S.: "Effect of rotation on isotropic turbulence: computation and modelling", *J. Fluid Mech.*, 154:321–336, 1985.

[20] Basara, B.: "Latest trends in CFD: the perspective of AVL's CFD solver", In *Proceeding of the International User Meeting*, Advanced Simulation Technologies, Graz, Austria, October 14-15 2003.

[21] Becker, R., Heuveline, V., and Rannacher, R.: "An optimal control approach to adaptivity in computational fluid dynamics", *Int. J. Numer. Meth. Fluids*, 40:105–120, 2002.

[22] Berger, M. J. and Collela, P.: "Local adaptive mesh refinement for shock hydrodynamics", *J. Comput. Phys.*, 82:64–84, 1989.

[23] Berger, M. J. and Oliger, J.: "Adaptive mesh refinement for hyperbolic partial differential equations", *J. Comput. Phys.*, 53:484–512, 1984.

[24] Berger, M.J.: "On conservation at grid interfaces", *SIAM J. Numer. Anal.*, 24(5):967–984, 1987.

[25] Blacker, T. D. and Stephenson, M. B.: "Paving: A new approach to Automated Quadrilateral Mesh Generation", *Int. J. Numer. Meth. Engng.*, 32:811–847, 1991.

[26] Bonet, J. and Peraire, J.: "An Alternating Digital Tree (ADT) algorithm for 3D geomteric searching and intersection problems", *Int. J. Numer. Meth. Engng.*, 31:1–17, 1991.

[27] Boussinesq, J.: "Théorie de l'écoulement tourbillant", *Mem. par. div. Sav., Paris*, 23, 1877.

[28] Bowyer, A.: "Computing Dirichlet tessellations", *The Computer Journal*, 24:162–166, 1981.

[29] Brandt, A.: "Multi-level adaptive solutions to boundary-value problems", *Math. Comp.*, 31:333, 1977.

[30] Calvo, N. A. and Idelsohn, S. R.: "All-hexahedral element meshing: Generation of the dual mesh recurrent subdivision", *Comput. Methods Appl. Mech. Engrg.*, 182:371–378, 2000.

[31] Cass, R. J., Benzley, S. E., Meyers, R. J., and Blacker, T. D.: "Generalized 3-D Paving: An automated Quadrilateral Surface Mesh Generation algorithm", *Int. J. Numer. Meth. Engng.*, 39:1475–1489, 1996.

[32] Cavendish, J. C., Field, D. A., and Frey, W. H.: "Automatic triangluation of arbitrary planar domains for the Finite Element Method", *Int. J. Numer. Meth. Engng.*, 8:679–696, 1974.

[33] Cavendish, J. C., Field, D. A., and Frey, W. H.: "An approach to automatic Three-dimensional Finite Element mesh generation", *Int. J. Numer. Meth. Engng.*, 21:329–347, 1985.

[34] Cebeci, T. and Smith, A.M.O.: *Analysis of turbulent boundary layers*: Academic Press, 1974.

[35] Chang, S. and Haworth, D. C.: "Adaptive grid refinement using cell-level and global imbalances", *Int. J. Numer. Meth. Fluids*, 24:375–392, 1997.

[36] Chen, W.L., Lien, F.S., and Leschziner, M. A.: "Local mesh refinement within a multi-block structured-grid scheme for general flows", *Comput. Meth. Appl. Mech. Engrg*, 144:327–369, 1997.

[37] Davidson, L.: "A pressure correction methods for unstructured meshes with arbitrary control volumes", *Int. J. Numer. Meth. Fluids*, 22:265–281, 1996.

[38] Demkowicz, L., Oden, J.T., Rachowicz, W., and Hardy, O.: "Toward a universal $h-p$ adaptive Finite Element strategy: Part 1: Constrained approximation and data structure", *Comput. Methods Appl. Mech. Engrg.*, 77:79–112, 1989.

[39] Dompierre, J., Vallet, M. G., Borgault, Y., Fortin, M., and Habashi, W. G.: "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III: Unstructured meshes", *Int. J. Numer. Meth. Fluids*, 39:675–702, 2002.

[40] Dompierre, J., Vallet, M. G., Fortin, M., Bourgault, Y., and Habashi, W. G.: "Anisotropic mesh adaptation: Towards a solver and user independent CFD": 35th Aerospace Sciences Meeting and Exhibit, January 6-10, 1997 / Reno, NV.

[41] Dukowicz, J. K., Cline, M. C., and Addesso, F. L.: "A General Topology Godunov Method", *J. Comput. Phys.*, 82:29–63, 1989.

[42] Dwyer, H. A.: "Grid Adaptation for Problems in Fluid Dynamics", *AIAA J.*, 22:1705–1712, 1984.

[43] Dwyer, H. A., Kee, R. J., and Sanders, B. R.: "Grid Adaptation for Problems in Fluid Dynamics", *AIAA J.*, 18:1205–1212, 1980.

[44] "ERCOFTAC : European Research Community on Flow, Turbulence and Combustion, Portal to Fluid Dynamics Database Resources": http://ercoftac.mech.surrey.ac.uk/.

[45] Erlebacher, G. and Eiseman, P. R.: "Adaptive Triangular Mesh Generation", *AIAA J.*, 25:1356–1364, 1987.

[46] Favre, A.: "Equations des gaz turbulents compressibles", *J. de Mécanique*, 4:361, 1965.

[47] Ferziger, J. H. and Perić, M.: "Further discussion of numerical errors in CFD", *Int. J. Numer. Meth. Fluids*, 23:1263–1274, 1996.

[48] Ferziger, J.H. and Perić, M.: *Computational methods for fluid dynamics*: Springer Verlag, Berlin-New York, 1995.

[49] Franklin, P., Gosman, A. D., and Kaludzercic, B.: "Development of a Parallel PACE Code", Periodic report, Imperial College, 1993.

[50] Gaskell, P. H. and Lau, A. K. C.: "Curvature-compensated convective transport: SMART, a new boundedness-preserving transport algorithm", *Int. J. Numer. Meth. Fluids*, 8:617–641, 1988.

[51] George, P. L.: "Improvements on Delaunay-based three-dimensional automatic mesh generator", *Finite Elements in Analysis and Design*, 25:297–317, 1997.

[52] George, P. L., Hecht, F., and Saltel, E.: "Automatic parallel generator with specified boundary", *Comput. Methods Appl. Mech. Engrg.*, 92:269–288, 1991.

[53] George, P. L. and Hermeline, F.: "Delaunay's mesh of a convex polyhedron in dimension d. Application to arbitrary polyhedra", *Int. J. Numer. Meth. Engng.*, 33:975–995, 1992.

[54] Gibson, M. M. and Dafa'Alla, A. A.: "Two-Equation Model for Turbulent Wall Flow", *AIAA J.*, 33(8):1514–1518, 1995.

[55] Gnoffo, P. A.: "A Finite-Volume, Adaptive Grid Algorithm Applied to Planetary Entry Flowfields", *AIAA J.*, 21:1249–1254, 1983.

[56] Habashi, W. G., Dompierre, J., Bourgault, Y., Ait-Ali-Yahia, D., Fortin, M., and Vallet, M. G.: "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles", *Int. J. Numer. Meth. Fluids*, 32:725–744, 2000.

[57] Hawken, D. F., Gottlieb, J. J., and Hansen, J. S.: "Review of Some Adaptive Node-Movement Techniques in Partial Differential Equations", *J. Comput. Phys.*, 95:254–302, 1991.

[58] Haworth, D.C., El Tahry, S.H., and Huebler, M.S.: "A global approach to error estimation and physical diagnostics in multidimensional fluid dynamics", *Int. J. Numer. Meth. Fluids*, 17(1):75–97, 1993.

[59] Haworth, D.C. and Jansen, K.: "Large-eddy simulation on unstructured deforming meshes: toward reciprocating IC engines", *Comp. Fluids*, 29:493–524, 2000.

[60] Hestens, M.R. and Steifel, E.L.: "Method of conjugate gradients for solving linear systems", *Journal of Research*, 29:409–436, 1952.

[61] Hinze, J.O.: *Turbulence*: McGraw-Hill, 1975.

[62] Hirsch, C.: *Numerical computation of internal and external flows*: John Willey & Sons, 1991.

[63] Ignat, L., Pelletier, D., and Ilinca, F.: "A universal formulation of two-equation models for adaptive computation of turbulent flows", *AIAA J.*, 33:2058–2065, 1995.

[64] Ilinca, A., Camarero, R., Trepanier, J. Y., and Reggio, M.: "Error Estimator and Adaptive Moving Grids for Finite Volume Schemes", *AIAA J.*, 33:2058–2065, 1995.

[65] Ilinca, F. and Pelletier, D.: "Positivity Preservation and Adaptive Solution for the $k-\epsilon$ Model of Turbulence", *AIAA J.*, 36:44–50, 1998.

[66] Ilinca, F., Pelletier, D., and Garon, A.: "An Adaptive Finite Element Method for a Two-equation Turbulence Model in Wall-bounded flows", *Int. J. Numer. Meth. Fluids*, 24:101–120, 1997.

[67] Issa, R.I.: "Solution of the implicitly discretized fluid flow equations by operator-splitting", *J. Comput. Phys.*, 62:40–65, 1986.

[68] Jacobs, D.A.H.: "Preconditioned Conjugate Gradient methods for solving systems of algebraic equations": Central Electricity Research Laboratories, 1980.

[69] Jasak, H.: *Error analysis and estimation in the Finite Volume method with applications to fluid flows*, PhD thesis, Imperial College, University of London, 1996.

[70] Jasak, H.: "Private communication", 2003: Imperial College, 2003.

[71] Jasak, H. and Gosman, A. D.: "Automatic Resolution Control for the Finite-Volume Method, Part 1: A-posteriori error estimates", *Num. Heat. Trans., Part B*, 38:237–256, 2000.

[72] Jasak, H. and Gosman, A. D.: "Automatic Resolution Control for the Finite-Volume Method, Part 2: Adaptive mesh refinement and coarsening", *Num. Heat. Trans., Part B*, 38:257–272, 2000.

[73] Jasak, H. and Gosman, A. D.: "Automatic Resolution Control for the Finite-Volume Method, Part 3: Turbulent flow applications", *Num. Heat. Trans., Part B*, 38:273–290, 2000.

[74] Jasak, H. and Gosman, A. D.: "Residual error estimate for the finite volume method", *Num. Heat. Trans., Part B*, 39:000–000, 2001.

[75] Jasak, H. and Gosman, A. D.: "Element residual error estimate for the finite volume method", *Comp. Fluids*, 32:223–248, 2003.

[76] Jasak, H., Weller, H. G., and Gosman, A. D.: "High Resolution NVD Differencing Scheme for Arbitrarily Unstructured Meshes", *Int. J. Numer. Meth. Fluids*, 31:431–449, 1999.

[77] Jin, H. and Prudhomme, S.: "A posteriori error estimation of steady-state finite element solutions of the Mavier-Stokes equations by a subdomain residual method", *Comput. Methods Appl. Mech. Engrg.*, 159:19–48, 1998.

[78] Jin, H. and Tanner, R. I.: "Generation of unstructured tetrahedral meshes by Advancing Front Technique", *Int. J. Numer. Meth. Engng.*, 36:1805–1823, 1993.

[79] Joe, B.: "Delaunay versus Min-max solid angle triangulations for Three-dimensional mesh generation", *Int. J. Numer. Meth. Engng.*, 31:987–997, 1991.

[80] Kaludzercic, B.: *Parallelisation of Eulerian and Lagrangian CFD Algorithms*, PhD thesis, Imperial College, University of London, 2000.

[81] Kelly, D. W.: "The self-equilibration of residuals and complementary a-postriori error estimates in the Finite Element method", *Int. J. Numer. Meth. Engng.*, 20:1491–1506, 1984.

[82] Knupp, P. M.: "Applications of mesh smoothing: copy, morph and sweep on unstructured quadrilateral meshes", *Int. J. Numer. Meth. Engng.*, 45:37–45, 1999.

[83] Lakehal, D. and Rodi, W.: "Calculation of the flow past a surface-mounted cube with two-layer turbulence models", *Journal of Wind Engineering and Industrial Aerodynamics*, 67 and 68:65–78, 1997.

[84] Launder, B.E. and Sharma, B.I.: "Application of the energy-dissipation model of turbulence to the calculation of a flow near a spinning disk", *Letters in Heat and Mass Transfer*, 1:131–138, 1974.

[85] Launder, B.E. and Spalding, D.B.: "The numerical computation of turbulent flows", *Comput. Methods Appl. Mech. Engrg.*, 3:269–289, 1974.

[86] Lee, M. J., Kim, J., and Moin, P.: "Structure of turbulence at high shear rate", *J. Fluid Mech.*, 216:561–583, 1990.

[87] Lehnhäuser, T. and Schäfer, M.: "Improved linear interpolation practice for finite-volume schemes on complex grids", *Int. J. Numer. Meth. Fluids*, 38:625–645, 2002.

[88] Leonard, B.P.: "Simple high-accuracy resolution program for convective modelling of discontinuities", *Int. J. Numer. Meth. Fluids*, 8:1291–1318, 1988.

[89] Li, T. S., Armstrong, C. G., and McKeag, R. M.: "Quad mesh generation for k-sided faces and hex mesh generation for trivalent polyhedra", *Finite Elements in Analysis and Design*, 26:279–301, 1997.

[90] Li, T. S., McKeag, R. M., and Armstrong, C. G.: "Hexahedral meshing using midpoint subdivision and integer programming", *Comput. Methods Appl. Mech. Engrg.*, 124:171–193, 1995.

[91] Lilek, Ž, Muzaferija, S., Perić, M., and Seidl, V.: "An implicit Finite-Volume Method using nonmatching blocks of structured grid", *Num. Heat. Trans., Part B*, 97:385–401, 1997.

[92] Löhner, R.: "Automatic unstructured grid generators", *Finite Elements in Analysis and Design*, 25:111–134, 1997.

[93] Löhner, R., Camberos, J., and Merriam, M.: "Parallel unstructured grid generation", *Comput. Methods Appl. Mech. Engrg.*, 95:343–357, 1992.

[94] Ltd., Nabla: *FOAM – The Complete Guide*: Nabla Ltd., 2001: Available from http://www.nabla.co.uk/.

[95] Maliska, C. R., Flávio, J., and Vasconcellos, V.: "An unstructured finite volume procedure for simulating flows with moving fronts", *Comput. Methods Appl. Mech. Engrg.*, 182:401–420, 2000.

[96] Marchi, C. H. and Carvalho da Silva, A. F.: "Unidimensional Numerical Solution Error Estimation for Convergent Apparent Order", *Num. Heat. Trans., Part B*, 42:167–188, 2002.

[97] Martinuzzi, R. an Tropea, C.: "The Flow Around Surface-Mounted, Prismatic Obstacles Placed in a Fully Developed Channel Flow", *J. Fluids Eng., Trans. ASME*, 115:85–92, 1993.

[98] Mavriplis, D. J.: "Adaptive Mesh Generation for Viscous Flows Using Delaunay triangulation", *J. Comput. Phys.*, 90:271–291, 1990.

[99] Meshkat, S. and Talmor, D.: "Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh", *Int. J. Numer. Meth. Engng.*, 49:17–30, 2000.

[100] Mishev, I. D.: "Finite Volume Methods on Voronoi Meshes", *Numer. Methods Partial Differential Equations*, 14:193–212, 1998.

[101] Müller-Hannemann, M.: "Hexahedral Mesh Generation by successive Dual Cycle Elimination", *Engineering with Computers*, 17:269–279, 1999.

[102] Muzaferija, S.: *Adaptive Finite Volume method for flow prediction using unstructured meshes and multigrid approach*, PhD thesis, Imperial College, University of London, 1994.

[103] Muzaferija, S. and Gosman, A. D.: "Finite-Volume CFD Procedure and Adaptive Error Control Strategy for Grids of Arbitrary Topology", *J.Comput. Phys.*, 138:766–787, 1997.

[104] Nakahashi, K and Deiwert, G. S.: "Three-Dimensional Adaptive Grid Method", *AIAA J.*, 24:948–954, 1986.

[105] "Numerical Recipes Home Page": http://www.nr.com/.

[106] Oden, J.T. and Prudhomme, S.: "New approaches to error estimation and adaptivity for the Stokes and Oseen equations", *Int. J. Numer. Meth. Fluids*, 31:3–15, 1999.

[107] Oden, J.T., Stroubolis, T., and Devloo, P.: "Adaptive Finite Element methods for the analysis of Inviscid Euler flow: Part 1. Fast Refinement/Unrefinement and moving mesh methods for unstructured meshes", *Comput. Methods Appl. Mech. Engrg.*, 59:327–362, 1986.

[108] Oden, J.T., Wu, W., and Ainsworth, M.: "An *a-posteriori* error estimate for Finite Element approximations of the Navier-Stokes equations", *Comput. Methods Appl. Mech. Engrg.*, 111:185–202, 1994.

[109] Oden, J.T., Wu, W., and Legat, V.: "An $h-p$ adaptive strategy for Finite Element approximations of the Navier-Stokes equations", *Int. J. Numer. Meth. Fluids*, 20:831–851, 1995.

[110] Owen, S. J. and Saigal, S.: "H-Morph: an indirect approach to advancing front hex meshing", *Int. J. Numer. Meth. Engng.*, 49:289–312, 2000.

[111] Owen, S. J., Staten, M. L., Canann, S. A., and Saigal, S.: "Q-Morph: an indirect approach to advancing front quad meshing", *Int. J. Numer. Meth. Engng.*, 44:1317–1340, 1999.

[112] Panton, R.L.: *Incompressible flow*: John Wiley and Sons, 1984.

[113] Papanastasiou, T. C, Georgiou, G. C., and Alexandrou, A. N.: *Viscous fluid flow*: CRC Press, 1999.

[114] Patankar, S. V. and Spalding, D. B.: "A Calculation Procedure for Heat, Mass and Momentum transfer in Three-Dimensional Parabolic Flows", *Int. J. Heat Mass Transfer*, 15:1787–1806, 1972.

[115] Patankar, S.V.: *Numerical heat transfer and fluid flow*: Hemisphere Publishing Corporation, 1981.

[116] Pelletier, D.: "Adaptive Finite Element Computations of Complex Flows", *Int. J. Numer. Meth. Fluids*, 31:189–202, 1999.

[117] Pelletier, D. and Ilinca, F.: "Adaptive Remeshing for the $k - \epsilon$ Model of Turbulence", *AIAA J.*, 35:640–646, 1997.

[118] Pelletier, D., Ilinca, F., and Turgeon, E.: "An Adaptive Finite Element Method for Forced Convection", *Int. J. Numer. Meth. Fluids*, 25:803–823, 1997.

[119] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C.: "Adaptive Remeshing for Compressible Flow Computations", *J. Comput. Phys.*, 72:449–466, 1987.

[120] Perić, M.: *A Finite Volume method for the prediction of Three-dimensional fluid flow in complex ducts*, PhD thesis, Imperial College, University of London, 1985.

[121] Phai, N. van: "Automatic mesh generation with tetrahedron elements", *Int. J. Numer. Meth. Engng.*, 18:273–289, 1982.

[122] Piomelli, U.: "Large-eddy simulation: achievements and challenges", *Progress in Aerospace Sciences*, 35(10):335–362, 1999.

[123] Pirzadeh, S.Z.: "An adaptive unstructured grid method by grid subdivision, local remeshing, and grid movement": 14th Computational Fluid Dynamics Conference, June 28-July 1, 1999 / Norfolk , Virginia.

[124] Pope, S. B.: *Turbulent Flows*: Cambridge University Press, 2000.

[125] Price, M. A. and Armstrong, C. G.: "Hexahedral mesh generation by medial surface subdivision: Part 2. Solids with flat and concave edges", *Int. J. Numer. Meth. Engng.*, 38:3335–3359, 1995.

[126] Price, M. A., Armstrong, C. G., and Sabin, M. A.: "Hexahedral mesh generation by medial surface subdivision: Part 1. Solids with convex edges", *Int. J. Numer. Meth. Engng.*, 38:3335–3359, 1995.

[127] Prudhomme, S. and Oden, J. T.: "A posteriori error estimation and error control for finite element approximations of the time-dependent Navier-Stokes equations", *Finite Elements in Analysis and Design*, 33:247–262, 1999.

[128] Prudhomme, S., Oden, J. T., Westermann, T., Bass, J., and Botkin, M. E.: "Practical methods for *a-posteriori* error estimation in engineering applications", *Int. J. Numer. Meth. Engng.*, 56:1193–1224, 2003.

[129] Ramakrishnan, R., Bey, K. S., and Thornton, E. A.: "Adaptive Quadrilateral and Triangular Finite-Element Scheme for Compressible Flows", *AIAA J.*, 28(1):51–59, 1990.

[130] Rhie, C.M. and Chow, W.L.: "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation", *AIAA J.*, 21:1525–1532, 1983.

[131] Rogers, M. M. and Moin, P.: "The structure of the vorticity field in homogeneous turbulent flows", *J. Fluid Mech.*, 176:33–66, 1987.

[132] Sampl, P.: "Semi-structured Mesh Generation based on Medial Axis", *Engineering with Computers*, 17:234–248, 2001.

[133] Sandri, D.: "A posteriori estimators for mixed finite element approximations of a fluid obeying the power law", *Comput. Methods Appl. Mech. Engrg.*, 166:329–340, 1998.

[134] Scarborough, J. B.: *Numerical Mathematical Analysis*: Hopkins Press, Baltimore, 1958.

[135] Schlichting, H.: *Boundary Layer Theory*: Pergamon Press LTD and Verlag G. Braun, 1955.

[136] Schneiders, R. and Bünten, R.: "Automatic generation of hexahedral finite elements meshes", *Computer Aided Geometric Design*, 12:693–707, 1995.

[137] Shephard, M. S. and Georges, M. K.: "Automatic three-dimensional mesh generation by the Finite Octree Technique", *Int. J. Numer. Meth. Engng.*, 32:709–749, 1991.

[138] Smagorinsky, J.: "General circulation experiments with the primitive equations", *Mon. Weather Rev.*, 91:99, 1963.

[139] Tam, A., Ait-Ali-Yahia, D., Robichaud, M. P., Moore, M., Kozel, V., and Habashi, W. G.: "Anisotropic mesh adaptation for 3D flows on structured and unstructured grids", *Comput. Methods Appl. Mech. Engrg.*, 189:1205–1230, 2000.

[140] Taniguchi, N., Arakawa, C., and Kobayashi, T.: "Construction of a Flow-Simulating Method with Finite Volume Based on a Voronoi Diagram", *JSME International Journal*, 34(1):18–23, 1991.

[141] Taniguchi, N. and Kobayashi, T.: "Finite Volume Method on the unstructured grid system", *Comp. Fluids*, 19:287–295, 1991.

[142] Tattersall, P. and McGuirk, J.: "Evaluation of Numerical Diffusion effects in viscous flow calculations", *Comp. Fluids*, 23(1):177–209, 1994.

[143] Tautges, T. J.: "The generation of hexahedral meshes for assmebly geometry: survey and progress", *Int. J. Numer. Meth. Engng.*, 50:2617–2642, 2001.

[144] Tautges, T. J., Blacker, T., and Mitchell, S. A.: "The Whisker Weaving Algorithm: A connectivity-based method for constructing All-hexahedral Finite Element meshes", *Int. J. Numer. Meth. Engng.*, 39:3327–3349, 1996.

[145] Thompson, J. F., Soni, B. K., and Weatherill, N. P.: *Handbook of Grid Generation*: CRC Press, 1999.

[146] Thompson, M. C. and Ferziger, J. H.: "An Adaptive Multigrid Technique for the Incompressible Navier-Stokes Equations", *J. Comput. Phys.*, 82:94–121, 1989.

[147] Turgeon, E and Pelletier, D.: "Computation of Jet Impingement Heat Transfer by an Adaptive Finite Element algorithm", *AIAA 98-2585*.

[148] Van Der Vorst, H.A.: "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems", *SIAM J. Sci. Comput.*, 13(2):631–644, 1992.

[149] Versteeg, H. K. and Malalasekera, W.: *An introduction to computational fluid dynamics The finite volume method*: Longman, 1st edition, 1995.

[150] Vijayan, P and Kallinderis, Y.: "A 3D Finite Volume Scheme for the Euler Equations on Adaptive Tetrahedral Grids", *J. Comput. Phys.*, 113:249–267, 1994.

[151] Vilsmeier, R. and Hänel, D.: "Adaptive methods on unstructured grids for Euler and Navier-Stokes equations", *Comp. Fluids*, 22(4-5):485–499, 1993.

[152] Volker, J.: "A numerical study of a posteriori error estimators for convection-diffusion equations", *Comput. Methods Appl. Mech. Engrg.*, 190:757–781, 2000.

[153] Wang, Z. J. and Srinivasan, K.: "An adaptive Cartesian grid generation method for 'Dirty' geometry", *Int. J. Numer. Meth. Fluids*, 39:703–717, 2002.

[154] Weatherill, N. P.: "A method for generating irregular computational grids in Multiply Connected Domains", *Int. J. Numer. Meth. Fluids*, 8:181–197, 1988.

[155] Weatherill, N. P. and Hassan, O.: "Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints", *Int. J. Numer. Meth. Engng.*, 37:2005–2039, 1994.

[156] White, D. R. and Tautges, T. J.: "Automatic scheme selection for toolkit hex meshing", *Int. J. Numer. Meth. Engng.*, 49:127–144, 2000.

[157] Wilcox, D.C.: *Turbulence Modeling for CFD*: DCW Industries, Inc., 1st edition, 1993.

[158] Wilson, J.K and Topping, B.H.V.: "Parallel adaptive tetrahedral mesh generation by the advancing front technique", *Computers and Structures*, 68:57–78, 1998.

[159] Yerry, M. A. and Shephard, M. S.: "Automatic three-dimensional mesh generation by the Modified-octree Technique", *Int. J. Numer. Meth. Engng.*, 20:1965–1990, 1984.

[160] Yoshimura, S., Yoshitaka, W., and Genki, Y.: "Automatic mesh generation of quadrilateral elements unsing intelligent local approach", *Comput. Methods Appl. Mech. Engrg.*, 179:125–138, 1999.

[161] Zhang, X. D., Trépanier, J. Y., and Camarero, R.: "A posteriori error estimation for finite-volume solutions of hyperbolic conservation laws", *Comput. Methods Appl. Mech. Engrg.*, 185:1–19, 2000.

[162] Zhu, J. Z., Zienkiewicz, O. C., Hinton, E., and Wu, J.: "A new approach to the development of automatic quadrilateral mesh generation", *Int. J. Numer. Meth. Engng.*, 32:849–866, 1991.

[163] Zienkiewicz, O. C. and Taylor, R. L.: *The Finite Element method, vol 1: Basic formulation and linear problems*: McGraw-Hill, 4th edition, 1989.

[164] Zienkiewicz, O. C. and Zhu, J. Z: "A simple error estimator and adaptive procedure for practical engineering analysis", *Int. J. Numer. Meth. Engng.*, 24:337–357, 1987.

[165] Zienkiewicz, O. C. and Zhu, J. Z: "The Superconvergent Patch Recovery and a posteriori Error Estimates, Part 1: The Recovery Technique", *Int. J. Numer. Meth. Engng.*, 33:1331–1364, 1992.

[166] Zienkiewicz, O. C. and Zhu, J. Z: "The Superconvergent Patch Recovery and a posteriori Error Estimates, Part 2: Error Estimates and Adaptivity", *Int. J. Numer. Meth. Engng.*, 33:1365–1382, 1992.