Volume weighted interpolation for unstructured meshes in the finite volume method

R.V. Coetzee (M.Eng)

Thesis submitted for the degree Doctor of Philosophy in Engineering at North-West University

Promotor: Prof. C.G. de K. du Toit
Co-promotor: Dr. O. Ubbink

November 2005
Potchefstroom Campus

Acknowledgements

I am grateful to all the people who enabled me to undertake and complete this research.

Foremost among these is Dr. Onno Ubbink to whom I am truly indebted for his enthusiasm, continuous interest, guidance and support throughout the course of this research.

To my promoter, Professor Jat du Toit for always being there to share ideas, for his constant support and encouragement and for taking care of all the administrative work that goes hand in hand with post graduate research.

To Louis Le Grange for the use of the commercial code Flo++ as well as for the development work on the basic FTK code that was used as a platform for this research.

To my friends and colleagues Dr. Jan Hendrik Kruger and Dr. Rufus Neethling for the many fruitful technical discussions as well as great friendship that made it all that much easier. Together we endured and ultimately triumphed.

Financial support for this research was provided by Sasol for which I am grateful.

Finally, eternal thanks are due to my wife Meliza for her constant patience and love throughout and to my parents for their continual encouragement and support through many years of study.

When all is said and done all recognition goes to our heavenly Father for making it all possible.

Abstract

The finite volume method is widely used for the numerical simulation of fluid flow because of its rigorous local conservation properties and its compatibility with arbitrary unstructured meshes for meshing complex domains. Interpolation plays an integral role in the finite volume method. Variables are located at cell centres but are also required at other positions such as cell faces. Variable values at these positions must be interpolated from cell values. In this thesis volume weighted interpolation is introduced as an alternative method of interpolation for the finite volume method. The main advantage of volume weighted interpolation is that variables can be interpolated conservatively between overlapping meshes. The accurate evaluation of convective fluxes on complex meshes remains a central issue in the finite volume method. While existing convection schemes perform well on structured orthogonal meshes, the use of orthogonal meshes is limited to simple domains. The application of volume weighted interpolation for convection modelling is investigated in this thesis in order to improve solutions on skew and non-orthogonal meshes. The method involves the construction of three-point interpolation stencils orthogonal to cell faces. A conservative interpolation is performed between the original mesh and the orthogonal stencil cells. The stencil is then used for the interpolation of face values of variables. Test cases are presented to test the interpolation stencil by using high-resolution convection schemes. Promising results are obtained with the stencil on unstructured meshes. Volume weighted interpolation also finds application as a pre- and post-processing tool for the finite volume method. Examples are presented to demonstrate how volume fraction fields can be initialised for two-phase flow simulations. Volume weighted interpolation can be used as a post-processing tool to map results from one mesh onto another as well as to calculate mass flows through surfaces. The applications described and examples presented in this thesis establish the potential of volume weighted interpolation as a valuable tool for the finite volume method.

Uittreksel

Die eindige volume metode word algemeen gebruik vir die simulasie van vloei. Die rede hiervoor is dat die metode konserwatief op selvlak is en saam met komplekse roosters gebruik kan word. Interpolasie vervul 'n integrale rol in die eindige volume metode. Veranderlikes word gestoor en opgelos by die middelpunte van selle, maar ook benodig by ander posisies soos byvoorbeeld gesigsmiddelpunte van selle. Hierdie waardes word vanaf die beskikbare selwaardes verkry deur middel van interpolasie. In hierdie proefskrif word volume geweegde interpolasie as 'n alternatiewe vorm van interpolasie vir die eindige volume metode bekend gestel. Een van die belangrikste voordele van volume geweegde interpolasie is dat veranderlikes konserwatief tussen oorvleulende roosters geïnterpoleer kan word. Die akkurate berekening van konvektiewe vloede oor selgesigte vorm 'n belangrike aspek van die eindige volume metode. Konveksie skemas wat goed werk op ortogonale roosters kan nie sonder probleme op komplekse roosters toegepas word nie. Die gebruik van volume geweegde interpolasie om oplossings op komplekse roosters te verbeter word in hierdie proefskrif ondersoek. Die metode behels die konstruksie van ortogonale roosters op selgesigte. Die waardes van die rooster selle word deur middel van 'n konserwatiewe interpolasie vanaf die basis rooster bereken. Gesigwaardes word vanaf die ortogonale rooster sel waardes bereken. Toetsgevalle waar die volume geweegde interpolasie tegniek saam met hoë resolusie konveksie skemas gebruik word, toon dat die tegniek effektief gebruik kan word om oplossings op komplekse roosters te verbeter. Volume geweegde interpolasie kan ook tydens voor en naverwerking gebruik word. Voorbeelde word gegee van hoe die tegniek gebruik kan word om aanvangswaardes vir twee-fase vloei simulasies te bereken. Tydens naverwerking kan volume geweegde interpolasie effektief gebruik word om veranderlikes na ander roosters te interpoleer asook om die massavloed deur oppervlaktes te bereken. Die verskillende toepassings en toetsgevalle wat in die proefskrif bespreek word vestig die metode van volume geweegde interpolasie as 'n belangrike gereedskapstuk vir die eindige volume metode.

Table of contents

Table of contents (cont.)

Table of contents (cont.)

List of figures

List of figures (cont.)

List of tables

Nomenclature

A	Acceptor cell, Area
A_x	x Component of face area vector
A_y	y Component of face area vector
A_z	z Component of face area vector
a	Coefficient
CS	Control Surface
CV	Control Volume
c_f	Courant number for control volume face
D	Donor cell
d_{pr}	Regular cell projection length
E	Solution error
F	Convective flux
f	Face, Face centre, Function
f'	Intersection of cell centre connection line with face
G_f	Mass flow across face
h	Perpendicular distance for tetrahedron volume calculation
$\dot{m}$	Mass flow
N	Neighbour cell, Number of base cells overlapped by mapped cell
n	Number of control volume faces / neighbour cells
P	Central cell in computational molecule, Momentum
S_b	Constant part of linear source term
S_P	Coefficient of dependent variable in linearized source term
S_ϕ	Source term
t	Time
δt	Time step
U	Upwind cell
u, v, w	Cartesian velocity components
V	Volume
x, y, z	Cartesian coordinates

Nomenclature (cont.)

∂V	Control Surface
δx	Distance
ϕ	Dependent variable
$\tilde{\phi}$	Normalised variable
α	Volume fraction
β_f	Downwind weighting factor
β_m	Gamma scheme blending factor
ρ	Density
Γ	Diffusion coefficient
η	Temporal integration weighting factor
ψ	Total quantity

Vectors

$\vec{A},\vec{B},\vec{C}$	Vector used in volume overlapping algorithm
$\vec{A}_f$	Face area vector
$\vec{d}$	Vector connecting computational point and its neighbour
$d\vec{A}$	Outward pointing differential surface element vector
$\vec{F}$	Vector function
$\vec{u}$	Velocity vector

Subscripts

A	Acceptor cell
D	Donor cell
DC	Deferred correction
f	Face
i	Index
nb	Neighbouring cell
P	Central cell of computational molecule
R	Regular cell
U	Upwind cell
u,v,w	Cartesian velocity components
ϕ	Dependent variable

Nomenclature (cont.)

Superscripts

a	Analytical
H	Higher order
L	Lower order
n	Number of time steps
o	Original
old	Previous iteration / time step

Abbreviations

BD	Blended Differencing
CBC	Convection Boundedness Criterion
CD	Central Differencing
CFD	Computational Fluid Dynamics
CICSAM	Compressive Interface Capturing Scheme for Arbitrary Meshes
COPLA	Combination Of Piecewise Linear Approximation
DC	Deferred Correction
DD	Downwind Differencing
DWF	Downwind Weighting Factor
FCT	Flux Corrected Transport
FTK	Finite volume toolkit
FVM	Finite Volume Method
HR	High-Resolution
MUSCL	Monotonic Upwind Scheme for Conservation Law
NVD	Normalised Variable Diagram
OOP	Object-Oriented Programming
OPIS	Orthogonal Projection Interpolation Stencil
PISO	Pressure Implicit with Splitting of Operators
QUICK	Quadratic Upstream Interpolation for Convective Kinetics
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SLIC	Simple Line Interface Calculation
UD	Upwind Differencing
VOF	Volume Of Fluid
VWI	Volume Weighted Interpolation

1. Introduction

"To look is one thing. To see what you look at is another. To understand what you see is another. To learn from what you understand is something else. But to act on what you learn is all that really matters."

-Winston Churchill-

1.1 Background

The motion of fluids plays an integral role in the world we live in, from the thundering oceans to the air we breathe, it controls, conveys and powers. Computational fluid dynamics (CFD) is a *computational technology* that is used to predict fluid flow, heat transfer and related phenomena through computer simulation. Computational fluid dynamics is based on the solution of the equations for mass, momentum and energy conservation by means of numerical solution methods. These conservation or *transport* equations are mathematical models describing the physics of fluid dynamics, in the form of coupled and often non-linear partial differential equations, (Bird *et al.*, 2002).

The finite element (Reddy, 1993), finite difference (Anderson, 1995) and finite volume methods (Versteeg and Malalasekera, 1995) are examples of numerical methods used in simulations. The finite volume method (FVM) is well established and widely used for fluid flow simulations with many commercial CFD codes based on this methodology, for example *Star-CD*[1], *CFX*[2] and *Fluent*[3]. The finite volume method is used extensively for simulation purposes because of its inherent suitability to model fluid flow in that it uses a control volume methodology that is locally conservative. The finite volume method therefore ensures conservation of mass, momentum and energy on a control volume basis, thereby automatically ensuring conservation for the whole domain.

When solving a problem using the finite volume method, the geometry of interest is divided into a finite number of non-overlapping and contiguous control volumes or *cells*. The transport equations are integrated over each control volume by approximating the variation of flow properties between cell centres with piecewise profiles which are constructed to support physical transport mechanisms for example convection and diffusion (Ubbink, 1997). The

[1] *www.cd-adapco.com*
[2] *www.ansys.com*
[3] *www.fluent.com*

construction of these piecewise profiles is based on *interpolation* and forms a critical part of the overall modelling process. By applying the piecewise profiles to the integrated transport equations, an algebraic equation is obtained for each cell, expressing the value of the dependent variable at the centre of the cell, ϕ_P, in terms of the neighbouring cells as well as any source terms that are applicable for that cell. Eq. (1.1) is the general form of the discretised transport equation for a dependent variable ϕ.

$$a_P\phi_P = \sum_{nb} a_{nb}\phi_{nb} + S_\phi \tag{1.1}$$

In Eq. (1.1), ϕ is the dependent variable that is solved during the simulation, a the coefficient of ϕ and S_ϕ the source term. The subscript P denotes the central cell and nb the neighbour cells, together forming a *computational molecule* around the central cell P.

The discretised conservation equation for each cell in the mesh is assembled together to form a system of linear equations which can be solved with a direct or iterative matrix solver. The result is a field of numerical values for the dependent variable at discrete locations across the domain, namely the centres of each control volume. As a result of the non-linear and coupled nature of the equations, it is often necessary to update the coefficient matrix and source terms with the latest available information and iterate the solution until convergence is reached.

Figure 1.1 shows a control volume with a fixed position in the three-dimensional Cartesian coordinate system. Fluid flows through the control surface around the control volume as indicated by the streamlines crossing it. The conservation equation for a general flow quantity inside such a control volume is described in words as

$$\begin{pmatrix} Rate\ of\ change \\ inside\ CV \end{pmatrix} + \begin{pmatrix} Convection\ out\ of \\ CV\ across\ CS \end{pmatrix} + \begin{pmatrix} Diffusion\ out\ of \\ CV\ across\ CS \end{pmatrix} = \begin{pmatrix} Sources \\ inside\ CV \end{pmatrix} \tag{1.2}$$

where CV and CS represent the control volume and control surface respectively. The first term on the left hand side of Eq. (1.2) is the rate of change term of the flow quantity inside the control volume, also known as the transient term. For transient simulations any imbalance in the remaining terms will result in an increase or a decrease of the flow quantity inside the control volume. The second and third terms in Eq. (1.2) are the convection and diffusion terms, which represent the net efflux across the control surface. Convective and diffusive fluxes are calculated across the control surface of each control volume. Sources are represented by the term on the right hand side of Eq.(1.2). Source terms account for the creation or destruction of the conserved flow quantity inside the control volume.

The value of a flow quantity for a control volume can only change if it is fluxed into or out of the control volume or if it is created or destroyed by sources or sinks. Eq. (1.2) establishes a balance between fluxes entering or leaving the control volume across the control surface and sources inside the control volume. The cell average value of a dependent variable is stored centrally for each cell in the finite volume method.

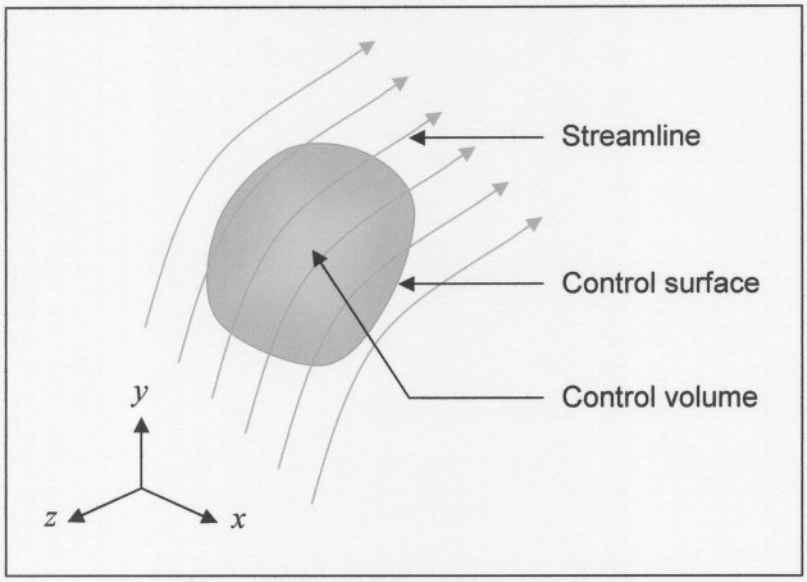


Figure 1.1: Arbitrary control volume

The earliest finite volume CFD codes were based on structured orthogonal meshes (Caretto *et al.*, 1972), (Patankar and Spalding, 1972). While these meshes are simple to construct and cells are referenced using straight forward indices, the application of orthogonal meshes are limited to simple geometries, limiting their usefulness. This shortcoming of the structured orthogonal mesh was addressed by extending the finite volume method for the inclusion of non-orthogonal structured meshes, (Peric, 1985). Modelling flows on non-orthogonal meshes increases the complexity of the various discretised conservation equations, however this development contributed to the usefulness of finite volume based CFD codes since the flow through complex geometries could now be solved, (Coelho and Peraira, 1993), (Choi *et al.*, 1993).

The next major milestone in the development of the finite volume method was the development of solution algorithms for arbitrary unstructured meshes Peric (1985). This allows the creation of meshes with arbitrary cell location and topology, providing the necessary flexibility to create the complex meshes required for most problems of engineering interest. For this reason algorithms based on arbitrary unstructured meshes became the method of choice for commercial CFD codes. Algorithms based on arbitrary unstructured meshes, uses lists to establish the connectivity between cells instead of the simple index systems used in codes for structured meshes.

Face addressing, is a programming methodology which is well suited for the implementation of the finite volume method on arbitrary unstructured meshes, (Ferziger and Peric, 1999). Face addressing considers the face between two cells as the primary mesh element. Fluxes are calculated across faces by expressing the face values of variables in the discretised transport equations in terms of the values of the variables at the centres of the cells bracketing the faces. By applying this methodology meshes can be constructed from arbitrary polyhedral cells as shown in Figure 1.2, where the control surface of a cell consists of a closed surface of faces, (Chow *et al.,* 1996). Polyhedral meshing for the finite volume method forms part of the latest advancements in commercial CFD codes (Peric, 2002).

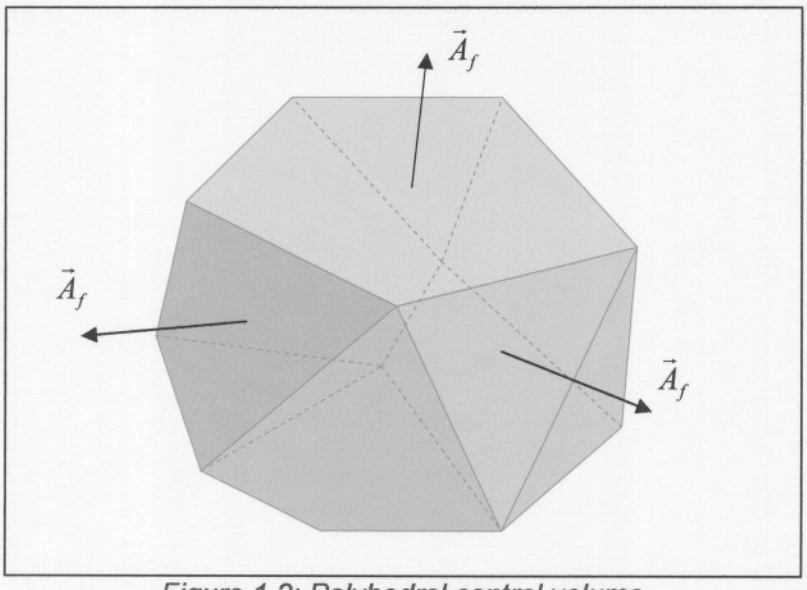


Figure 1.2: Polyhedral control volume

This thesis is based on the very general framework of the finite volume method where face addressing is used in conjunction with arbitrary unstructured meshes. While providing the flexibility to mesh complex domains, this approach does have its drawbacks as it allows the creation of meshes with poor characteristics. Some examples are non-orthogonal and skew meshes as well as warped control volume faces, which affects the accuracy and stability of the solution.

With face addressing, very limited information is available for each face where gradients and face values are required to model diffusion and convection respectively. Apart from geometrical information such as face area vectors, face node coordinates, etc. a face only has access to the values of the dependent variables at the centres of the cells bracketing that face. These values are required for interpolation purposes, i.e. for constructing the piecewise profiles required during discretisation to approximate the variation of variables between cell centres. Since interpolation plays a significant role in the finite volume method, the limited information available at cell faces remains a problem that needs to be addressed. Higher-order face value

interpolation for example, becomes a major issue when using face addressing algorithms on arbitrary unstructured meshes.

The general rule for mesh generation is to construct meshes that are as close to orthogonal as possible while keeping the skewness (conjunctional) error as small as possible, (Croft, 1998). When the line connecting the centres of two cells is parallel to the face area vector of the face located between the cells, the mesh is orthogonal. When this line lies at an angle in relation to the face area vector, the mesh is non-orthogonal. Figure 1.3 shows an example of an orthogonal and non-orthogonal mesh.

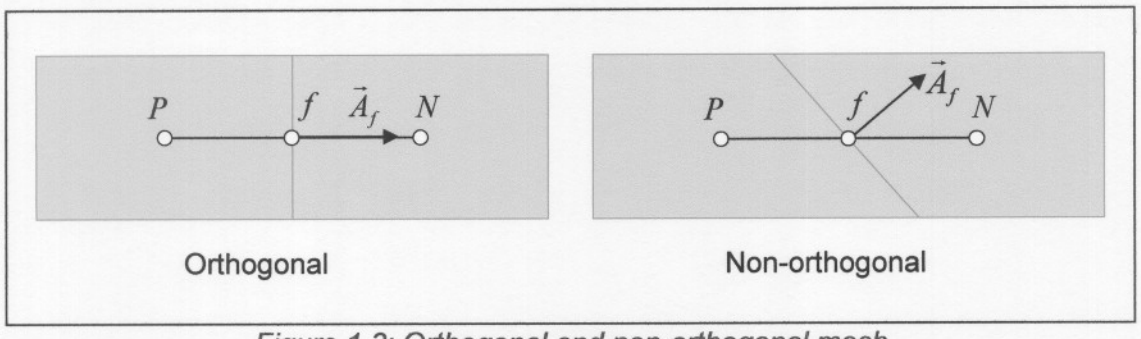


Figure 1.3: Orthogonal and non-orthogonal mesh

Additional terms are introduced into the discretised equations when using non-orthogonal meshes. The extra terms originate from the calculation of the gradient of a dependent variable across a control volume face, which is used in the discretised diffusion term modelled by means of Fick's law of diffusion (Bird *et al.*, 2002). The gradient cannot be calculated only in terms of the variable values at the cell centres of cells P and N, and requires the values of the neighbour cells of P and N to be taken into consideration as well. The contributions of these *skew neighbour* cells to the gradient are usually treated in an explicit manner while the contributions of cells P and N are treated implicitly (Jasak, 1996). This concept of utilising the information from additional cells in the calculation of face fluxes is important for the present work where the idea is extended to the modelling of convective transport.

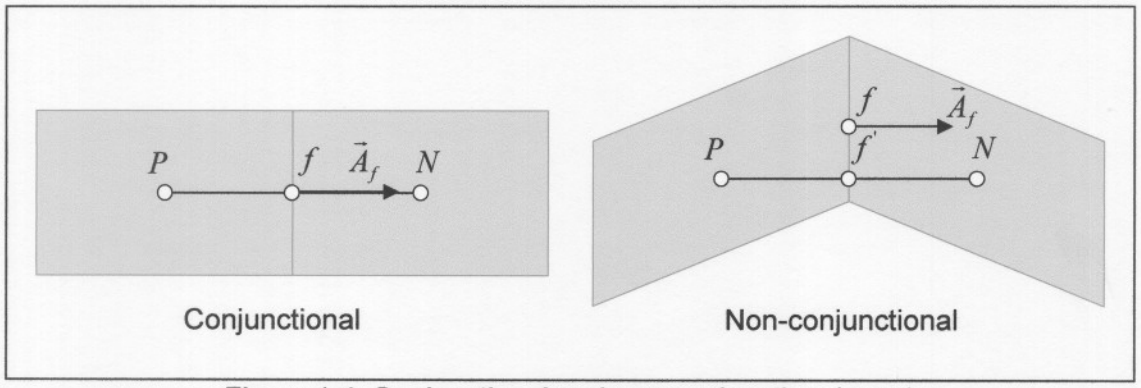


Figure 1.4: Conjunctional and non-conjunctional mesh

When the line connecting the centres of two cells passes through the centre of the face located between the cells, the mesh is conjunctional with a zero skewness error (Croft, 1998). When this line does not pass through the centre of the face located between the cells, the mesh is non-conjunctional. Figure 1.4 shows an example of a conjunctional mesh with zero skewness error and a non-conjunctional mesh.

Meshes that are both non-orthogonal and non-conjunctional are commonly found where complex domains are meshed. Figure 1.5 shows an example of a mesh that is both non-orthogonal and non-conjunctional. The degree of orthogonality, as well as the degree of skewness, can be used to gauge the general quality of a mesh. A mesh may have orthogonal and conjunctional regions as well as regions consisting of highly deformed cells.

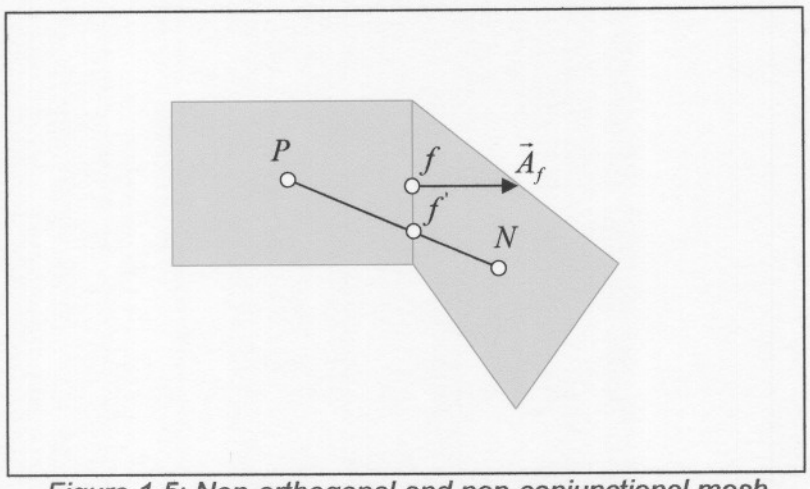


Figure 1.5: Non-orthogonal and non-conjunctional mesh

The choice of interpolation method plays an important role within the finite volume method and influences the outcome of simulations. Linear interpolation seems like a logical choice but cannot be used indiscriminately, (Versteeg and Malalasekera, 1995). Higher-order interpolation schemes are required for accurate solutions in many applications, for example when modelling segregated multiphase flows, (Ubbink, 1997). Volume weighted interpolation (VWI) is introduced in this thesis as an alternative interpolation methodology for the finite volume method. Its implementation is described and its uses investigated.

The most important advantage of volume weighted interpolation is that the interpolation of variables between overlapping meshes is *conservative*. Volume weighted interpolation is based on the concept that the value of a cell overlapping other cells in a mesh can be expressed in terms of the values associated with the cells that are overlapped, based on the volume fractions of the overlaps between the different cells. Figure 1.6 shows a cell overlapping cells in a mesh. The volume weighted cell value is calculated from the values of the base cells overlapped by the cell as well as the volume fractions of the overlaps between the cells. To perform volume

weighted interpolation the common volume between overlapping cells is required. An algorithm to perform these volume overlapping calculations is presented in this thesis.

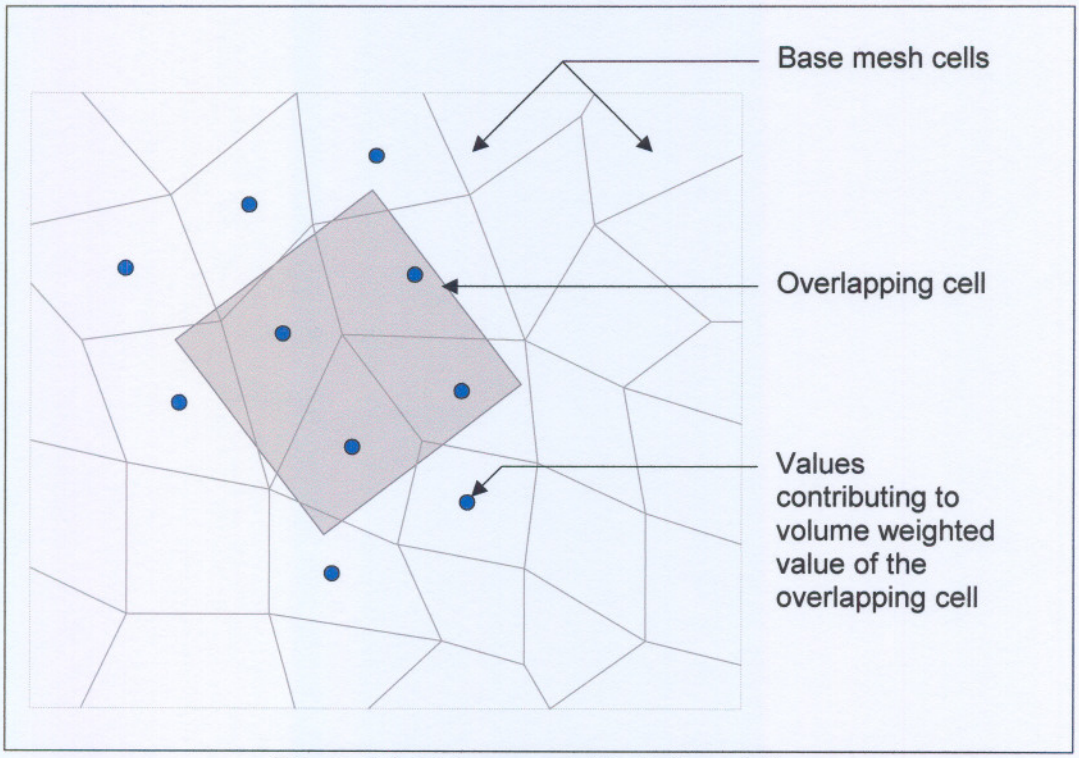


Figure 1.6: Volume weighted interpolation

The accurate evaluation of convective fluxes on complex meshes remains a challenge for CFD in general and the finite volume method in particular. Many different interpolation schemes or *differencing schemes* have been developed for the purpose of modelling convective transport with the finite volume method, yet the modelling of convective transport has been termed "embarrassingly difficult", "...computational dynamics' ultimate embarrassment..." (Leonard, 1991). The intricacies of convection modelling are often overlooked in favour of the use of differencing schemes that are stable but diffusive and which at least guarantees a solution, regardless of its accuracy. Accuracy, stability and boundedness of numerical simulations are essential and these characteristics are largely determined by the choice of convection differencing scheme. While an improvement in mesh quality will always improve the results of a simulation, mesh improvements are not always practical or possible. Volume weighted interpolation makes it possible to apply high-resolution differencing schemes, that perform very well on structured orthogonal meshes, to arbitrary unstructured meshes, thereby retaining the advantages of these schemes on complex meshes.

Additional applications of volume weighted interpolation within the finite volume method are also described in this thesis. One such application area is post-processing results from a simulation. Volume weighted interpolation can be utilised effectively in this area to analyse the results from

simulations and to calculate mass flows across arbitrary surfaces. During pre-processing, volume weighted interpolation is a useful tool for initialising complex fields of variables on structured or unstructured meshes.

1.2 Present contribution

There is a need for an improved face value interpolation method for arbitrary unstructured meshes in the finite volume method. The prediction of face values for convective transport has long been a contentious issue in the simulation of fluid flow. Interpolation stencils that work extremely well on orthogonal meshes are not available for arbitrary unstructured meshes. The need for more accurate face value interpolation methods for non-orthogonal and skew meshes is described in this thesis.

A novel convective flux calculation stencil, OPIS (Orthogonal Projection Interpolation Stencil), based on volume weighted interpolation, is presented for the finite volume method. OPIS involves the construction of meshes orthogonal to cell faces where an interpolated value is required. Volume weighted interpolation is used to interpolate variables conservatively from the base mesh to the constructed cells of the stencil. The constructed orthogonal meshes provide the required three-point stencils for higher-order flux calculations on arbitrary unstructured meshes. Results of scalar convection problems using high-resolution schemes are presented to demonstrate the capabilities of OPIS for simulations of convective transport. OPIS provides a natural extension of the finite volume method for the calculation of upwind cell values for three-point stencils that is required for higher-order interpolation of face values. The benefits derived from using stencils based on volume weighted interpolation for arbitrary unstructured meshes, is investigated and evaluated in this thesis. Interpolated face values are used in the evaluation of surface integrals while cell values are used in the evaluation of volume integrals. In an effort to integrate the transport equations as accurately as possible using numerical integration, the evaluation of volume and surface integrals should be as accurate as possible.

An additional benefit of volume weighted interpolation is that the techniques developed in this thesis can be used for both pre- and post-processing tasks. Performing post-processing tasks on unstructured meshes is complicated by the data structures used as well as the location of cell values in arbitrary locations. It is therefore cumbersome to create graphs of results along straight lines or curves across a mesh. Volume weighted interpolation provides a solution for these problems through the conservative mapping of results to meshes that are more suitable for the creation of graphs during post-processing. It is often required to calculate the mass flux across surfaces during post-processing. Volume weighted interpolation can be used effectively for this task by creating *surface meshes* that overlaps the computational mesh in the areas where mass flow results are sought. Momentum is interpolated conservatively from the original

mesh to the surface mesh. The velocities for the surface cells are calculated from the interpolated momentum values. Mass flows through the surfaces are then calculated from the interpolated velocities of the surface cells. Volume weighted interpolation can also be used to map initial variable distributions onto structured or unstructured meshes. These applications of volume weighted interpolation for pre- and post-processing are explained and demonstrated in detail in this thesis. Volume weighted interpolation is established in this thesis as a valuable tool for the finite volume method.

1.3 Outline of thesis

Chapter two presents the basis for volume weighted interpolation. This includes a description of the algorithm used to calculate the overlapping volume between cells and the data structures used to store all the relevant information required to perform volume weighted interpolation. Examples are presented to show how variables are interpolated conservatively between overlapping meshes. Chapter three describes the application of volume weighted interpolation to model convective transport and demonstrates the capabilities of the Orthogonal Projection Interpolation Stencil, by applying the stencil in scalar convection simulations on arbitrary unstructured meshes. The pre- and post-processing capabilities of volume weighted interpolation with examples to demonstrate its uses are presented in Chapter four. Chapter five summarises the research with conclusions as well as suggestions for future research opportunities in this field.

2. Volume weighted interpolation

2.1 Introduction

The concept of volume weighted interpolation (VWI) is introduced in this chapter. Examples are presented to illustrate how volume weighted interpolation works. A very important requirement for the application of volume weighted interpolation to any problem is the ability to calculate the common volume of two overlapping cells. The algorithm that is used for volume overlapping calculations is presented in this chapter. Furthermore, to apply volume weighted interpolation it is necessary to store the relevant volume overlapping information in appropriate data structures for easy access by the solver. These data structures are briefly described in this chapter. Also included in this chapter is a section on FTK (Finite volume toolkit), the object oriented research code that was used for all the simulation work in this thesis.

2.2 Volume Weighted Interpolation

2.2.1 Background and terminology

The terminology relating to volume weighted interpolation is explained by means of the following definitions:

Base cell — A base cell is a cell with a known value that is used to interpolate the value of a mapped cell overlapping the base cell.

Base mesh — A base mesh is a mesh with known cell values. Base mesh values are used for the interpolation of mapped cell values.

Mapped cell — A mapped cell overlaps one or more base cells in a base mesh. The mapped cell value is interpolated from the base cell values through volume weighted interpolation.

Mapped mesh — A mapped mesh consists of a number of mapped cells. The values of all the mapped cells in the mapped mesh are computed through volume weighted interpolation.

Common volume — When two cells overlap in three-dimensional space, the volume shared by both cells is referred to as the common volume of the two cells.

Volume overlap — When a mapped cell and a base cell overlaps, the volume overlap equals the common volume shared by the cells.

Volume fraction Volume fraction equals the volume overlap of a mapped cell and a base cell, divided by the total volume of the mapped cell. When a mapped cell is completely overlapped by base cells, the sum of the volume fractions of the mapped cell must be equal to one in order to satisfy volume conservation requirements.

Volume weighted interpolation is based on the principle that the volume fractions between overlapping cells can be used to interpolate or *map* variables conservatively between overlapping meshes. The volume overlap between cells is used to determine the contribution of a base cell to the mapped cell that is overlapping the base mesh. The interpolated value for the mapped cell is calculated by means of Eq. (2.1).

$$\phi_{mapped\ cell} = \sum_{n=1}^{N} (\alpha\phi)_n \tag{2.1}$$

In Eq.(2.1) ϕ is the conserved quantity to be interpolated, for example mass or momentum, α the volume fraction for base cell n and N the total number of base cells overlapping the mapped cell. When this calculation is repeated for every cell in a mapped mesh, the interpolated cell values of the mapped mesh is obtained.

The following three examples demonstrate the conservative interpolation of variables between overlapping meshes using volume weighted interpolation.

2.2.2 Volume weighted interpolation: Example 1

Figure 2.1 shows a mesh consisting of two similar base cells. Five cases are shown, each with a shaded region representing a mapped cell which overlaps the base mesh. The volume fractions with which the base cells overlap the mapped cell are also indicated in each case. The sum of the volume fractions equals one in all five cases. This is required when the mapped cell is located inside the bounding surface of the base mesh, therefore no boundary overlapping occurs.

In this example, base cell values of 50 and 100 were selected for cell one and two respectively. The calculated value for the mapped cell is a function of the volume overlapped by each base cell and the base cell values. The greater the volume of a mapped cell overlapped by a base cell, the greater the contribution of that base cell will be to the value of the mapped cell and vice versa. The mapped cell value is therefore a weighting between base cell values.

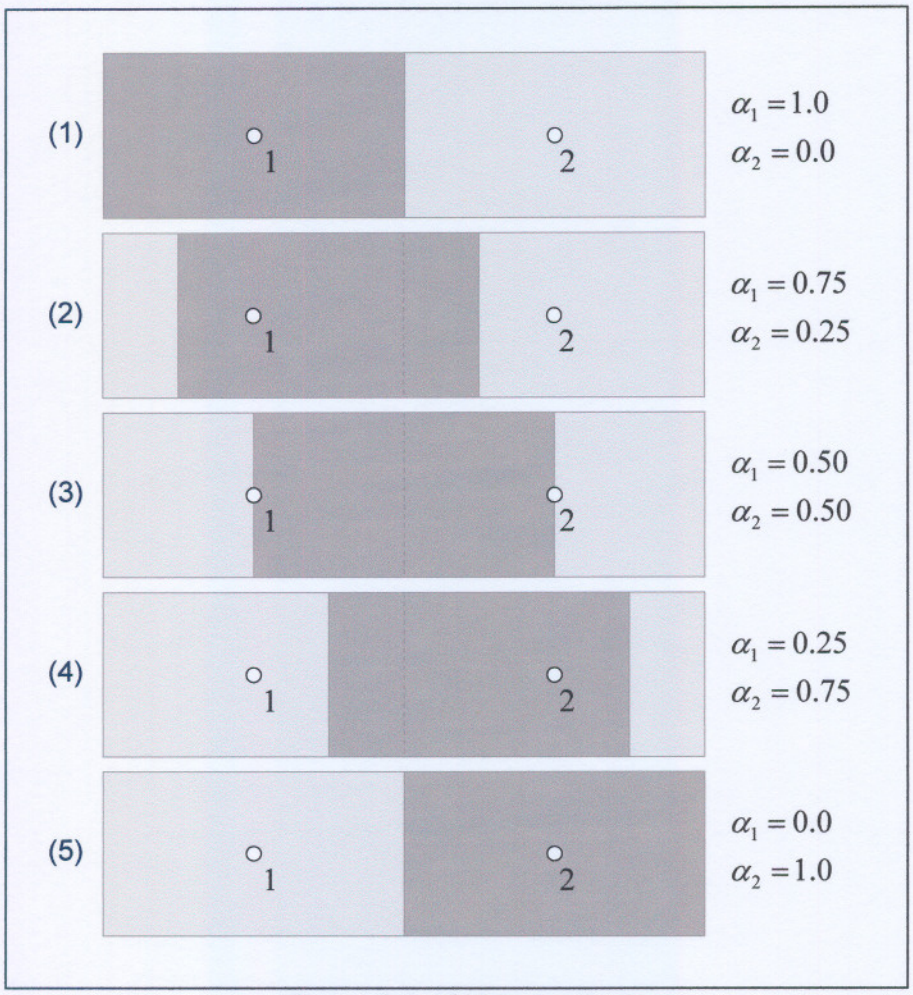


Figure 2.1: VWI example 1

The interpolated value of the mapped cell is calculated from the base cell values and volume fractions using Eq. (2.1):

Case 1:

$$\begin{aligned}\phi_{mapped\ cell} &= \alpha_1\phi_1 + \alpha_2\phi_2 \\ &= 1.0(50) + 0.0(100) \\ &= 50.0\end{aligned}$$

Case 2:

$$\begin{aligned}\phi_{mapped\ cell} &= \alpha_1\phi_1 + \alpha_2\phi_2 \\ &= 0.75(50) + 0.25(100) \\ &= 62.5\end{aligned}$$

Case 3:

$$\begin{aligned}\phi_{mapped\ cell} &= \alpha_1\phi_1 + \alpha_2\phi_2 \\ &= 0.5(50) + 0.5(100) \\ &= 75.0\end{aligned}$$

Case 4:

$$\begin{aligned}\phi_{mapped\ cell} &= \alpha_1\phi_1 + \alpha_2\phi_2 \\ &= 0.25(50) + 0.75(100) \\ &= 87.5\end{aligned}$$

Case 5:

$$\begin{aligned}\phi_{mapped\ cell} &= \alpha_1\phi_1 + \alpha_2\phi_2 \\ &= 0.0(50) + 1.0(100) \\ &= 100.0\end{aligned}$$

2.2.3 Volume weighted interpolation: Example 2

Figure 2.2 shows four base cells and a mapped cell overlapping the base cells. The volume fractions of the four base cells are also indicated, adding to one as required for volume conservation.

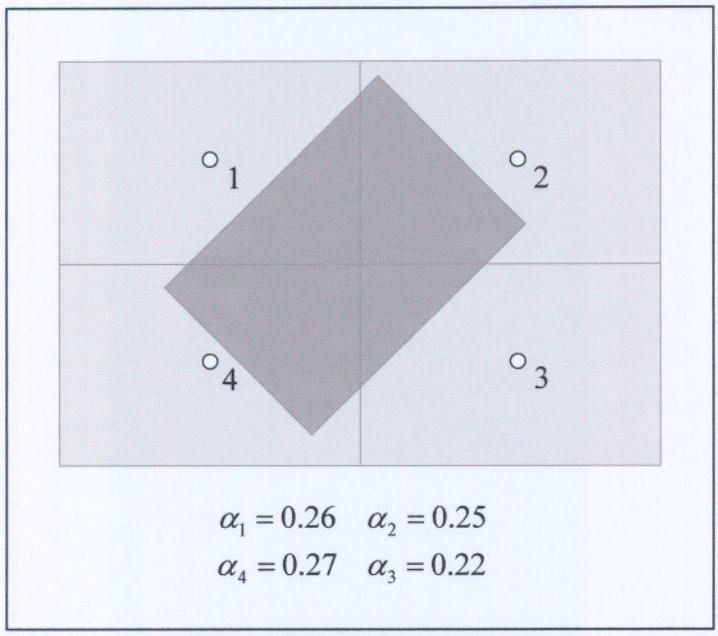


Figure 2.2: VWI example 2

When the base cell values are all equal, the interpolated mapped cell value should be equal to the base cell values. This is demonstrated by the following example.

$$\phi_1 = \phi_2 = \phi_3 = \phi_4 = \phi_{base\ cell}$$

$$\begin{aligned}\phi_{mapped\ cell} &= \alpha_1\phi_1 + \alpha_2\phi_2 + \alpha_3\phi_3 + \alpha_4\phi_4 \\ &= 0.26\left(\phi_{base\ cell}\right) + 0.25\left(\phi_{base\ cell}\right) + 0.22(\phi_{base\ cell}) + 0.27(\phi_{base\ cell}) \\ &= (0.26 + 0.25 + 0.22 + 0.27)(\phi_{base\ cell}) \\ &= \phi_{base\ cell}\end{aligned}$$

The interpolated value for the mapped cell equals the values of the base cells as required. In the following example different values are used for each of the four base cells.

$$\phi_1 = 50; \phi_2 = 70; \phi_3 = 90; \phi_4 = 60$$

$$\begin{aligned}\phi_{mapped\ cell} &= \alpha_1\phi_1 + \alpha_2\phi_2 + \alpha_3\phi_3 + \alpha_4\phi_4 \\ &= 0.26(50) + 0.25(70) + 0.22(90) + 0.27(60) \\ &= 66.5\end{aligned}$$

The value of 66.5 is therefore the volume weighted value of the mapped cell for the given volume fractions and base cell values. The interpolated value is bounded between the base cell values as required.

2.2.4 Volume weighted interpolation: Example 3

Two meshes with common boundaries and different cells are shown in Figure 2.3. For the purpose of this example the volume of each of the four orthogonal base cells is 0.2. The values of the base cells were assigned as follows:

$$\phi_1 = 20; \phi_2 = 40; \phi_3 = 30; \phi_4 = 80$$

The base cells do not have to be orthogonal and may be of any shape and orientation as long as the two meshes share the same boundaries.

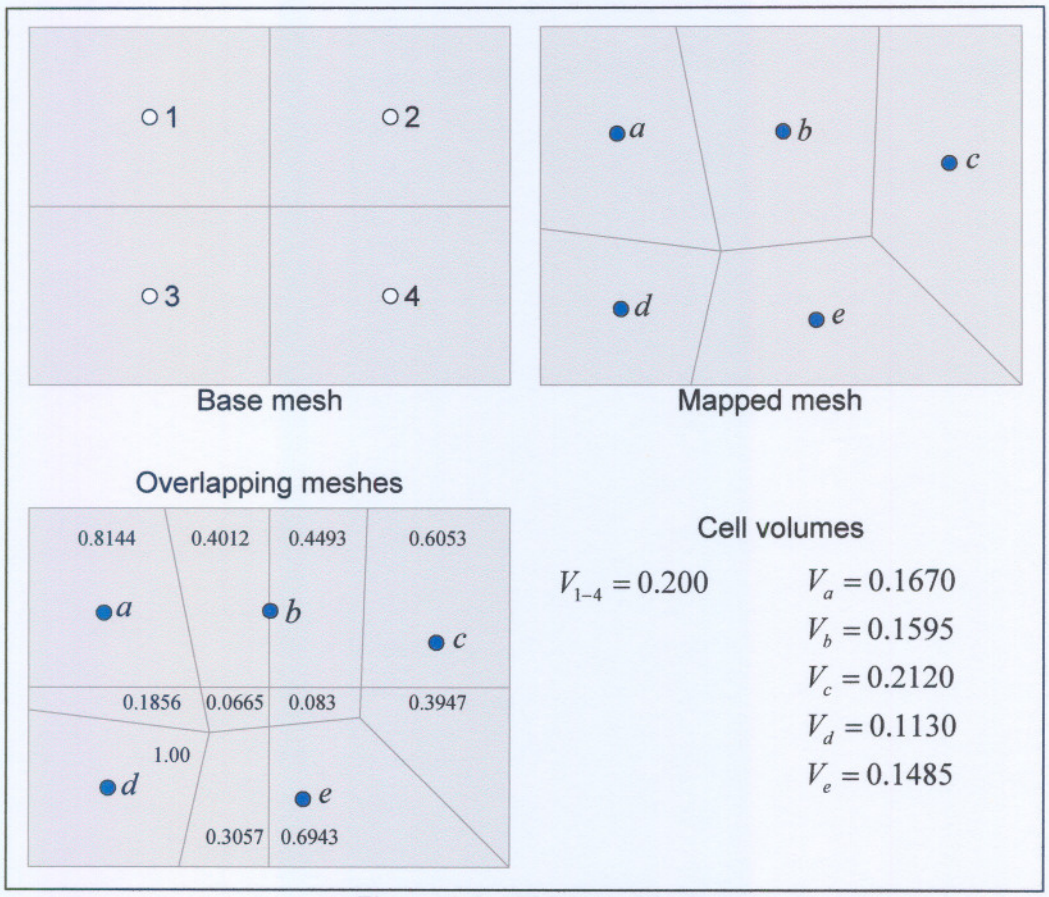


Figure 2.3: VWI example 3

The figure also shows the volume fractions for the base cells overlapping the cells of the mapped mesh. The sum of the volume fractions of each mapped cell equals one as required for volume conservation. The total quantity of the base mesh equals the sum of the base cell

values multiplied by the cell volume of each base cell. Given that the volume of each base cell equals 0.2, the total value for the mesh is calculated as:

$$\begin{aligned}\psi_{total} &= \phi_1 V_1 + \phi_2 V_2 + \phi_3 V_3 + \phi_4 V_4 \\ &= (20\times 0.2) + (40\times 0.2) + (30\times 0.2) + (80\times 0.2) \\ &= 34\end{aligned}$$

The total quantity of ψ contained in the base mesh where ϕ is expressed as the base cell value per unit volume and $\psi = \phi V$, is therefore 34. In this example where the boundaries of the two meshes coincides, the value of ψ may not change as a result of the interpolation. Any change in the value of ψ will indicate that the interpolation was not performed conservatively and that there was an increase or decrease in the conserved quantity after the interpolation was performed. This example is therefore a useful and reliable test for conservative interpolation between overlapping meshes.

The values of the mapped cells a to e are calculated by means of Eq. (2.1) using the volume fractions indicated in Figure 2.3.

$$\begin{aligned}\phi_a &= 0.8144\phi_1 + 0.1856\phi_3 \\ &= 21.85\end{aligned}$$

$$\begin{aligned}\phi_b &= 0.4012\phi_1 + 0.4493\phi_2 + 0.0665\phi_3 + 0.083\phi_4 \\ &= 34.63\end{aligned}$$

$$\begin{aligned}\phi_c &= 0.6053\phi_2 + 0.3947\phi_4 \\ &= 55.79\end{aligned}$$

$$\begin{aligned}\phi_d &= 1.00\phi_3 \\ &= 30.0\end{aligned}$$

$$\begin{aligned}\phi_e &= 0.3057\phi_3 + 0.6943\phi_4 \\ &= 64.71\end{aligned}$$

The volume of mapped cell d falls within the volume of base cell three. The volume fraction is therefore exactly one and the interpolated value of cell d equal to the value of base cell three. To determine if the interpolation was conservative, the total value for the mapped mesh is calculated by adding together the interpolated mapped cell values multiplied to the mapped cell volumes.

$$\psi_{total} = \phi_a V_a + \phi_b V_b + \phi_c V_c + \phi_d V_d + \phi_e V_e$$
$$= (21.85 \times 0.1670) + (34.63 \times 0.1595) + (55.79 \times 0.2120) + (30 \times 0.1130) + (64.71 \times 0.1485)$$
$$= 34$$

The total mapped mesh value is 34, equal to the total base mesh value; therefore a *conservative mapping* between the two meshes was performed. The same results will be obtained by mapping from the non-orthogonal mesh to the orthogonal mesh. With volume weighted interpolation a conservative mapping can be performed between any two meshes regardless of whether the meshes are orthogonal or non-orthogonal. The following section describes the algorithm that is used to calculate the volume fractions required for volume weighted interpolation.

2.3 Volume overlapping calculation

This section describes an algorithm for volume overlapping calculations between different cells. The algorithm follows a systematic approach to volume overlapping calculation which is accurate and can readily be implemented as a subroutine (du Toit, 2005). The algorithm is based on the notion that any polyhedral cell can be divided into a number of non-overlapping tetrahedrons. The tetahedron forms the basic element for volume calculations.

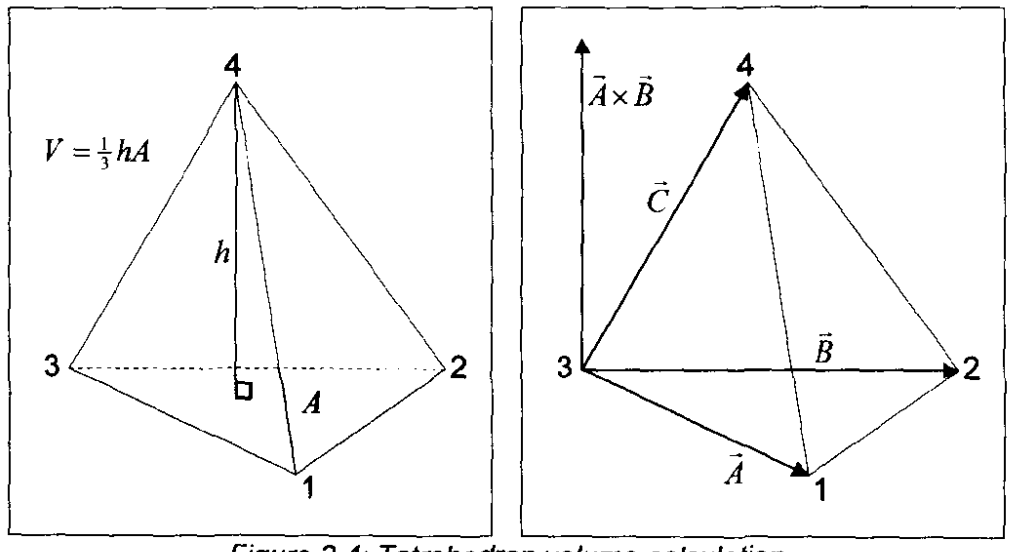


Figure 2.4: Tetrahedron volume calculation

The volume of a tetrahedron equals the surface area of a face multiplied by a third of the distance to the node opposite the face as shown in Figure 2.4. To facilitate volume calculation for tetrahedrons in the Cartesian coordinate system, the volume is expressed in vector notation. The surface area of the face, A, is obtained by calculating the vector cross product of the two

vectors forming the edges of the triangular surface. The absolute value of this cross product is divided by two to obtain the surface area, Eq. (2.2).

$$A = \tfrac{1}{2}\left|\vec{A}\times\vec{B}\right| \tag{2.2}$$

The cross product produces a vector normal to the surface and oriented in a direction which satisfies the *right hand rule*. The height, h, perpendicular to the surface A is obtained by calculating the scalar product between this vector and the vector $\vec{C}$ shown in Figure 2.4.

$$h = \left(\frac{\vec{A}\times\vec{B}}{\left|\vec{A}\times\vec{B}\right|}\right)\cdot\vec{C} \tag{2.3}$$

Since a third of the height h multiplied by the surface area A equals the volume of the tetrahedron, the volume equals a sixth of the scalar triple product, Eq.(2.4).

$$V = \tfrac{1}{6}\left(\vec{A}\times\vec{B}\right)\cdot\vec{C} \tag{2.4}$$

Vectors $\vec{A}$, $\vec{B}$ and $\vec{C}$ in Figure 2.4 are written in terms of the node coordinates as:

$$\begin{aligned}\vec{A} &= (x_1 - x_3, y_1 - y_3, z_1 - z_3)\\ \vec{B} &= (x_2 - x_3, y_2 - y_3, z_2 - z_3)\\ \vec{C} &= (x_4 - x_3, y_4 - y_3, z_4 - z_3)\end{aligned} \tag{2.5}$$

The cross product is written as:

$$\vec{A}\times\vec{B} = \begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ A_x & A_y & A_z\\ B_x & B_y & B_z\end{vmatrix} = \begin{pmatrix}(y_1 - y_3)(z_2 - z_3) - (z_1 - z_3)(y_2 - y_3)\\ (z_1 - z_3)(x_2 - x_3) - (x_1 - x_3)(z_2 - z_3)\\ (x_1 - x_3)(y_2 - y_3) - (y_1 - y_3)(x_2 - x_3)\end{pmatrix} \tag{2.6}$$

The volume of the tetrahedron is therefore written in coordinate form as:

$$V = \frac{1}{6}\begin{pmatrix}\left((y_1 - y_3)(z_2 - z_3) - (z_1 - z_3)(y_2 - y_3)\right)\cdot(x_4 - x_3) +\\ \left((z_1 - z_3)(x_2 - x_3) - (x_1 - x_3)(z_2 - z_3)\right)\cdot(y_4 - y_3) +\\ \left((x_1 - x_3)(y_2 - y_3) - (y_1 - y_3)(x_2 - x_3)\right)\cdot(z_4 - z_3)\end{pmatrix} \tag{2.7}$$

With the volume calculation of a single tetrahedron available, the volume overlap between any two tetrahedrons can be calculated using a systematic calculation procedure. An example of two overlapping tetrahedrons is shown in Figure 2.5. The figure also shows the common volume shared by the tetrahedrons. It is this common volume that must be calculated in order to perform volume weighted interpolation.

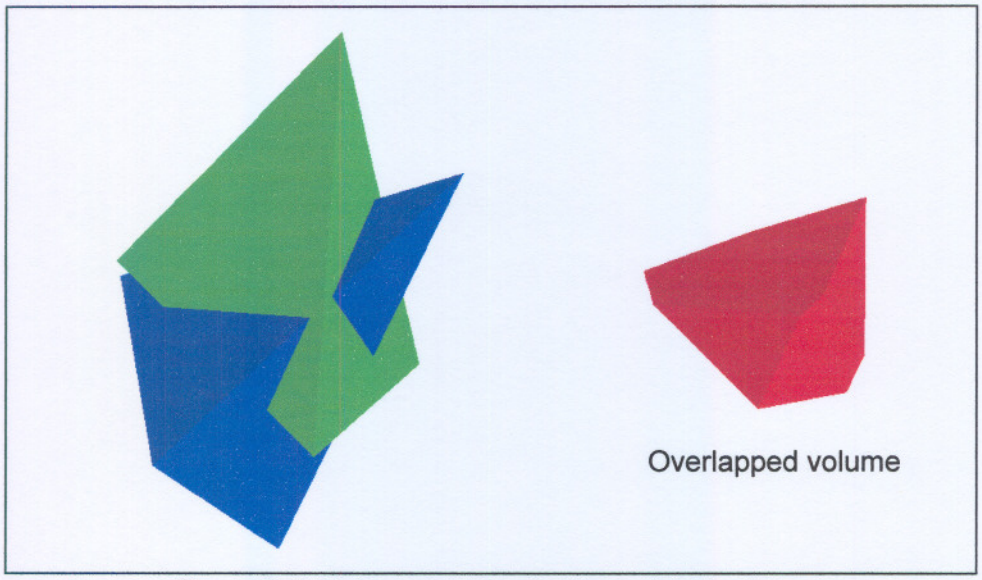


Figure 2.5: Volume overlap between tetrahedrons

The algorithm is based on a *clipping* principle starting off with two tetrahedrons A and B. The nodes of the tetrahedrons are numbered in such a way that when looking from node four towards the plane containing nodes one to three, the nodes are numbered in an anticlockwise direction.

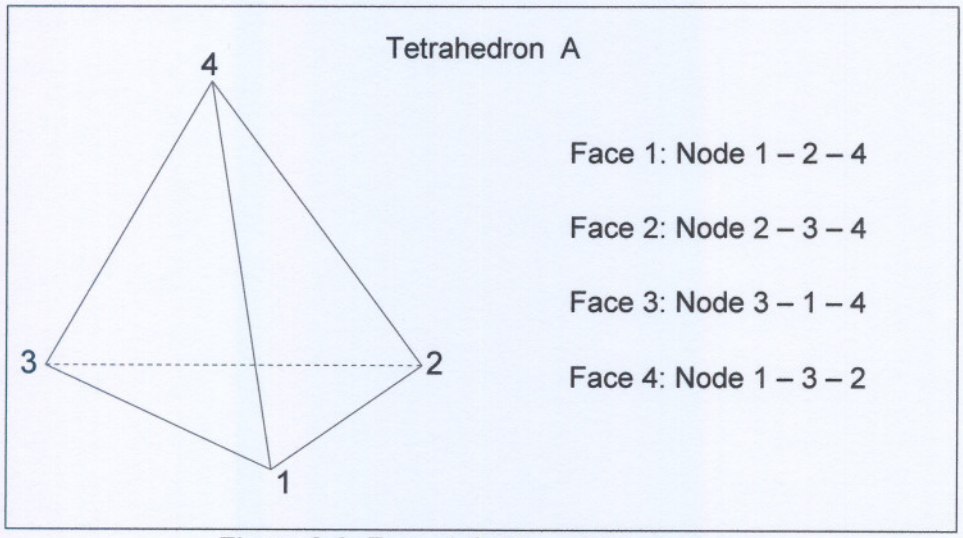


Figure 2.6: Face definition of tetrahedron

Tetrahedron A is considered fixed while tetrahedron B can be subdivided into new tetrahedrons in accordance with the requirements of the algorithm. The algorithm loops once through each of the four faces of tetrahedron A. The face numbers are defined in terms of node numbers as shown in Figure 2.6 and are numbered anticlockwise when viewing the face from outside the tetrahedron.

For each face of tetrahedron A the volume formed by the three nodes defining the face and the four nodes of tetrahedron B or a sub tetrahedron of tetrahedron B is calculated. The volumes are calculated by using the method described above, but by defining the two vectors describing the face in such a way that $\vec{A}\times\vec{B}$ points outwards in relation to tetrahedron A. An example is shown in Figure 2.7 for face number one of tetrahedron A. There are three possibilities for the location of each of the nodes of tetrahedron B in relation to the face of tetrahedron A, namely:

1. The node is located on the surface formed by the face, but not necessarily inside the perimeter formed by the face edges. In this case the calculated volume will be zero.
2. The node is located in front of the plane that is formed by the face of tetrahedron A. *In front* implies that the node is positioned on the side of the plane formed by the face in the direction in which the vector $\vec{A}\times\vec{B}$ points, Figure 2.7. In this case the calculated volume will be positive.
3. The node is located behind the plane described in 2, Figure 2.7. In this case the calculated volume will be negative.

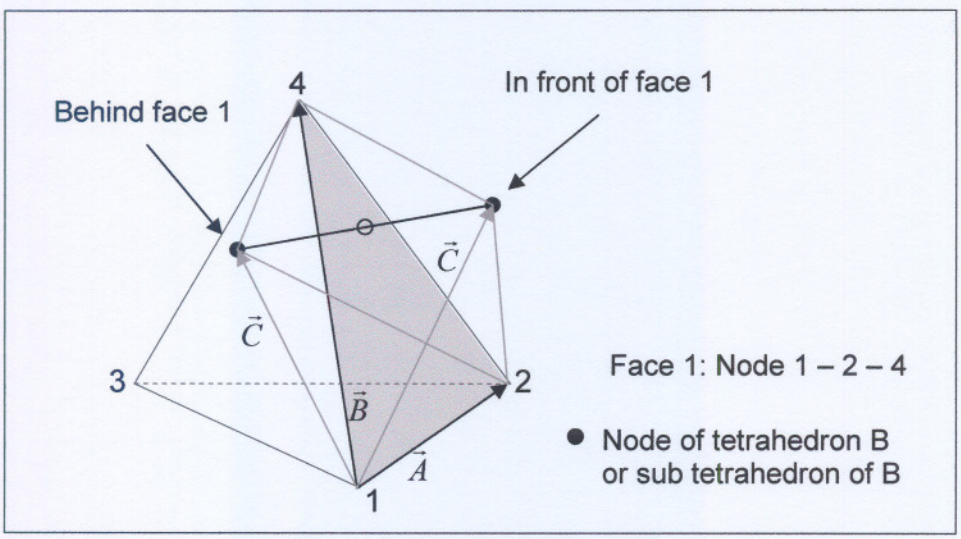


Figure 2.7: Volume calculation

For face one of tetrahedron A, the algorithm calculates the volume formed by the three nodes of the face and each of the four nodes of tetrahedron B, given that tetrahedron B is the only tetrahedron in the queue for processing at this point. A total of four volumes are calculated. The following five possibilities are based on the four calculated volumes:

Case 1: Number of positive and zero volumes equals four

This result indicates that tetrahedron B lies completely on the outside of tetrahedron A with zero overlapping. Some nodes of tetrahedron B may be located on the plane that is formed by the

face of tetrahedron A but this does not contribute to the overlapped volume. In this case the tetrahedron is deactivated and will not be processed further.

Case 2: Number of negative and zero volumes equals four

This result indicates that tetrahedron B lies completely behind the plane that is formed by the face. In this case tetrahedron B remains in queue for processing with the remaining faces of tetrahedron A and therefore remains active.

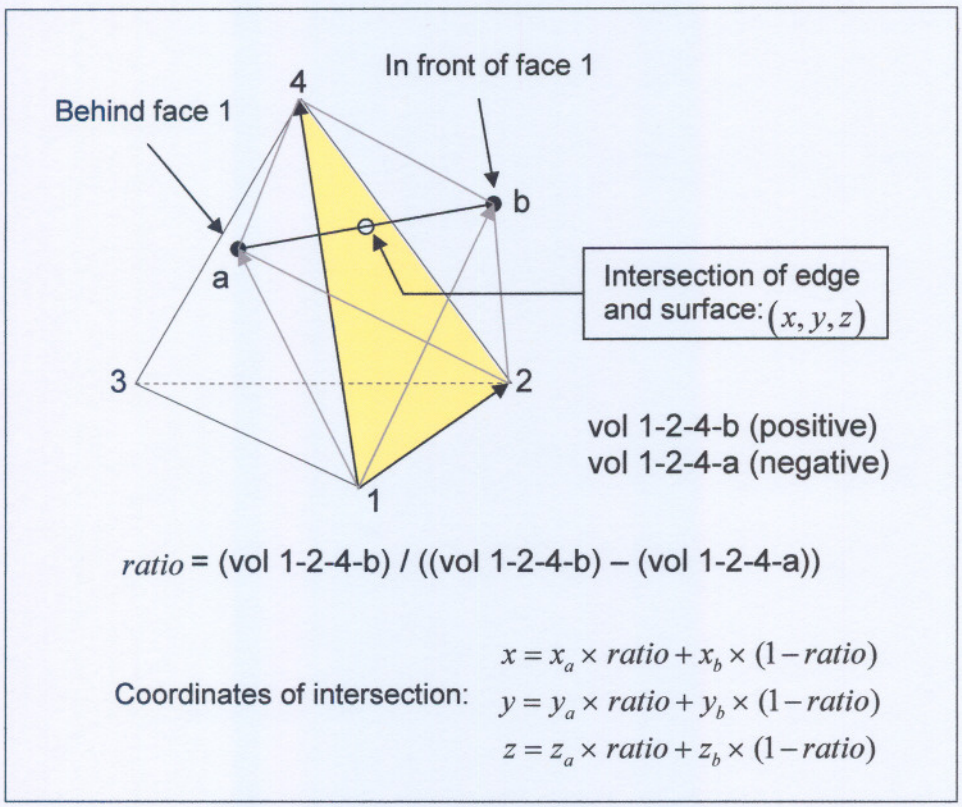


Figure 2.8: Intersection of line and surface

Case 3: Number of negative volumes equals one

This result indicates that one node of tetrahedron B lies behind the face. A new tetrahedron is formed by the face of tetrahedron A dividing tetrahedron B into two sections. The node coordinates of this new tetrahedron are computed and stored as a modification of the original tetrahedron B. Tetrahedron B is therefore clipped by the face, removing the section that falls in front of the face. The newly formed sub tetrahedron remains active and in queue for further processing. Figure 2.10 shows an example of the new tetrahedron that is formed by the clipping process. The intersection of the edge and surface is calculated using the ratios of the volumes that were already calculated. This procedure is illustrated in Figure 2.8. The volumes can be used to determine ratios since $V = \frac{1}{3}hA$ applies and the area is shared by both tetrahedrons. Even when the line connecting nodes a and b is not normal to the surface the normal

component given by the height h is directly related to the length of the line segment through a trigonometric relationship and therefore still applicable.

Case 4: Number of negative volumes equals two

This result indicates that two nodes of tetrahedron B are located behind the face. In this case the tetrahedron is deactivated and clipped by the face, while the remaining section behind the face is divided into three new tetrahedrons. The new tetrahedrons are activated and queued for further processing. Figure 2.9 shows how the clipped polyhedron is divided into three new tetrahedrons. In this example the section to the right of the plane is divided into three new tetrahedrons.

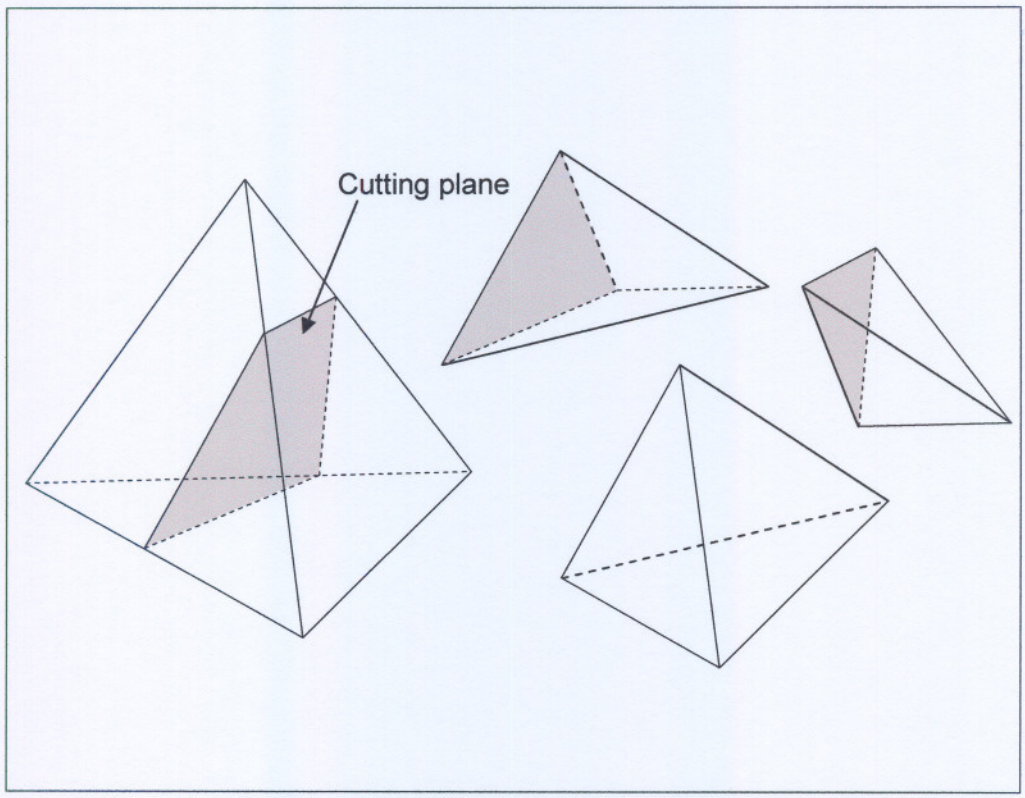


Figure 2.9: Splitting a tetrahedron into sub tetrahedrons – 2 points

Case 5: Number of negative volumes equals three

This result indicates that three nodes of tetrahedron B are located behind the face. In this case the tetrahedron is deactivated and clipped by the face, while the remaining section behind the face is divided into three new tetrahedrons. The new tetrahedrons are activated and queued for further processing. Figure 2.10 shows an example of how the remaining part is divided into three tetrahedrons which together, occupies the same space as the truncated tetrahedron.

This process is repeated for the remaining faces of tetrahedron A with the sub tetrahedrons of tetrahedron B being processed and in turn being subdivided into smaller tetrahedrons. When

the process is completed an array of sub tetrahedrons of tetrahedron B remains which are all located within the geometrical space of tetrahedron A. The volumes of these active sub tetrahedrons are calculated using Eq. (2.4) and added together to provide the overall volume overlapped between tetrahedron A and B. The function *tetmap* (du Toit, 2005) accepts the node coordinates of tetrahedron A and B (in the correct order according to the numbering convention) and returns the common volume between the tetrahedrons.

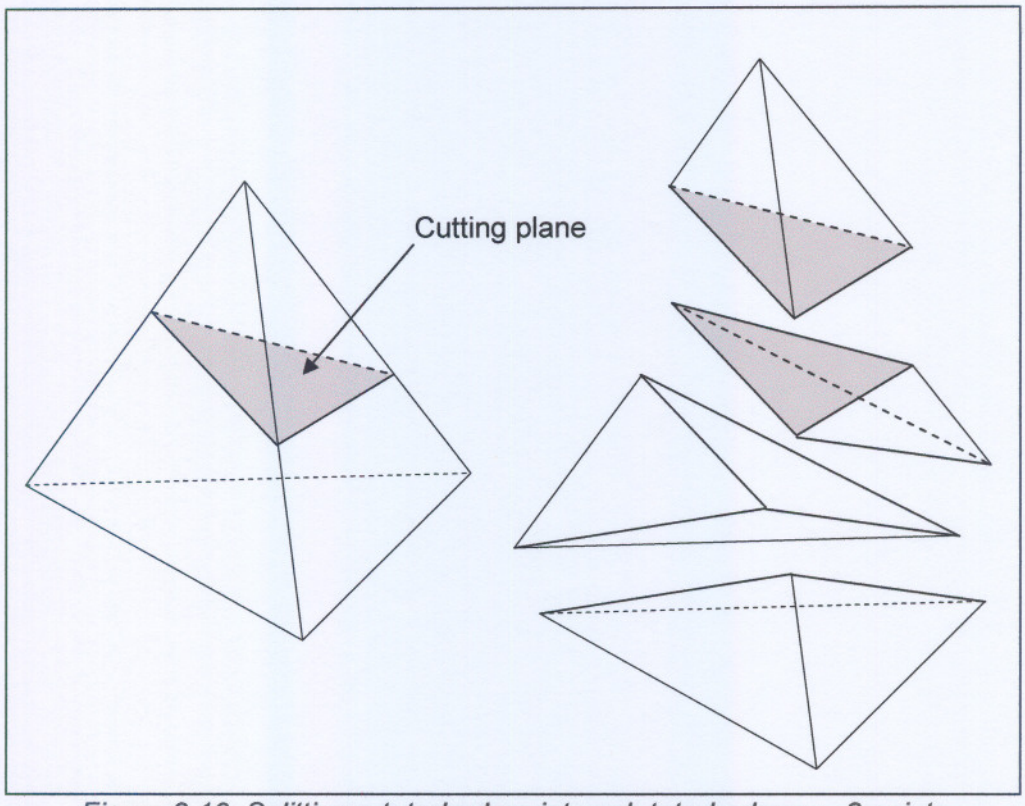


Figure 2.10: Splitting a tetrahedron into sub tetrahedrons – 3 points

Any polyhedral cell can be divided into a set of non-overlapping tetrahedrons so that the tetrahedrons occupy the same space as the polyhedral cell. Since meshes consisting of hexahedral cells are used in this study, the routine *hexmap* (du Toit, 2005) is used to compute the common volume between any two hexahedral cells A and B. This routine uses *tetmap* to compute the overlapped volume between tetrahedrons.

Each hexahedral cell is divided into six non-overlapping tetrahedral cells. The overlapped volume of each tetrahedron in hexahedron A is calculated with each tetrahedron in hexahedron B and added together to obtain the total overlapped volume between the two hexahedral cells. An example of two overlapping hexahedrons is shown in Figure 2.11. To ensure that the overlapping volume is calculated correctly, it is essential that the hexahedron is surrounded by planar faces, i.e. the four nodes defining a face must lie on the same plane. Warped faces will result in inaccurate volume calculations.

The procedure described in this section can be extended to calculate the common volume between any two multi-faced polyhedral cells. The only modification required is to divide the polyhedral cell into tetrahedrons, a process that can readily be automated. There are three possibilities for volume overlapping of any two cells located in three-dimensional space.

Zero overlapping

Zero overlapping simply implies that the two cells are far enough apart so that there is no overlapping between them, in other words, there is no common volume.

Partial overlapping

Partial overlapping occurs when the cells are in close proximity to one another so that a common volume is shared by both the cells. The overlapping algorithm described in this section is used to calculate the common volume.

Complete overlapping

Complete overlapping can imply one of two possible conditions. Firstly, it can imply that the two cells have exactly the same geometry and orientation so that the overlapping volume equals the volumes of each of the two cells. Secondly, it can imply that one of the cells is completely enclosed within the volume of the larger cell with no vertices or edges protruding the surface of the larger cell. In this case the overlapping volume equals the volume of the smaller cell.

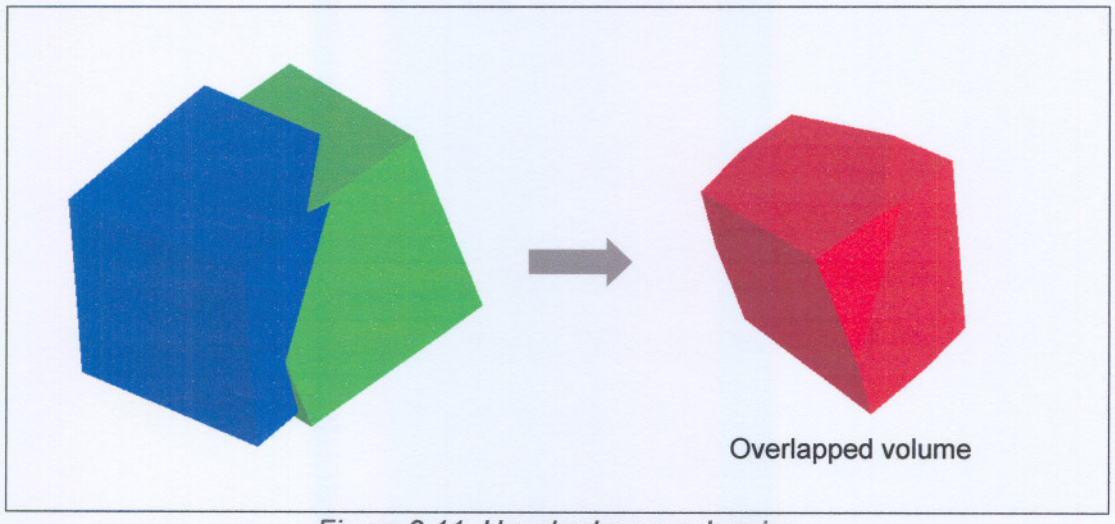


Figure 2.11: Hexahedron overlapping

2.4 Data structures

To perform volume weighted interpolation in a finite volume CFD code, the volume overlapping data must be easily accessible to the internal data structures of the code. In this work the route of object-oriented programming (OOP) and dynamic memory allocation using the programming language *C++* was followed to achieve effective data storage and access (Stroustrup, 1997).

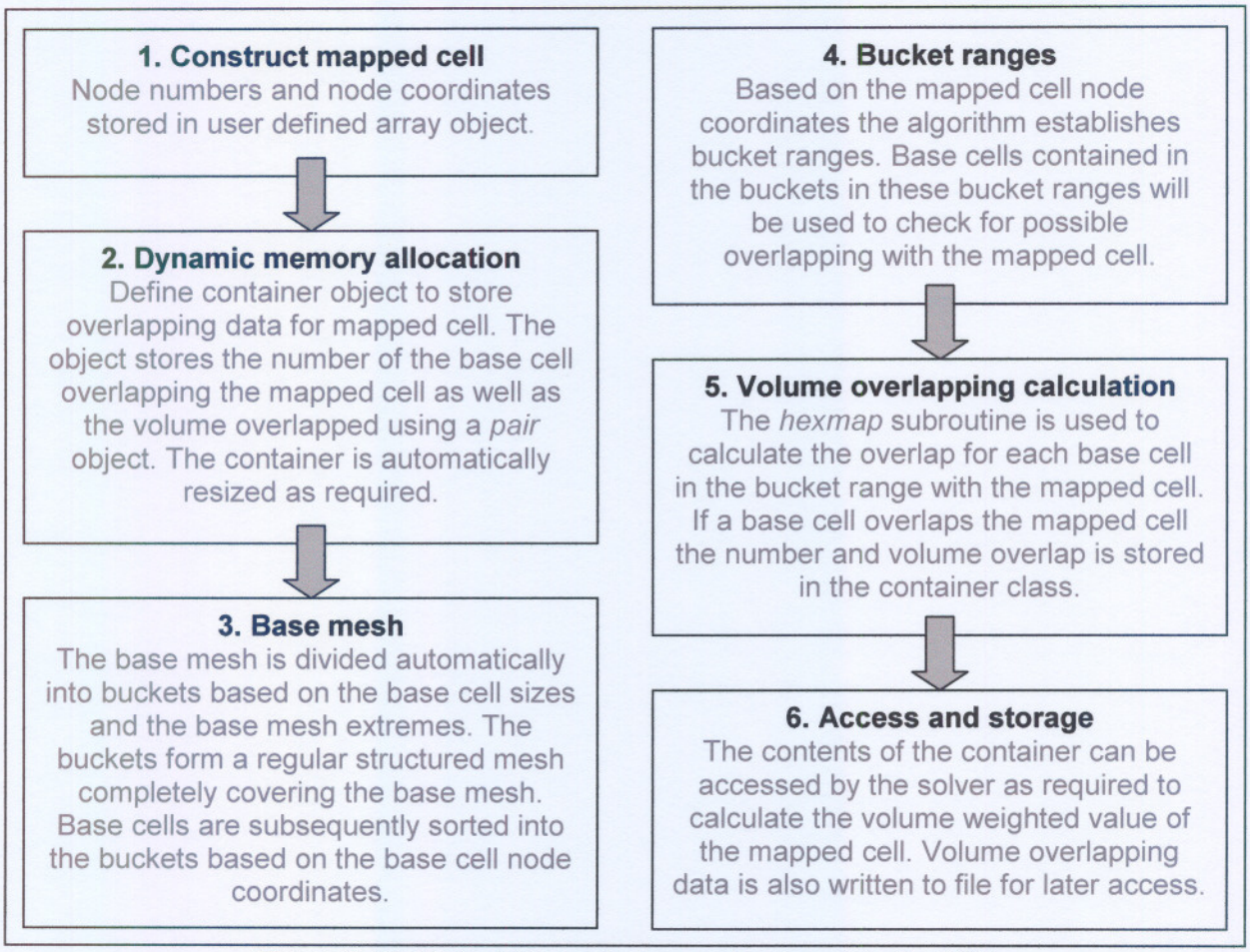


Figure 2.12: Calculating volume overlapping data for VWI

Classes were created for user defined types to facilitate the storage of all the required information in dynamic memory structures. The volume overlapping data is calculated before the start of the simulation and stored for access by the solver when required. The data can also be written to file for later access to prevent re-calculation when a simulation is restarted. The use of dynamic memory is an absolute requirement in this work as it is not known before a simulation how many base cells will be overlapped by a mapped cell. The specific data structures used for volume weighted interpolation applied to convective transport is described in more detail in Chapter 3.

A *bucket search algorithm* is used to improve the efficiency of volume overlapping calculations. The base mesh is automatically divided into a number of buckets containing base cells. For any mapped cell, the buckets that should be searched for overlapping cells, is identified by the algorithm and only the cells within those buckets are processed. This prevents the algorithm from using all the cells in the base mesh in order to calculate the overlapped volume for a single mapped cell. Figure 2.12 summarizes the process followed to calculate the volume overlapping data of a single mapped cell overlapping a base mesh. In cases where more than one mapped cell is present, the process is extended to accommodate all the mapped cells. This is achieved by using multidimensional dynamic array elements from the C++ standard template library.

2.5 Finite volume toolkit

In recent years the object oriented approach to modelling transport processes with the finite volume method has become popular since the finite volume method lends itself effectively to an object oriented implementation, Weller *et al.* (1998) and Gooch (2002). The Finite volume toolkit (FTK) is an object oriented finite volume platform for unstructured computational meshes, employing the collocated variable arrangement, finite volume discretisation and segregated pressure-based solution algorithms for pressure velocity coupling.

	Operator
Transient term:	*DDT()*
Convection term:	*DIV()*
Diffusion term:	*LAPLACE()*
Source term:	*SC()*

Scalar transport equation definition in FTK

```
CFiniteVolumeEquation variableEquation
(
      FVM::DDT(density, variable, timestep, temporal scheme)
    + FVM::DIV(massflux, variable, convection scheme, blending factor)
    + FVM::LAPLACE(viscosity, variable)
    + FVM::SC(volume source for variable)
);

variableEquation.Solve();
variable.UpdateBoundaries();
```

Figure 2.13: Extract from FTK source code

FTK was developed to fulfil in the basic requirements of a general purpose simulation platform based on the finite volume method (Kruger, 2005). Object oriented programming techniques were applied to create the required data types, while operator overloading allows for normal mathematical symbols to be used for the basic mathematical operations applicable to user defined types. It is generally recognised that object oriented programming enables the creation of a code that is easier to validate and maintain than procedural based codes (Weller *et al.*, 1998).

In FTK a computational mesh consists of cell volumes, inner faces and boundary faces. Classes were developed for each of these mesh elements. Boundary faces are located on the outside of the computational mesh while inner faces are located inside the mesh with cells on either side of the face. FTK contains classes that allow all the aspects of the numerical modelling process to be controlled. This includes the specification of boundary conditions, solution parameters, solution algorithms, iterative matrix solvers, etc. Dynamic scalar and vector storage arrays are used to store scalar and vector field variables. FTK provides a range of implicit and explicit differential field operators for the solution of partial differential equations.

The general transport equation for a scalar dependent variable described in Chapter 1, consists of a transient, convection, diffusion and source term. Figure 2.13 shows an extract from the FTK source code. Transport equations are constructed using the four operators as shown. Each operator accepts parameters that indicate the scalar variable field to be solved, the temporal integration scheme to be used, the convection scheme, etc. Any term that does not fit into the standard forms of the convection and diffusion terms, forms part of the source term, for example boundary fluxes and non-orthogonal contributions to the diffusion term.

FTK uses internal lists to establish the connectivity between cells, faces and boundaries for arbitrary unstructured meshes, and provides internal data structures for variable storage and manipulation. Vector variables such as velocity, are handled on a component basis and pressure velocity coupling established using segregated solution algorithms such as SIMPLE (Patankar, 1980) and PISO (Issa,1986). FTK utilises the collocated variable arrangement, where all dependent variables are solved for and stored at cell centres. To prevent pressure-velocity decoupling, the interpolation method developed by Rhie and Chow (1983) is applied to interpolate velocities to control volume faces when solving the Navier-Stokes equations for fluid flow.

2.6 Closure

In this chapter the concept of volume weighted interpolation was introduced. Three examples demonstrated how conservative interpolation between meshes is performed using the method of volume weighted interpolation. The conservative interpolation of variables between meshes is an essential requirement in the finite volume method. When interpolation is not performed conservatively, the inherent conservation property of the finite volume method is violated.

The algorithm used for volume overlapping calculations was described in this chapter. It is based on the calculation of the common volume between any two tetrahedrons and extended for the calculation of the common volume between hexahedral cells. The dynamic data structures used to store and access volume overlapping data was briefly described. FTK, the object oriented research code that was used for all numerical work in this thesis, was also introduced. The use of volume weighted interpolation for modelling convective transport on arbitrary unstructured meshes is considered in Chapter 3.

3. Volume weighted interpolation for convective transport

3.1 Introduction

Chapter 2 introduced the concept of volume weighted interpolation (VWI). In this chapter this technique is applied to the modelling of convective transport within the framework of the finite volume method. The chapter starts off with the finite volume discretisation of the general transport equation for a scalar dependent variable. The need for face value interpolation follows directly from this discretisation. The chapter provides background information on convection schemes that are used to calculate face values. This includes high-resolution convection schemes that are based on the normalised variable diagram. The results from several test cases show how well these schemes perform when applied to orthogonal meshes. An orthogonal projection interpolation stencil based on volume weighted interpolation is then introduced. This stencil allows high-resolution schemes to be implemented on arbitrary unstructured meshes. A comparison between the results obtained for orthogonal and arbitrary unstructured meshes for similar test cases, is presented to test the proposed interpolation stencil.

3.2 Finite volume discretisation

3.2.1 The general transport equation

Finite volume discretisation involves the transformation of transport equations into algebraic equations. This process consists of two parts, namely spatial discretisation and equation discretisation (Hirsch, 1988). Spatial discretisation involves the creation of a mesh across the flow domain while equation discretisation refers to the transformation of a partial differential equation into an algebraic equation that is compatible with the computational mesh and consistent with the partial differential equation.

Eq. (3.1) is the partial differential form of the general transport equation for a scalar variable ϕ.

$$\frac{\partial}{\partial t}(\rho\phi)+\vec{\nabla}\cdot(\rho\vec{u}\phi)-\vec{\nabla}\cdot(\Gamma\vec{\nabla}\phi)=S_{\phi} \tag{3.1}$$

In Eq. (3.1) ρ is the fluid density, $\vec{u}$ the velocity vector, Γ is the diffusion coefficient and S_{ϕ} the source term. $\vec{\nabla}\cdot$ represents the divergence operator.

To discretise Eq. (3.1), each term of the equation is integrated over a control volume which forms part of the computational mesh. This spatial integration of Eq. (3.1) over an arbitrary control volume, results in the integral form of the general transport equation given by Eq. (3.2).

$$\int_V \left(\frac{\partial}{\partial t}(\rho\phi) \right) dV + \int_V \vec{\nabla} \cdot (\rho \vec{u} \phi) dV - \int_V \vec{\nabla} \cdot \left(\Gamma \vec{\nabla} \phi \right) dV = \int_V S_\phi dV \tag{3.2}$$

For transient simulations, each term in Eq. (3.1) must also be integrated over a time interval δt.

Convective and diffusive fluxes are calculated across control volume faces in the finite volume method. These fluxes are formulated as surface integrals. The divergence theorem of Gauss is applied to the convection and diffusion terms in Eq. (3.2) to transform the volume integrals into surface integrals (Anton, 1995). For a vector function $\vec{F}$ over a domain V with an outward oriented surface, ∂V, the volume integral of the divergence of $\vec{F}$ can be written as a surface integral,

$$\int_V \left(\vec{\nabla} \cdot \vec{F} \right) dV = \int_{\partial V} \vec{F} \cdot d\vec{A} \tag{3.3}$$

provided that the vector function $\vec{F}$ has continuous first partial derivatives across the domain. In Eq. (3.3) $d\vec{A}$ is an outward oriented differential surface area vector and ∂V the *closed surface* surrounding the volume V. With the application of Eq. (3.3) to the convection and diffusion terms in Eq. (3.2), the following integral form of the transport equation is obtained:

$$\int_V \left(\frac{\partial}{\partial t}(\rho\phi) \right) dV + \int_{\partial V} (\rho \vec{u} \phi) \cdot d\vec{A} - \int_{\partial V} \left(\Gamma \vec{\nabla} \phi \right) \cdot d\vec{A} = \int_V S_\phi dV \tag{3.4}$$

The surface integrals in Eq. (3.4) represent the net efflux through the control surface due to convection and diffusion respectively. For the purpose of this work diffusive flux is eliminated from the transport equation and only convective transport considered. Any diffusion that may be present within a solution therefore originated from numerical diffusion and not physical diffusion. Eq. (3.4) is transformed into a convective transport equation by eliminating the diffusion term from the equation.

$$\int_V \left(\frac{\partial}{\partial t}(\rho\phi) \right) dV + \int_{\partial V} (\rho \vec{u} \phi) \cdot d\vec{A} = \int_V S_\phi dV \tag{3.5}$$

Eq. (3.5) is discretised on a term by term basis in the following sections, beginning with the spatial integration of each term.

3.2.2 The time derivative term

The time derivative term in Eq. (3.5) is a volume integral. In the finite volume method volume integrals are evaluated by multiplying the integrand with the cell volume. Since the integrand is evaluated at the cell centre, volume integrals are easily calculated and no interpolation is required when using the collocated variable arrangement.

$$\int_V \left(\frac{\partial}{\partial t}(\rho\phi) \right) dV = \left(\frac{\partial}{\partial t}(\rho\phi) \right)_P V_P \tag{3.6}$$

The volume integral is approximated by the product of the integrand as defined at the cell centre and the volume of the cell, V_P.

3.2.3 The convection term

The convection term in Eq. (3.5) is a surface integral. In the finite volume method surface integrals are evaluated by calculating the scalar product of the integrand at the face centre and the face area vector. Since the face value of the integrand is not available it must be determined in terms of the cell values surrounding the face through interpolation. There are two levels of approximation present when surface integrals are evaluated. Firstly the face centre value is used to approximate the face value and secondly the approximation of the face value in terms of cell values through interpolation. The accurate calculation of surface integrals is one of the most important aspects in the finite volume method. During discretisation the surface integral for the control surface is split into separate surface integrals for the faces surrounding a cell.

$$\begin{aligned}
\int_V \vec{\nabla} \cdot (\rho\vec{u}\phi)\, dV &= \int_{\partial V} (\rho\vec{u}\phi) \cdot d\vec{A} \\
&= \sum_{f=1}^{n} \left(\int_f (\rho\vec{u}\phi) \cdot d\vec{A} \right) \\
&= \sum_{f=1}^{n} \left((\rho\vec{u}\phi)_f \cdot \vec{A}_f \right) \\
&= \sum_{f=1}^{n} \left(\left((\rho\vec{u})_f \cdot \vec{A}_f \right) \phi_f \right) \\
&= \sum_{f=1}^{n} \left(G_f \phi_f \right)
\end{aligned} \tag{3.7}$$

In Eq. (3.7) G_f is the mass flow through the cell face. The mass flow calculation requires the interpolated values of the velocity vector and the fluid density at the face. The face value of the dependent variable ϕ_f must be interpolated using a suitable convection differencing scheme. $G_f\phi_f$ represents the convective flux of the flow quantity across face f.

The convection term has the physical characteristic of boundedness that dictates that the bounds of ϕ given by its initial distribution may not be violated. If ϕ represents a scalar quantity that varies between zero and one, for example, a solution with values greater than one and smaller than zero violates the boundedness characteristic of the convection term. The discretised form of the convection term must preserve the boundedness characteristic, (Jasak and Weller, 1995).

3.2.4 The source term

The source term in Eq. (3.5) is also a volume integral and therefore discretised in the same way as the time derivative term. Source terms include all terms that cannot be expressed in the standard form of the convection and diffusion terms. It includes sources or sinks of the flow quantity within a cell as well as sources originating from fluxes across boundary faces.

The general approach with regard to source terms is to linearize the term into a constant part S_b and a part that is a function of the value of the dependent variable, $S_P\phi_P$.

$$S_\phi = S_b + S_P\phi_P \tag{3.8}$$

$S_P\phi_P$ can be treated implicitly or explicitly or any combination thereof, but it is essential that the coefficient S_P is negative for an implicit calculation to preserve the diagonal dominance of the coefficient matrix. Volume integration of Eq. (3.8) produces the following form of the source term.

$$\begin{aligned}\int_V S_\phi dV &= \int_V \left(S_b + S_P\phi_P\right) dV \\ &= S_b V_P + S_P\phi_P V_P\end{aligned} \tag{3.9}$$

Patankar (1980) presents various linearisation methods for common source terms. In general an explicit treatment of the source term will slow convergence since the source term is evaluated using a lagged value of the dependent variable.

3.3 Temporal integration

For transient simulations each term in Eq. (3.5) is integrated over a time increment δt.

$$\int_t^{t+\delta t}\left[\left(\frac{\partial}{\partial t}(\rho\phi)\right)_P V_P + \sum_{f=1}^{n}\left(G_f\phi_f\right)\right]dt = \int_t^{t+\delta t}\left(S_b V_P + S_P\phi_P V_P\right)dt \tag{3.10}$$

The temporal integration of the time derivative term follows directly.

$$\begin{aligned}\int_t^{t+\delta t}\left(\frac{\partial}{\partial t}(\rho\phi)\right)_P V_P dt &= V_P\int_t^{t+\delta t}\left(\frac{\partial}{\partial t}(\rho\phi)\right)_P dt \\ &= \left((\rho\phi)_P^{t+\delta t} - (\rho\phi)_P^{t}\right)V_P\end{aligned} \tag{3.11}$$

In a stationary mesh, the cell geometry remains constant throughout the simulation. The volume can therefore be moved outside of the time integral term.

The variation of the remaining integrands with time is not known. The integration is therefore performed in terms of a weighting factor η weighing the integration between the new and old time level values of the integrand. The general formulation for the time integration of an integrand f is written as:

$$\begin{gathered}\int_t^{t+\delta t} f\,dt = \left[(1-\eta)f^{t} + (\eta)f^{t+\delta t}\right]\delta t \\ 0 \le \eta \le 1\end{gathered} \tag{3.12}$$

Three choices for the weighting factor η in Eq. (3.12) are explicit ($\eta = 0$), implicit ($\eta = 1$) and Crank Nicolson ($\eta = 0.5$). In this work the Crank Nicolson scheme is used for all transient simulations. The solution of transient problems is obtained by a time marching process, starting off with the initial values of the field variables at time t and solving the equations for the prescribed time interval with either a fixed time step δt, or a variable time step. This process is repeated until the required time interval for the simulation has elapsed.

3.4 Convection differencing schemes

The logical choice for ϕ_f in (3.7) would be the value obtained by linear interpolation from the cell values bracketing the face. This differencing scheme is known as central differencing (CD). Although central differencing seems to be the obvious method to calculate ϕ_f, this scheme is

only successful for modelling diffusion dominated flows and leads to unbounded results for flows where convection dominates.

For flows where the Peclet number > 2, the central differencing scheme can cause unphysical oscillations in the solution due to negative coefficients in the coefficient matrix of the discretised transport equation (Versteeg and Malalasekera, 1995). To overcome the negative coefficient problem caused by the central differencing scheme, Courant, Isaacson and Rees (1952) developed the upwind differencing (UD) scheme. As the name implies, the upwind differencing scheme approximates the value of ϕ_f by the value of the *donor* cell of face f. The upwind cell is determined by the flow direction across the face.

$$\phi_f = \begin{cases} \phi_P & \text{if} \quad G_f \geq 0 \\ \phi_N & \text{if} \quad G_f < 0 \end{cases} \tag{3.13}$$

While eliminating unphysical oscillations from the solution by guaranteeing positive coefficients in the coefficient matrix of the discretised equation, the upwind differencing scheme can introduce unacceptable levels of numerical diffusion into the solution.

The third order QUICK scheme (Leonard, 1979), approximates ϕ_f by fitting a parabolic profile through the donor, acceptor and upwind cells. Although accurate, the QUICK scheme does not guarantee bounded solutions. On an orthogonal mesh with equally spaced control volumes, the face value is calculated as:

$$\phi_f = -\frac{1}{8}\phi_U + \frac{3}{4}\phi_D + \frac{3}{8}\phi_A \tag{3.14}$$

In Eq. (3.14) ϕ_U represents the value of the upwind cell to the donor cell of face f, ϕ_D the donor cell value and ϕ_A the acceptor cell value. The selection of the upwind, donor and acceptor cells are determined by the flow direction through the face. The cells form a three-point stencil around the face f.

Figure 3.1 shows how the face value ϕ_f is calculated using the UD, CD and QUICK schemes respectively. The UD and QUICK schemes require the flow direction to calculate the face value, therefore including transportiveness into the formulation, (Versteeg and Malalasekera, 1995). In addition the QUICK scheme requires a three-point stencil to calculate the face value.

Three-point stencils are commonly used in convection schemes but cannot be applied directly with face addressed algorithms where only the values of the cells bracketing a face are available for interpolation purposes.

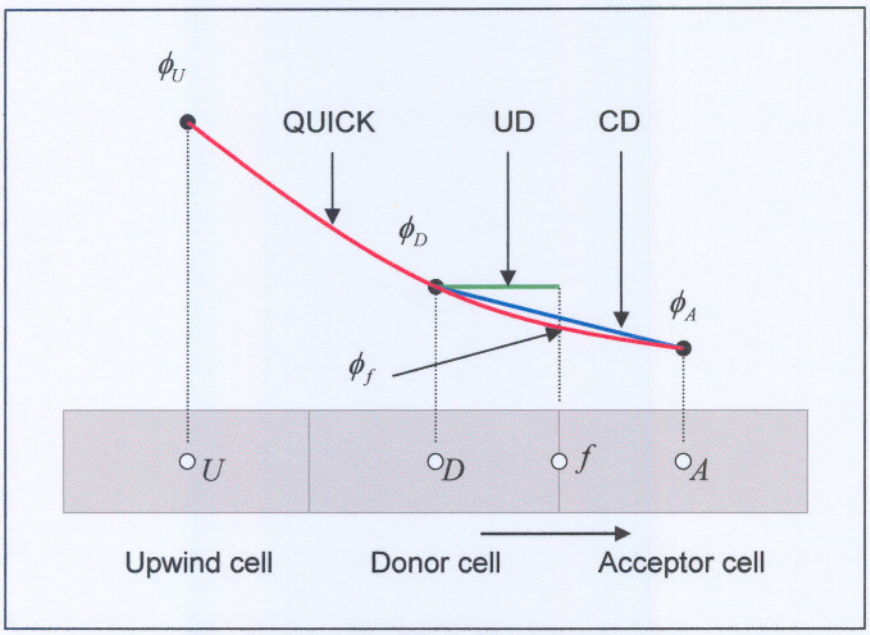


Figure 3.1: UD, CD & QUICK schemes

The following example demonstrates how the UD, CD and QUICK schemes perform when modelling the transient convection of a complex block wave profile.

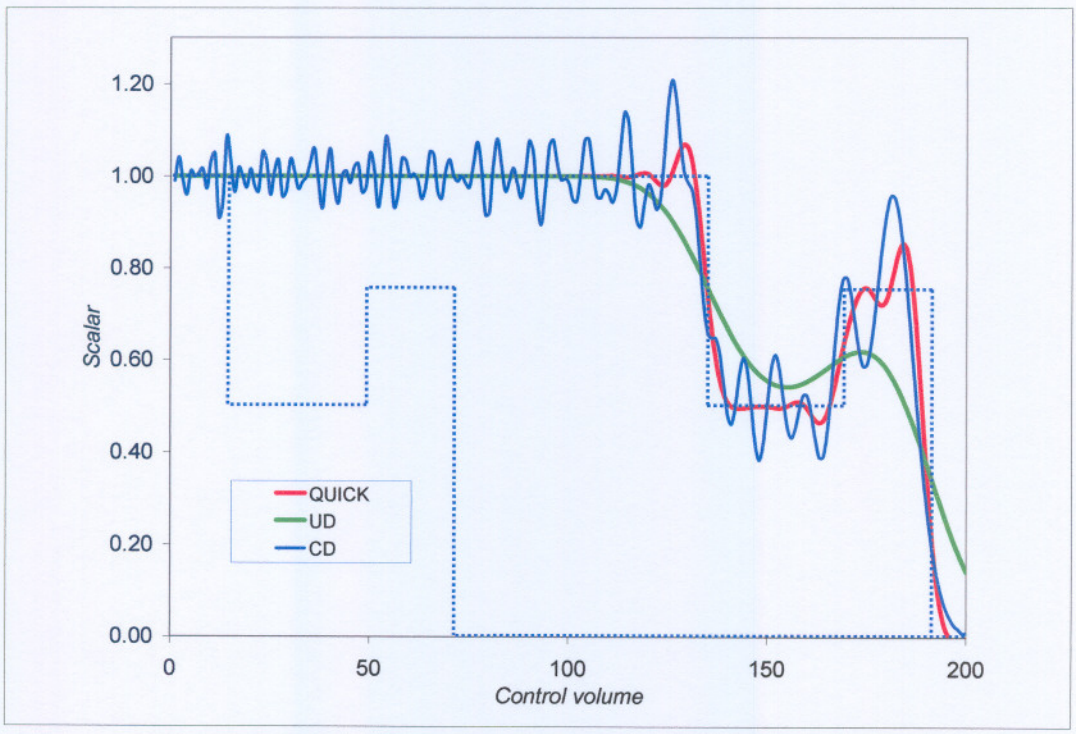


Figure 3.2: One-dimensional convection: UD, CD & QUICK

An equally spaced mesh is used with a uniform velocity of 1 m/s to the right. Convection without any diffusion is modelled. The results obtained with the UD, CD and QUICK schemes are shown together with the analytical solution in Figure 3.2. The Courant number for this example is 0.001. The simulation was terminated after three seconds corresponding to 120000 time steps of 2.5e-5 seconds each.

The central differencing scheme produces a solution that is unbounded with peaks and troughs around the analytical solution. One application where boundedness is essential is the volume of fluid (VOF) method where volume fractions are used to identify different fluids or fluid phases, (Ubbink, 1997). The upwind differencing scheme produces a solution that is bounded but severely diffusive. Note how the 0.75 peak of the block wave is smoothed out by the upwind differencing scheme. The QUICK scheme, although also unbounded in regions, produces a substantial improvement over the solution obtained with the central differencing scheme. This example clearly shows the significant influence of different face value interpolation schemes on a solution.

In the presence of steep gradients, the central differencing scheme fails to produce a bounded solution; the reason being that the face value calculated is either too low or too high causing the solution to overshoot or undershoot. A steep, positive gradient will predict a too high a face value which could result in the donor cell donating more than it has to offer. A steep negative gradient will cause the donor cell to donate more than the acceptor cell is able to accept, thereby causing positive unboundedness. The unboundedness caused by a differencing scheme is also referred to as numerical dispersion. To improve the accuracy of the upwind differencing scheme, several other convection schemes were developed over the years. Among these are the upwind biased low order schemes, for example the hybrid and power-law schemes, that produce only marginally improved results compared to upwind differencing (Leonard and Mokhtari, 1990).

The calculation of ϕ_f for the convection term remains a contentious issue and an area of active research and development in the finite volume method. To improve the boundedness and accuracy characteristics of convection schemes, blended differencing (BD) schemes were developed which blends lower and higher-order differencing schemes to suppress numerical dispersion while limiting numerical diffusion (Ferziger and Peric, 1999). The amount of blending required to produce bounded solutions is unfortunately not known before a simulation, resulting in a trail and error approach towards selecting appropriate blending factors. The ideal convection scheme should produce bounded solutions while limiting numerical diffusion. These requirements have led to the development of high-resolution (HR) schemes (Jasak *et al.*, 1999). High-resolution schemes are composite higher-order schemes, which produce bounded

solutions by including flux-limiters into the formulation. The following section describes high-resolution convection schemes.

3.5 High-resolution convection schemes

3.5.1 The normalised variable diagram

The concept behind high-resolution schemes is introduced by referring to the normalised variable diagram (NVD) (Leonard, 1991). Schemes based on the normalised variable diagram are implemented using three-point stencils similar to the QUICK scheme. The derivation of a differencing scheme based on the normalised variable diagram is performed for one-dimensional flow calculations and implemented on a face by face basis for simulations of multi-dimensional flows. In the finite volume method the scheme definition is the interpolation rule for the face value which can be expressed in terms of a three-point stencil for most schemes, (Tao *et al.*, 2004).

$$\phi_f = f(\phi_A, \phi_D, \phi_U) \tag{3.15}$$

When using a three-point stencil, a face value is determined by using at most two upstream nodes and a single downstream node. The face value can therefore be interpolated to at most third order accuracy, (Leonard, 1991). The term *monotonic* describes boundedness with regard to numerical methods. Three-point schemes are monotonic increasing or decreasing when the following conditions apply:

$$\begin{aligned} &\text{monotonic increasing:} \quad \phi_A < \phi_D < \phi_U \\ &\text{monotonic decreasing:} \quad \phi_U > \phi_D > \phi_A \end{aligned} \tag{3.16}$$

When (3.16) is satisfied for the entire domain, the solution will be free of unphysical oscillations. Leonard and Mokhtari (1990) defined two conditions that must be satisfied to ensure bounded solutions. The first condition requires the calculated face value to be bounded between the values of the cells bracketing the face. This condition includes the situation where $\phi_D = \phi_A$. The second condition requires that if $\phi_D = \phi_U$, the face value must be set equal to this value. To simplify the analysis of bounded convection schemes, Leonard (1991) defined the normalised variable as

$$\tilde{\phi} = \frac{\phi - \phi_U}{\phi_A - \phi_U} \tag{3.17}$$

Eq. (3.17) can be used to obtain expressions for the normalised donor cell and face values.

$$\tilde{\phi}_D = \frac{\phi_D - \phi_U}{\phi_A - \phi_U} \quad \tilde{\phi}_f = \frac{\phi_f - \phi_U}{\phi_A - \phi_U} \tag{3.18}$$

The normalised upwind and acceptor cell values reduces to zero and one respectively.

$$\tilde{\phi}_U = \frac{\phi_U - \phi_U}{\phi_A - \phi_U} = 0 \quad \tilde{\phi}_A = \frac{\phi_A - \phi_U}{\phi_A - \phi_U} = 1 \tag{3.19}$$

The solution in the stencil is monotone whenever the normalised donor cell value lies between zero and one. Higher-order approximations of the face value may then be performed.

The purpose of the normalised variable diagram is to express the normalised face value as a functional relationship of the normalised donor cell value, in other words the normalised variable diagram plots the normalised face value as a function of the normalised donor cell value. Gaskell and Lau (1988) defined the convection boundedness criterion (CBC) for implicit calculations. This criterion indicates in which regions of the normalised variable diagram a convection differencing scheme will produce bounded solutions. For implicit calculations the convection boundedness criterion is defined by the following parameters:

$$\begin{aligned} &\tilde{\phi}_f = \tilde{\phi}_D && \text{for} \quad \tilde{\phi}_D < 0 \quad \text{or} \quad \tilde{\phi}_D > 1 \\ &\tilde{\phi}_D < \tilde{\phi}_f < 1 && \text{for} \quad 0 < \tilde{\phi}_D < 1 \end{aligned} \tag{3.20}$$

For explicit flow calculations the convection boundedness criterion is adjusted to include the Courant number, effectively reducing the area within the normalised variable diagram where bounded solutions will be obtained as the Courant number increases from zero to one, Leonard (1991). The normalised face value for explicit calculations is calculated as a function of the normalised donor cell value as well as the face Courant number.

$$\tilde{\phi}_f = \left(1 - c_f\right)\tilde{\phi}_f^* + c_f \tilde{\phi}_D \tag{3.21}$$

In Eq. (3.21) $\tilde{\phi}_f^*$ is the normalised face value for implicit flow calculations and c_f the Courant number for face f, defined as

$$c_f = \frac{\left|\vec{u}_f \delta t\right|}{\delta x} \tag{3.22}$$

The CBC for explicit flow calculations is defined as

$$\begin{aligned} \tilde{\phi}_f &= \tilde{\phi}_D && \text{for} \quad \tilde{\phi}_D < 0 \quad \text{or} \quad \tilde{\phi}_D > 1 \\ \tilde{\phi}_D &< \tilde{\phi}_f < \min\left(1, \frac{\tilde{\phi}_D}{c_f}\right) && \text{for} \quad 0 < \tilde{\phi}_D < 1 \end{aligned} \tag{3.23}$$

The convection boundedness criterion for implicit and explicit flow calculations is shown graphically as the shaded regions in the normalised variable diagram, Figure 3.3. The convection boundedness criterion includes the line representing the upwind differencing scheme, where $\tilde{\phi}_f = \tilde{\phi}_D$. The upwind differencing scheme is the only scheme that is unconditionally bounded.

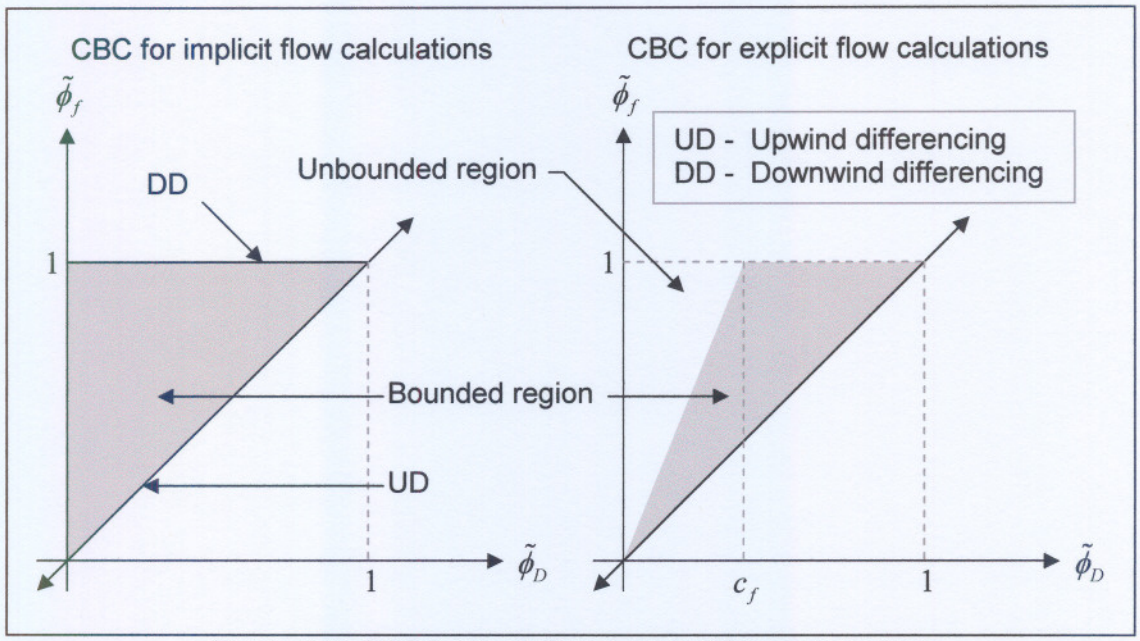


Figure 3.3: CBC for implicit and explicit flow calculations

For the definition of a bounded differencing scheme the normalised face value is expressed as a non-linear function of the normalised donor cell value, in such a way that the functional relationship is contained within the CBC region of the normalised variable diagram, Eq. (3.24).

$$\tilde{\phi}_f = f\left(\tilde{\phi}_D\right) \tag{3.24}$$

Once the normalised donor cell value is available, the normalised face value can be calculated from the functional relationship (3.24), which is non-linear by necessity, except for the first order upwind differencing scheme.

The closer a scheme is located towards the upwind differencing line on the normalised variable diagram the more stable and diffusive the scheme will be. The contrary is true for schemes that are located closer to the downwind differencing (DD) line, Figure 3.3. These schemes will be more compressive at the cost of stability (Ubbink, 1997). Schemes based on the normalised variable diagram switch locally between different convection schemes based on the normalised donor cell value. In regions where the donor cell value is bounded between the upwind and acceptor cell values, i.e when $0 < \tilde{\phi}_D < 1$ higher-order face value interpolations are applied. When the donor cell value is unbounded, the first order upwind differencing scheme is applied for the face value calculation to restore boundedness.

3.5.2 Examples of high-resolution schemes

Examples of high-resolution schemes include Gamma (Jasak *et al.*, 1999), SOUCUP (Zhu and Rodi, 1991), SMART (Gaskell and Lau, 1988), ULTIMATE-QUICKEST (Leonard, 1991), COPLA (Zhu and Rodi, 1991), MUSCL (van Leer, 1997) and OSHER (Chakravarthy and Osher, 1983).

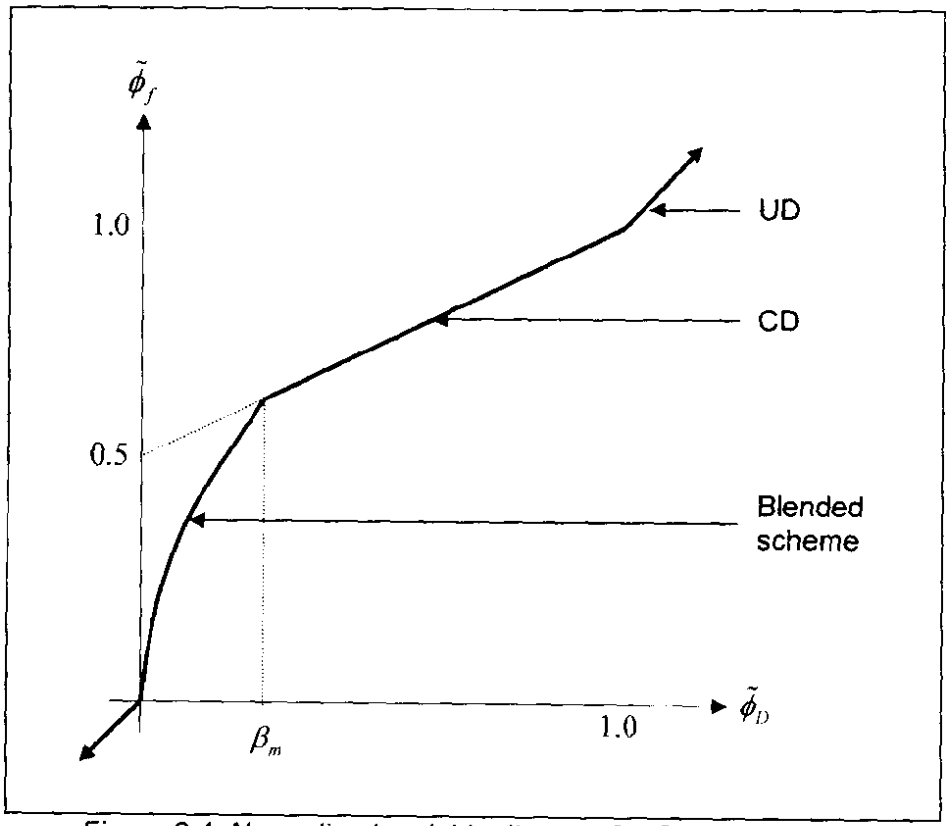


Figure 3.4: Normalised variable diagram for Gamma scheme

High-resolution schemes have also been developed specifically for interface capturing applications. These schemes are bounded and highly compressive in order to minimise numerical diffusion. Compressive schemes are in general not suitable for applications other than where steep gradients in the dependent variable are present as these schemes tend to

change any finite gradient into a step profile (Ubbink, 1997). Examples of high-resolution interface capturing schemes include CICSAM (Ubbink and Issa, 1999), HRIC (Muzaferija and Peric, 1998) and Inter-Gamma (Jasak and Weller, 1995).

The Gamma and Inter-Gamma schemes are used extensively in this thesis. These schemes are discussed in more detail in the following sections.

3.5.2.1 Gamma

The Gamma scheme (Jasak *et al.*, 1999) uses a local blend between central differencing and upwind differencing with an adaptive filter to maintain monotonic solutions of the dependent variable. The value of the normalised donor cell indicates whether the solution is bounded or not within the discretisation stencil. If unbounded, upwind differencing is used for the face value calculation; if bounded, either central differencing is used or a blend between central differencing and upwind differencing based on the value of the normalised donor cell value. A factor β_m ranging between 0.1 and 0.5 is specified to control the blending characteristics of the Gamma differencing scheme.

The normalised variable diagram for the Gamma scheme is shown in Figure 3.4. The Gamma scheme is not a function of the Courant number. The scheme is therefore efficient for transient simulations of steady state flow since large time step sizes are allowed. For transient flow however, caution should be exercised when applying explicit time integration since unbounded solutions could result from violating the Courant limit of one. When using the Crank Nicolson time integration scheme which is second order accurate, the time step limit is only slightly less restrictive than for explicit time integration. The functional relationship of the Gamma scheme is summarised as follows:

$$\tilde{\phi}_f = \begin{cases} \tilde{\phi}_D & \text{if} \quad \tilde{\phi}_D \le 0 \quad \text{or} \quad \tilde{\phi}_D \ge 1 \\ \frac{1}{2} + \frac{1}{2}\tilde{\phi}_D & \beta_m < \tilde{\phi}_D < 1 \\ -\frac{\tilde{\phi}_D^2}{2\beta_m} + \left(1 + \frac{1}{2\beta_m}\right)\tilde{\phi}_D & 0 < \tilde{\phi}_D \le \beta_m \end{cases} \tag{3.25}$$

The Gamma scheme is a bounded version of the central differencing scheme.

3.5.2.2 Inter-Gamma

The Inter-Gamma scheme (Jasak and Weller, 1995) was specifically developed for interface tracking applications. The Inter-Gamma scheme has the necessary compressive characteristics that are required for interface tracking. The normalised variable diagram for the Inter-Gamma scheme is shown in Figure 3.5.

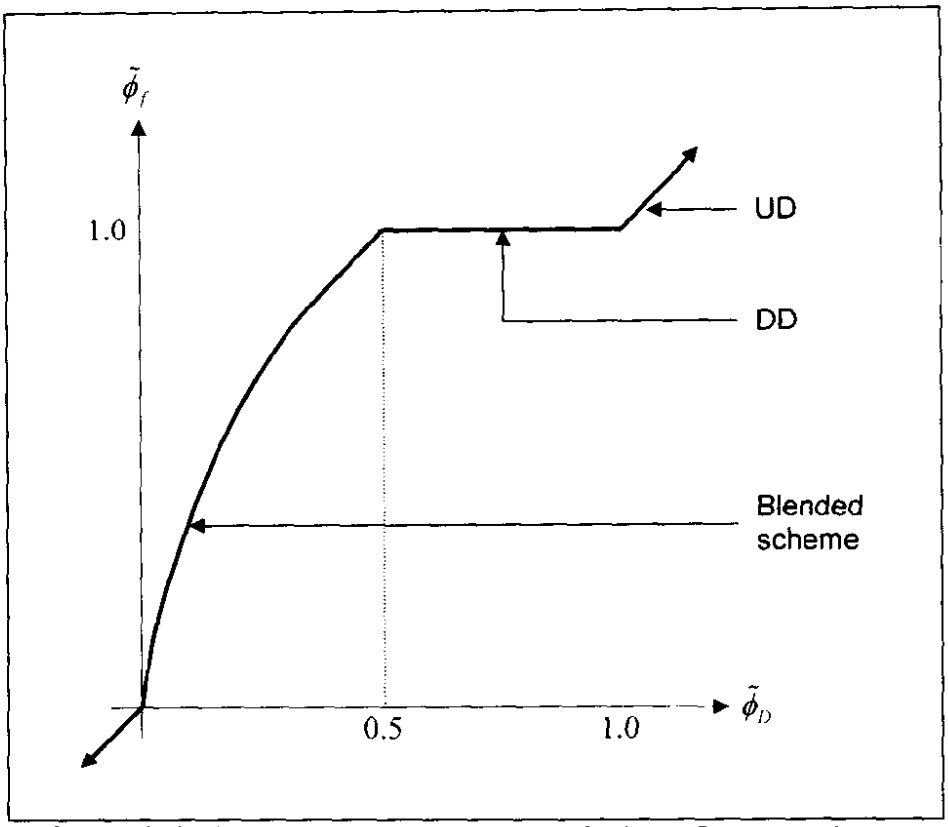


Figure 3.5: Normalised variable diagram for Inter-Gamma scheme

The scheme is divided into three regions on the normalised variable diagram, namely:

1. $\tilde{\phi}_D \leq 0$ or $\tilde{\phi}_D \geq 1$, $\tilde{\phi}_f = \tilde{\phi}_D$. The upwind differencing scheme is used in this region.

2. $0.5 < \tilde{\phi}_D < 1$, $\tilde{\phi}_f = 1$. The downwind differencing scheme is used in this region to provide the compressive behaviour of the scheme.

3. $0 < \tilde{\phi}_D < 0.5$, $\tilde{\phi}_f = -2\tilde{\phi}_D^2 + 3\tilde{\phi}_D$. This functional relationship creates a smooth transition from upwind to downwind differencing over this range of normalised donor cell values.

The Inter-Gamma scheme is expected to preserve boundedness while maintaining a sharp resolution of the interface, (Jasak and Weller, 1995). Numerical diffusion is introduced to preserve boundedness while downwind differencing is applied to reconstruct the sharp

interface. Jasak and Weller (1995) recommend that a Courant number of less than 1/3 be used for multidimensional interface tracking simulations and less than 1/2 for one-dimensional simulations.

The one-dimensional convection example described in section 3.4 was repeated using the Gamma and Inter-Gamma schemes with all other variables remaining unchanged. In this example the Gamma scheme was used with the recommended blending factor of $\beta_m = 0.5$, (Jasak *et al.*, 1999). The results are shown on the graph in Figure 3.6, including the results for the upwind differencing scheme for comparison purposes.

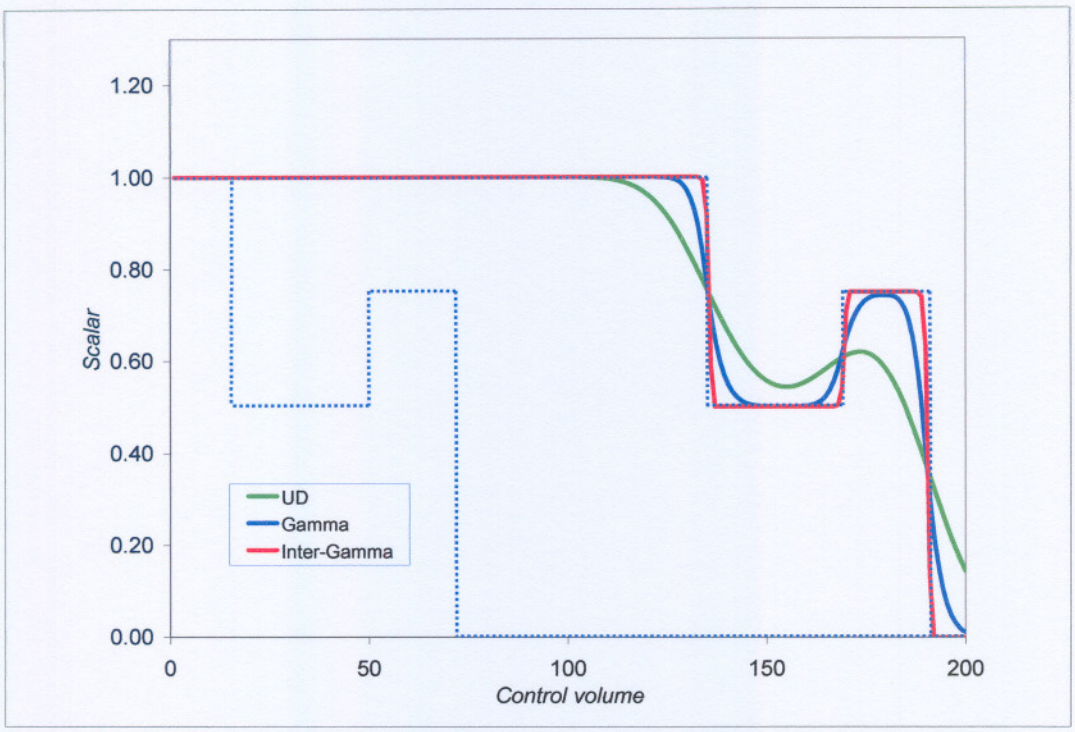


Figure 3.6: One-dimensional convection: UD, Gamma & Inter-Gamma

Both the Gamma and Inter-Gamma schemes produce bounded results. The Gamma scheme solution follows the block wave profile reasonably well while the Inter-Gamma scheme produces a solution that closely resembles the analytical solution. This example clearly demonstrates the capabilities of high-resolution schemes for volume tracking applications.

3.5.3 Implementation of high-resolution schemes

3.5.3.1 Deferred correction method

High-resolution schemes are incorporated into existing low order codes using the method of deferred correction (DC). Higher-order schemes apply extended stencils, consisting of the cells bracketing the face as well as additional cells to allow for a higher-order interpolation of the face value. The direct implementation of such schemes is complex and scheme dependent. Deferred

correction, introduced by Khosla and Rubin (1974) facilitate the implementation of higher-order flux approximations without the disadvantage of increased computational molecules. This is achieved by calculating the higher-order flux approximations explicitly, while the coefficient matrix is assembled using the first order upwind differencing scheme which guarantees a positive coefficient matrix. The higher-order contribution to the face flux is incorporated into the source term of the algebraic equation. The price to be paid for this convenience is a reduced rate of convergence due to the explicit implementation (Darwish and Moukalled, 1996).

When using the method of deferred correction, the higher-order flux approximation is combined with an implicit lower order approximation that requires only the coefficients of the cells bracketing the face across which the flux is calculated (Ferziger and Peric, 1999).

$$F_f = F_f^L + \left(F_f^H - F_f^L\right)^{old} \tag{3.26}$$

In Eq. (3.26), F_f is the convective flux approximation, F_f^L is the flux calculated with the lower order scheme, for example upwind differencing and F_f^H the higher-order flux approximation. The superscript old indicates that the term in brackets is evaluated using the results from the previous iteration. Upon convergence the lower order flux approximations of the current and previous iterations are equal.

$$F_f^L = \left(F_f^L\right)^{old} \tag{3.27}$$

The lower order flux approximations therefore sums to zero, leaving only the higher-order flux approximation, Eq. (3.28).

$$F_f = \left(F_f^H\right)^{old} \tag{3.28}$$

The partially discretised form of the general convective transport equation for time-dependent flow calculations is written as:

$$\left(\frac{\partial}{\partial t}(\rho\phi)\right)_P V_P + \sum_{f=1}^{n}\left(G_f\phi_f\right) = S_b V_P + S_P V_P \phi_P \tag{3.29}$$

The convective flux term in Eq. (3.29) is written in the following form, incorporating deferred correction into the formulation:

$$\sum_{f=1}^{n} G_f \phi_f = \sum_{f=1}^{n} F_f = \sum_{f=1}^{n} \left[F_f^L + \left(F_f^H - F_f^L \right)^{old} \right] \tag{3.30}$$

The explicit contribution in (3.30) is calculated for every face of the control volume and then transferred to the source term of the equation. Each control volume face therefore contributes to the source term of the discretised equation of that control volume.

$$\left(\frac{\partial}{\partial t} (\rho \phi) \right)_P V_P + \sum_{f=1}^{n} F_f^L = S_b V_P + S_P \phi_P V_P - \sum_{f=1}^{n} \left(F_f^H - F_f^L \right)^{old} \tag{3.31}$$

The convective flux on the left hand side of Eq. (3.31) is modelled using the upwind differencing scheme.

$$\sum_{f=1}^{n} F_f^L = \sum_{f=1}^{n} G_f \phi_D \tag{3.32}$$

A blending factor β_{UDS} can be added to the flux expression (3.26) to blend the higher-order flux approximation with the first order upwind differencing flux approximation.

$$F_f = F_f^L + \beta_{UDS} \left(F_f^H - F_f^L \right)^{old} \quad 0 \le \beta_{UDS} \le 1 \tag{3.33}$$

Any higher-order scheme can be implemented using the method of deferred correction without any fundamental changes to the internal structure of the code. The explicit treatment of higher-order flux terms weakens the linkage between individual equations, thereby affecting the convergence rate (Darwish and Moukalled, 1996). This is because individual equations are linked together through the convective and diffusive flux terms.

3.5.3.2 Downwind weighting factor method

An alternative implementation of high-resolution schemes is through the use of the downwind weighting factor (DWF) method, developed by Leonard and Mokhtari (1990), whereby a weighting factor β_f is calculated to produce a face value ϕ_f based on a weighting between the donor and acceptor cells bracketing the face under consideration.

$$\phi_f = \left(1 - \beta_f \right) \phi_D + \beta_f \phi_A \tag{3.34}$$

The downwind weighting factor β_f for the upwind and central differencing schemes is not a function of the solution, but it is for schemes based on the normalised variable diagram:

$$\beta_f = f(\phi) \tag{3.35}$$

The weighting factor is a function of the normalised donor and normalised face value.

$$\beta_f = \frac{\tilde{\phi}_f - \tilde{\phi}_D}{1 - \tilde{\phi}_D} \tag{3.36}$$

The weighting factor can also be expressed in equivalent un-normalised form as:

$$\beta_f = \frac{\phi_f - \phi_D}{\phi_A - \phi_D} \tag{3.37}$$

As an example the first order upwind differencing scheme which requires that $\tilde{\phi}_f = \tilde{\phi}_D$ may be considered. Substituting this condition into (3.36), yields a downwind weighting factor of zero which, when substituted into Eq. (3.34), produces a face value equal to the donor cell value as required.

$$\phi_f = \phi_D \tag{3.38}$$

The downwind weighting factor β_f is a blending factor which blends the first order upwind and first order downwind differencing schemes. The downwind component of the face value approximation can also be implemented by using deferred correction, while the upwind component is treated implicitly. The value of β_f indicates whether the calculated face value lies within the bounding values of the cells bracketing the face. If $\beta_f < 0$ or $\beta_f > 1$ the calculated face value is unbounded and may be bounded by adjusting the weighting factor to the nearest bounded value, either one if $\beta_f > 1$ or zero if $\beta_f < 0$. The method of deferred correction is obtained by substituting the weighting factor definition (3.37) into (3.34).

$$\begin{aligned} \phi_f &= (1 - \beta_f)\phi_D + \beta_f \phi_A \\ &= \phi_D + \left(\frac{\phi_f - \phi_D}{\phi_A - \phi_D} \right)(\phi_A - \phi_D) \\ &= \phi_D + (\phi_f - \phi_D) \end{aligned} \tag{3.39}$$

ϕ_f in (3.39) is the higher-order interpolated face value. By calculating the term in brackets in (3.39) explicitly, the standard deferred correction formulation is obtained from the downwind weighting factor formulation.

$$\phi_f = \phi_{f,D} + \left(\phi_f - \phi_{f,D}\right)^{old} \tag{3.40}$$

$\phi_{f,D}$ in (3.40) represents the upwind differencing scheme, which is treated implicitly in the deferred correction formulation.

The downwind weighting factor method allows for a "pseudo implicit" implementation with an explicit component due to the fact that the downwind weighting factor is calculated explicitly from the solution of the previous iteration. This implementation is unfortunately highly unstable and requires a small under-relaxation factor to improve stability. Negative coefficients in the coefficient matrix leads to unbounded results similar to the oscillations caused by the central differencing scheme for convection dominated flows (Darwish and Moukalled, 1996). Unbounded results can therefore be obtained regardless of the fact that the downwind weighting factor is calculated to conform to the convection boundedness criterion of Gaskell and Lau (1988).

Alternative implementations to the deferred correction and downwind weighting factor methods have been developed for the application of high-resolution schemes in CFD codes (Darwish and Moukalled, 1996). These implementations are however not considered in this thesis.

Both the method of deferred correction as well as the downwind weighting factor method requires the face value of the dependent variable ϕ_f interpolated from cell values using higher-order interpolation schemes. High-resolution schemes which is based on the normalised variable diagram can be used to determine ϕ_f, but requires the calculation of the normalised donor cell value $\tilde{\phi}_D$, Eq. (3.18). For this calculation a three-point stencil is required. Three-point stencils are naturally available on structured orthogonal meshes. Normalised donor cell values can therefore be calculated without any problems. The availability of three-point stencils becomes an issue when using arbitrary unstructured meshes for simulations. The following section describes the implementation of high-resolution schemes for these meshes.

3.5.4 High-resolution schemes for arbitrary unstructured meshes

Three-point stencils are required for the application of high-resolution schemes, providing upstream information for face value calculations. The selection of an upwind cell for a face is not a trivial matter. The problem is further complicated for meshes with highly deformed control volumes i.e. large degrees of non-orthogonality.

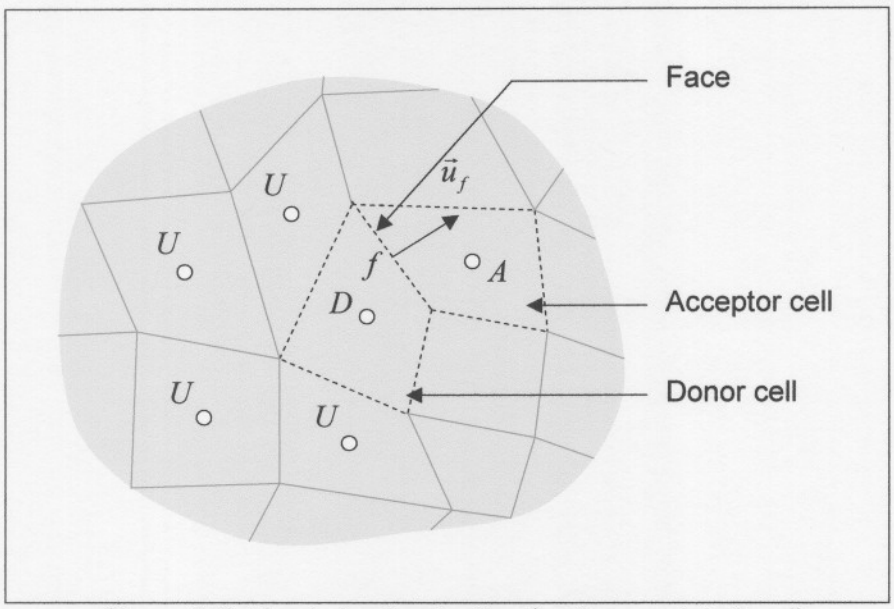


Figure 3.7: Upwind cell selection for three-point stencil

Figure 3.7 shows a section of a typical arbitrary unstructured mesh. The figure shows that the donor and acceptor cells are determined by the direction of flow through the face. The problem is that of selecting an appropriate upwind cell value, of which a few possible candidates are shown. Jasak *et al.* (1999) addressed this problem by adopting a method for calculating the normalised donor cell value directly by making use of the gradient of the dependent variable over the donor cell.

By eliminating the upwind cell from the formulation, this method overcomes the problem of selecting an appropriate upwind cell. This method is equivalent to Eq. (3.18) for an orthogonal mesh.

$$\begin{aligned}\tilde{\phi}_D &= 1 - \frac{\left(\vec{\nabla}\phi\right)_f \cdot \vec{d}}{2\left(\vec{\nabla}\phi\right)_D \cdot \vec{d}} \\ &= 1 - \frac{\phi_A - \phi_D}{2\left(\vec{\nabla}\phi\right)_D \cdot \vec{d}}\end{aligned} \tag{3.41}$$

In Eq. (3.41) $\vec{d}$ is the vector connecting the centres of the donor and acceptor cells. Ubbink (1997) proposed an alternative formulation to Eq. (3.41) by calculating a projected upwind cell value from the gradient of the dependent variable over the donor cell.

$$\phi_U^* = \phi_A - 2\left(\vec{\nabla}\phi\right)_D \cdot \vec{d} \tag{3.42}$$

The normalised donor cell value is then calculated using the standard formulation, Eq.(3.18). Figure 3.8 shows the upwind cell approximations for both flow directions across a face.

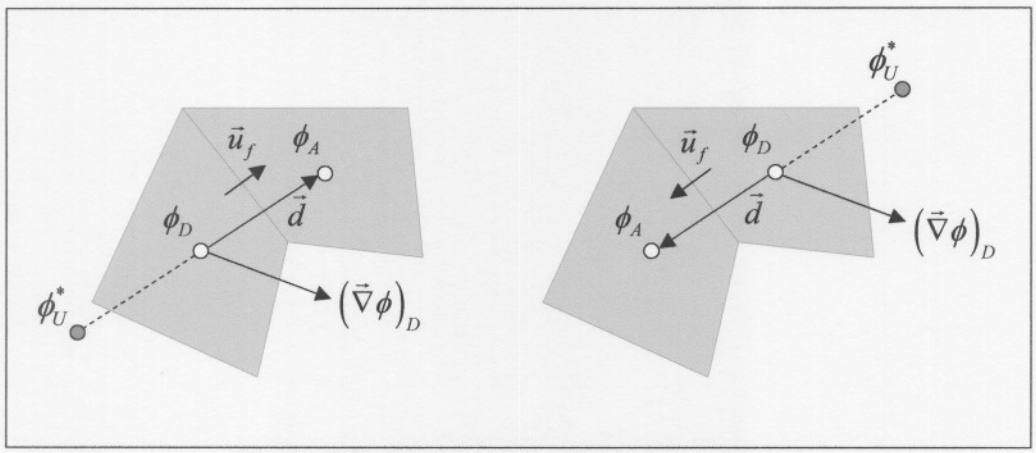


Figure 3.8: Upwind cell value approximation

In both Eq. (3.41) and (3.42) the gradient of ϕ over the donor cell is calculated using Eq. (3.43) which is derived from the theorem of Gauss.

$$\left(\vec{\nabla}\phi\right)_P \approx \frac{1}{V_P}\sum_{f=1}^{n}\phi_f \vec{A}_f \tag{3.43}$$

The face value ϕ_f in Eq. (3.43) is calculated using linear interpolation from the cell values bracketing the face. V_P is the volume of the donor cell.

An important issue arises from using these two methods to obtain the normalised donor cell value. Neither of the two formulations guarantees a bounded upwind cell value (Ubbink, 1997). Although the method of Jasak *et al.* (1999) does not explicitly calculate an upwind value, the normalised donor cell value calculated using this technique, does not guarantee a bounded solution.

Ubbink (1997) elaborates on this issue with reference to the specific application of volume capturing schemes for multiphase flow simulations. A solution to the unboundedness that may

occur when using Eq. (3.42) to project an upwind value is to bound the projected value explicitly between the known bounds of the solution when the value is unbounded.

$$\begin{aligned} \phi_U^* &= \phi_{\text{upper bound}} \quad \text{if} \quad \phi_U^* > \phi_{\text{upper bound}} \\ \phi_U^* &= \phi_{\text{lower bound}} \quad \text{if} \quad \phi_U^* < \phi_{\text{lower bound}} \end{aligned} \tag{3.44}$$

For multiphase flow applications where volume fractions are convected across the mesh, the physical bounds on the solution are zero and one. These known physical bounds of the solution (3.44) is used to bound the upstream value if it is unbounded before the normalised donor cell value is computed.

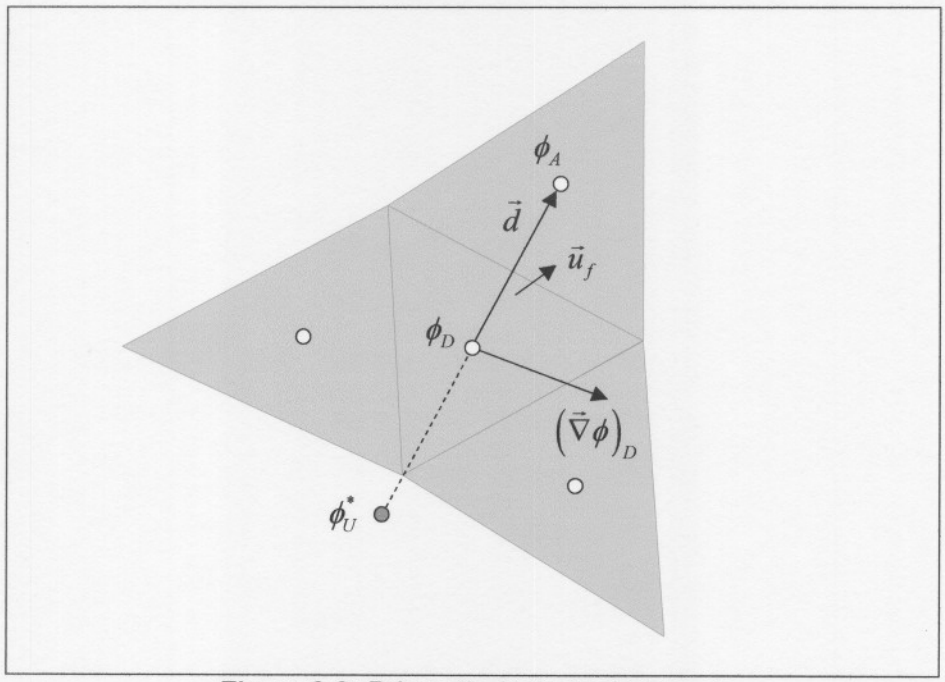


Figure 3.9: Prismatic / tetrahedral mesh

The problem with general simulations is that the bounds of the solution are seldom known. The general transport equation contains source terms which results in a solution that may well be unbounded when considering the boundary conditions applied to a problem. One example is flow through a nozzle. The velocity in the throat area of the nozzle will be higher than the inlet or outlet velocities due to continuity. Although unbounded with regard to boundary conditions due to the presence of source terms, the solution should always be monotonic, therefore free from non-physical oscillations.

Figure 3.9 shows a section of a mesh consisting of triangular prism cells. The method proposed by Jasak *et al.* (1999), Eq. (3.41), may fail to accurately predict an upwind cell value. Calculating the gradient over the donor cell using Eq. (3.43) involves the donor cell value as well as its neighbour cells. These cell values are used to calculate the face values ϕ_f in Eq. (3.43). In

Figure 3.9 the cell in which ϕ_U^* resides, is not even accounted for in the gradient calculation of the donor cell. The same problem may be encountered when using tetrahedral meshes which is widely used for automatic mesh generation. The issue of obtaining three-point stencils for high-resolution schemes on arbitrary unstructured meshes is addressed in the following section through a method that is fundamentally based on volume weighted interpolation.

3.6 Orthogonal projection interpolation stencil

3.6.1 Introduction

Figure 3.10 shows two control volumes that form part of an arbitrary unstructured mesh. One of the cells is labelled as the owner cell and the other the neighbour cell of face f. These *cell identities* are assigned during mesh generation and determine the orientation of the face area vector which points from the owner cell in the direction of the neighbour cell. A velocity vector interpolated to the face is also shown in the figure.

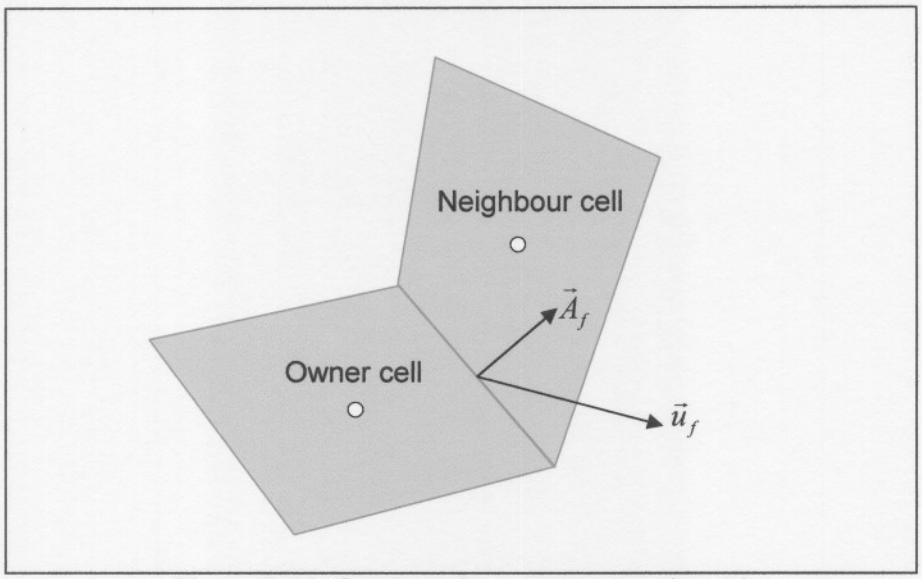


Figure 3.10: Section of an unstructured mesh

The concept behind the Orthogonal Projection Interpolation Stencil (OPIS) is to include additional cells in the face value interpolation and not only the cells bracketing a face, which on an arbitrary unstructured mesh, are the only information that is available to the face. One way to include additional cells in convective flux calculations involves the construction of a regular orthogonal mesh on the face between the two cells. Figure 3.11 shows a schematic representation of the regular mesh constructed orthogonal to face f.

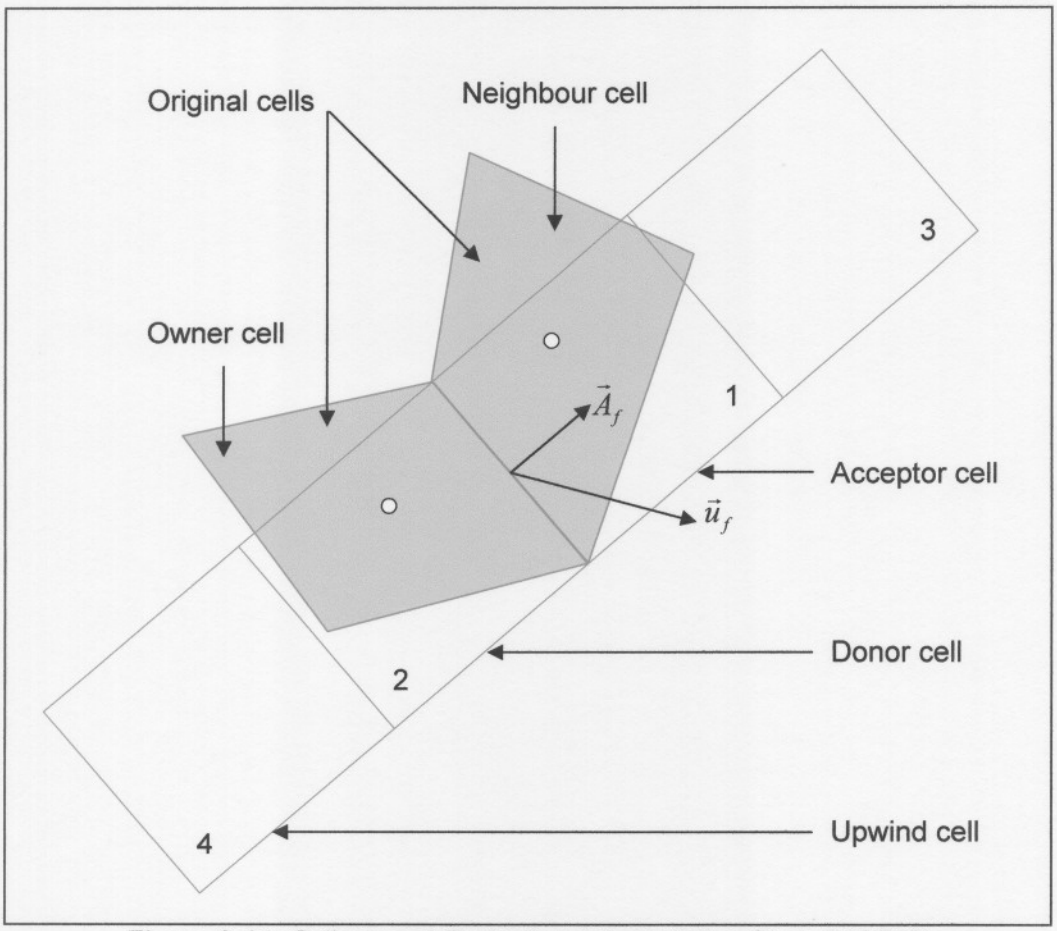


Figure 3.11: Orthogonal Projection Interpolation Stencil (OPIS)

In this thesis the terms *regular mesh* and *regular cell(s)* refer to cells constructed to facilitate face value interpolations. The regular mesh consists of four cells, which are always numbered as shown in the figure in relation to the orientation of the face area vector. In the finite volume method cell values represent the average value of a flow quantity inside a control volume. For the Orthogonal Projection Interpolation Stencil the field values ϕ of the original mesh are used to calculate values for the regular cells constructed normal to the face. In this process original mesh cells overlapped by regular cells contribute to the calculation of face values.

Higher-order interpolation requires the availability of a donor, acceptor and upwind cell, which is determined by the flow direction through the face on which the regular mesh is constructed, Figure 3.11. The four cells allow the stencil to be used for both flow directions. OPIS also addresses the issue of mesh skewness, since the regular cells are constructed normal to a face and in terms of the regular cells; the face centre is located centrally. Figure 3.12 shows the triangular prismatic mesh with the outlines of a few additional mesh cells also shown.

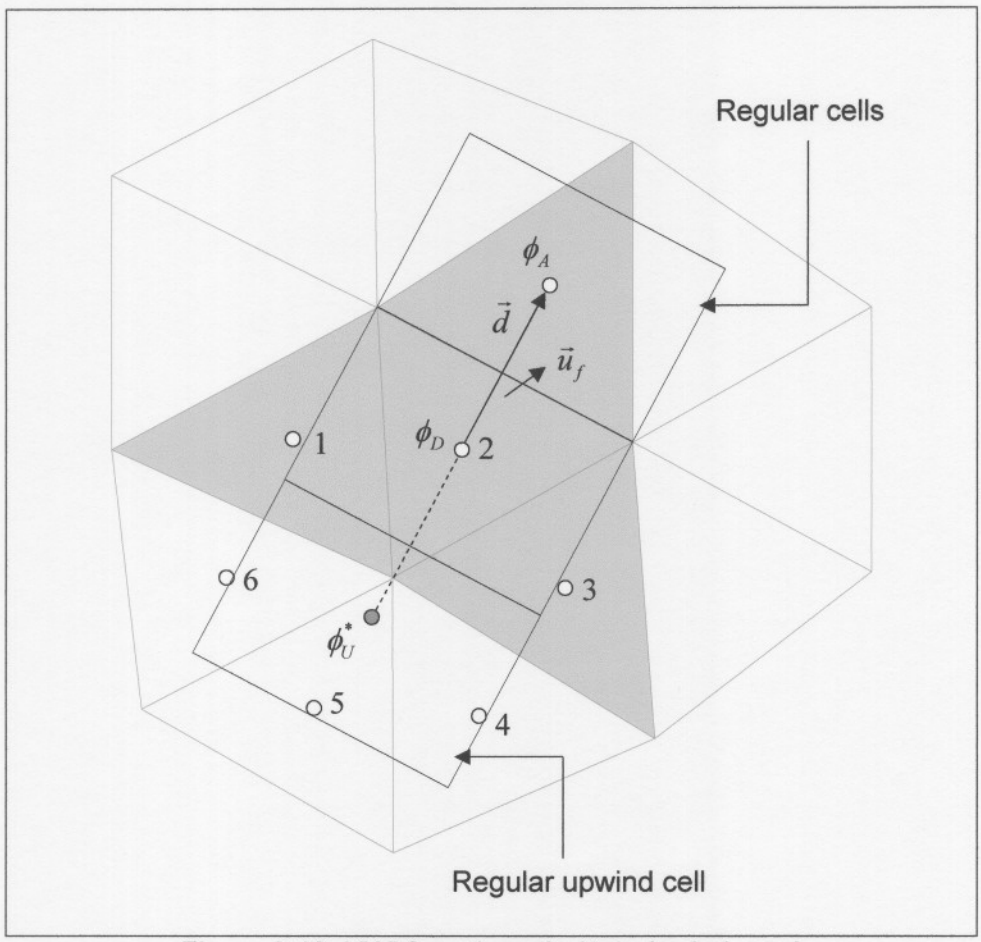


Figure 3.12: VWI for prismatic / tetrahedral mesh

Instead of involving only the donor and its neighbour cells in the upwind cell calculation, the weighted upwind cell value includes contributions from several cells numbered one to six in the figure. The contribution of each cell to the regular cell is determined by the fraction of volume overlapped between the cell and the regular cell. The concept of volume weighted interpolation for modelling convective transport is further explored in the remainder of this chapter.

Figure 3.13 shows a section of an unstructured mesh consisting of hexahedral cells, shown as quadrilateral cells in this simplified two-dimensional diagram. Indicated on this figure is an internal face f with a set of four regular cells constructed orthogonal to the face, as well as the original owner and neighbour cells bracketing the face. Only three of the four regular cells are shaded. This selection is determined by the flow direction across the face during run-time. The shaded cells are used in the calculation of the face value of the dependent variable and represent the donor, acceptor and upwind cells as indicated in the figure. OPIS calculates a representative cell value for each regular cell based on a conservative volume weighted interpolation from the original mesh. The face value is calculated from these regular cell values using an appropriate high-resolution convection scheme.

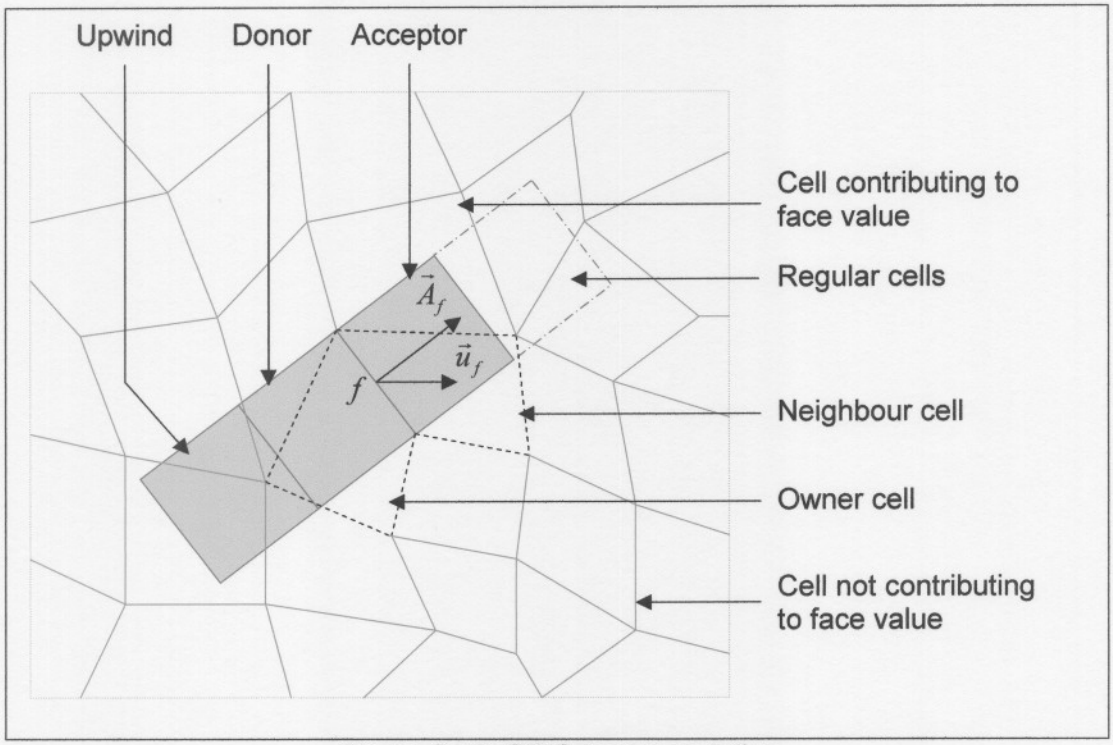


Figure 3.13: OPIS implementation

OPIS addresses the problem of obtaining a bounded upwind cell value required for high-resolution schemes on arbitrary unstructured meshes where upwind cells are not readily available or accessible. OPIS can also be used for extrapolation to boundary faces by constructing orthogonal cells to the interior of the mesh and mapping the solution to these cells where after the regular cell data is used for extrapolation purposes.

3.6.2 Construction of regular cells

Meshes consisting of four regular cells are constructed for each inner face of the computational mesh. Alternatively mesh quality tests can be implemented and OPIS applied only at the faces where required, i.e. in regions consisting of highly deformed cells. This will improve the overall performance by applying volume weighted interpolation only where required.

The four regular cells are constructed orthogonal to the face by projecting new corner vertices from the original face vertices in the direction of the face area vector. The volumes of the projected cells determine the *region of influence* in the face value calculation and may be adjusted by the user. There are several possibilities to calculate the regular cell volumes. The method used in this thesis is shown in Figure 3.14. On an orthogonal mesh with equally spaced cells this method produces a regular cell volume that is equal to the cell volumes of the mesh.

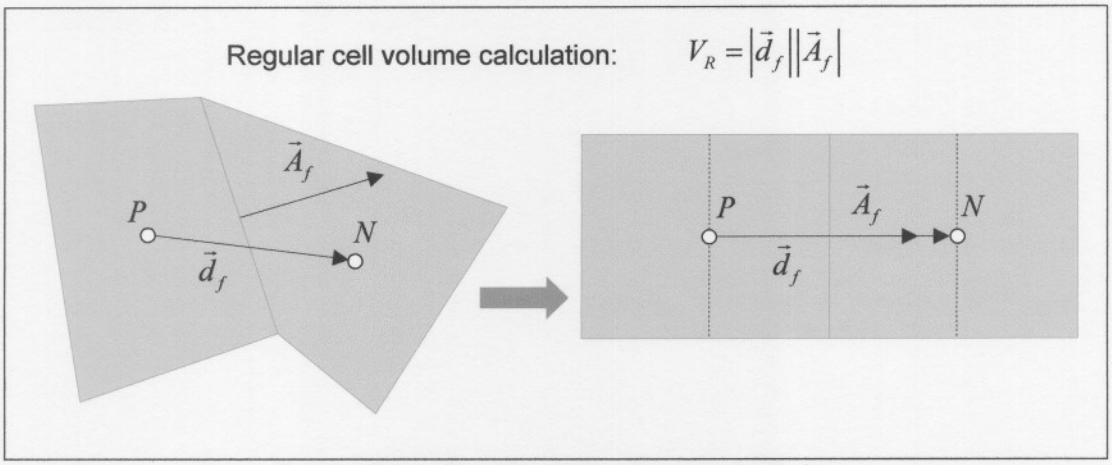


Figure 3.14: Regular cell volume calculation

Although not strictly required, the regular mesh is constructed of cells with equal volume. Using cells with equal volume and therefore equal dimensions, simplifies the interpolation of face values. With the volume of the regular cells available, the orthogonal projection length d_{pr} is determined as follows:

$$d_{pr} = \frac{V_R}{|\vec{A}_f|} \tag{3.45}$$

In Eq. (3.45), V_R is the volume of the regular cell. The four regular cells are constructed by projecting new corner nodes from the original face using d_{pr} in the direction of the face area vector $\vec{A}_f$.

The flow direction through a face is determined at run-time from the sign of the mass flow across the face which is based on the orientation of the face area vector used in the construction of the regular mesh. For each face the upwind, donor and acceptor cells are assigned with a regular cell number.

$$\left.\begin{aligned} U &= Cell\ 4 \\ D &= Cell\ 2 \\ A &= Cell\ 1 \end{aligned}\right\} \quad if \quad G_f \geq 0$$

$$\left.\begin{aligned} U &= Cell\ 3 \\ D &= Cell\ 1 \\ A &= Cell\ 2 \end{aligned}\right\} \quad if \quad G_f < 0 \tag{3.46}$$

The flow direction through the face is checked after each iteration to account for possible flow reversals.

3.6.3 Data structures for regular cell storage

For each inner face of the original mesh, a list is constructed consisting of the four regular cells and their calculated corner vertices, as shown in Figure 3.15.

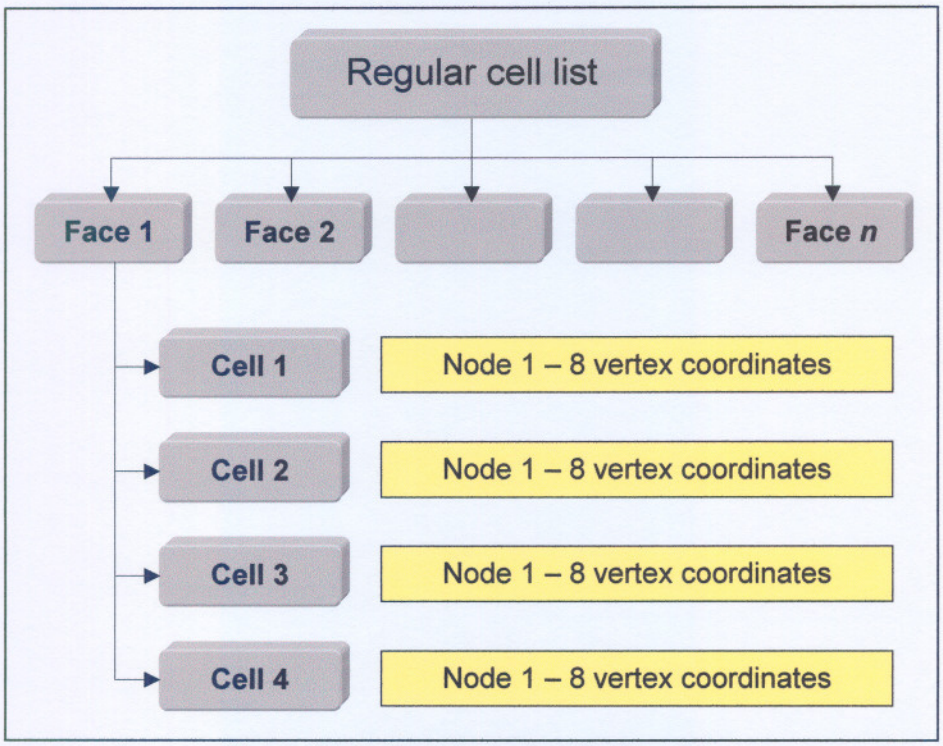


Figure 3.15: Regular cell definition data

This list is constructed once before the start of a simulation for a stationary mesh. The scheme therefore suffers little in terms of computational overhead. For each of the cells one to four, a list of cells is created, identifying the cells in the original mesh that overlap the regular cell. The number of entries in this list is determined by the mesh structure and will differ from cell to cell and from face to face. The number of cells overlapping a regular cell depends on the *degree of unstructuredness* of the mesh.

A bucket search algorithm is used to search for cells overlapping each regular cell in the mesh. When a cell is found that overlaps a regular cell, it is added to the overlapping list and the overlapped volume for that cell is stored. The search is continued until no further overlapping cells are found for that regular cell. The result of this search is a set of volume overlapping data for the mesh as shown in Figure 3.16.

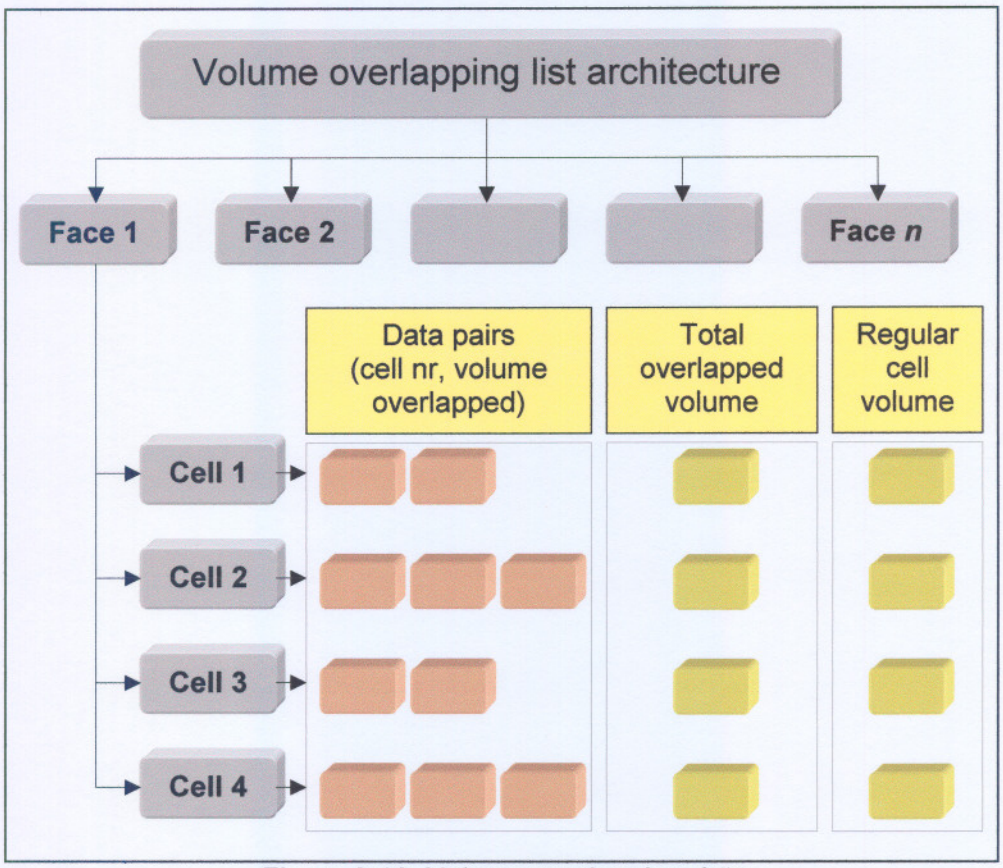


Figure 3.16: Volume overlapping data

This dynamic data structure can easily be accessed by the solver to calculate the volume weighted values of the regular cells for each face. The application of volume weighted interpolation to simulations on dynamic meshes will impose a severe penalty in terms of simulation time since the volume overlapping data will have to be recalculated after each change in the computational mesh, instead of only once before the start of a simulation.

3.6.4 Face value interpolation

Once the regular cell values of the dependent variable have been determined from the conservative interpolation, the face value is calculated. In principle, any appropriate interpolation scheme can be used, for example upwind differencing, central differencing or QUICK, but the stencil is best suited for differencing schemes based on the normalised variable diagram to take advantage of the boundedness characteristics of these schemes. Instead of using all three regular cells per face, it is also possible to use only the predicted upwind cell value to calculate the normalised donor cell value. This technique is required for highly compressive schemes where overfilling or over emptying of the cells bracketing a face is possible due to the large gradients present in the solution. One example is interface tracking for two-phase flow simulations.

The OPIS scheme is implemented using the method of deferred correction combined with the downwind weighting factor method where the face value is expressed as a weighting between the values of the donor and acceptor cells of the face.

$$\phi_f = (1-\beta_f)\phi_D + \beta_f \phi_A \tag{3.47}$$

In the following section test cases are presented to demonstrate the capabilities of OPIS for modelling convection on arbitrary unstructured meshes.

3.7 Test cases

3.7.1 Introduction

In this section test cases are presented in which the Orthogonal Projection Interpolation Stencil is applied to the modelling of convection on arbitrary unstructured meshes. The equation to be solved in each case is Eq. (3.1) without any physical diffusion and source terms and with the volume fraction α as the dependent variable.

$$\frac{\partial \alpha}{\partial t} + \vec{\nabla} \cdot (\vec{u}\alpha) = 0 \tag{3.48}$$

The test cases are simple with known analytical solutions. No attempt is made to couple the solution of Eq. (3.48) to the solution of the momentum equations. Analytic velocity fields are used in all the test cases. This allows a comparison of the characteristics of convection differencing schemes to be made.

Each test case is performed using two different meshes. A structured orthogonal mesh is used to perform a reference simulation. The simulation is repeated on an arbitrary unstructured mesh consisting of non-orthogonal cells. The two meshes are created to consist of approximately the same number of cells. The ideal case is when the simulation is performed on the orthogonal mesh where a three-point stencil is available for the application of high-resolution schemes. Two sets of simulations are performed on the unstructured mesh. The first uses Eq. (3.41) to determine the normalised donor cell value in the absence of a three-point stencil. The simulation on the unstructured mesh is then repeated using the Orthogonal Projection Interpolation Stencil. The results from the three simulations are then compared to determine how the convection schemes perform in each case.

3.7.2 Case 1: Rotational flow

In this test case the convection of a scalar block wave in a rotational velocity field is modelled. Figure 3.17 shows a schematic representation of the domain and configuration for this test case.

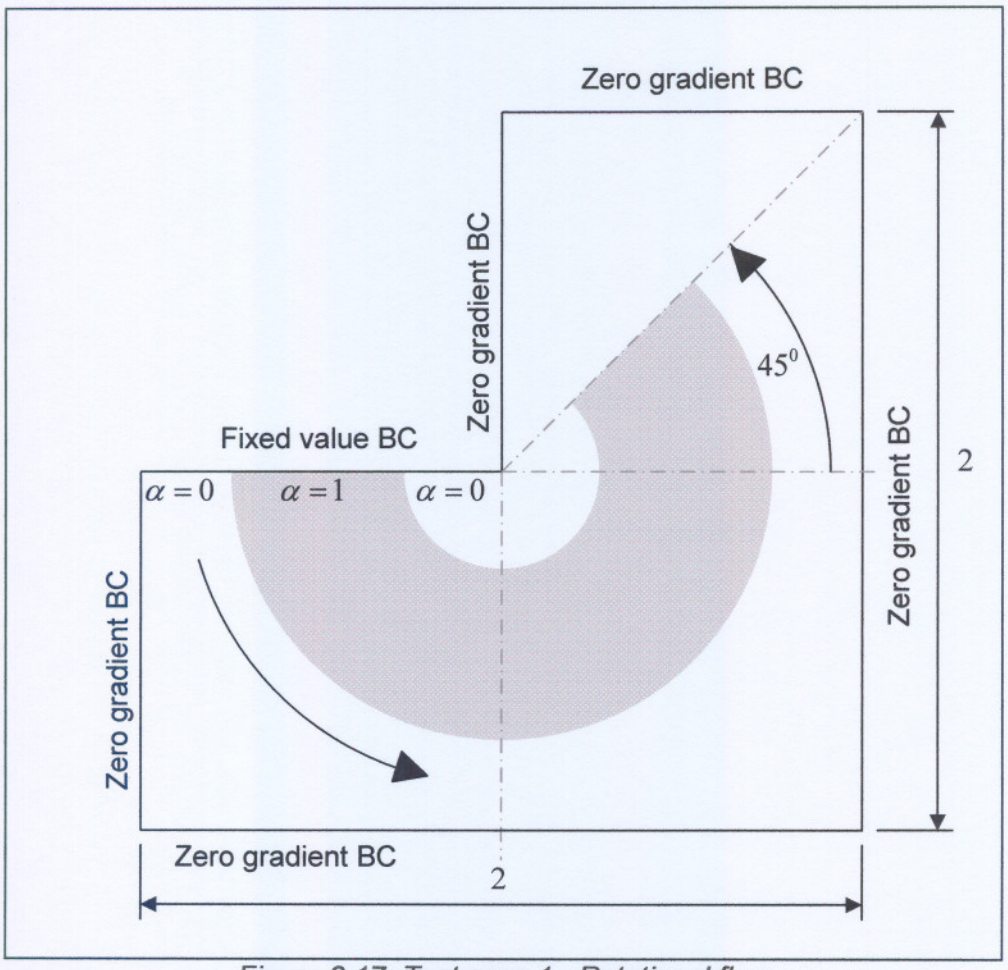


Figure 3.17: Test case 1 - Rotational flow

The block wave is defined using a fixed value boundary condition and convected into the domain by the rotational velocity field. Zero gradient boundary conditions are applied at all the remaining boundary surfaces. The initial condition is zero volume fraction values for all the cells in the mesh. An anticlockwise rotational velocity field with a magnitude of 1 rad/s is defined with the origin located at the centre of the domain.

The analytical solution for this test case is shown after the block profile was convected through 225°. While this test case may seem trivial, it presents several modelling challenges. Firstly, the solution should be bounded with cell volume fractions between zero and one throughout the domain. Secondly, the block wave profile should be convected without diffusing the interface which is defined by cell values between zero and one. Thirdly, the solution should closely

resemble the analytical solution shown in Figure 3.17 after the required number of time steps in a transient simulation. The solution should therefore be time accurate. At the given rotational speed of 1 rad/s after a given time interval, the front should be at the same position where the analytical solution would be at that time. The spatial and temporal integration of the convection equation must therefore be accurate. In this test case different convection schemes are used to show the characteristics of these schemes while the temporal integration is performed using the Crank Nicolson scheme. The time step size is determined from the specified Courant number, the velocity and the mesh characteristics.

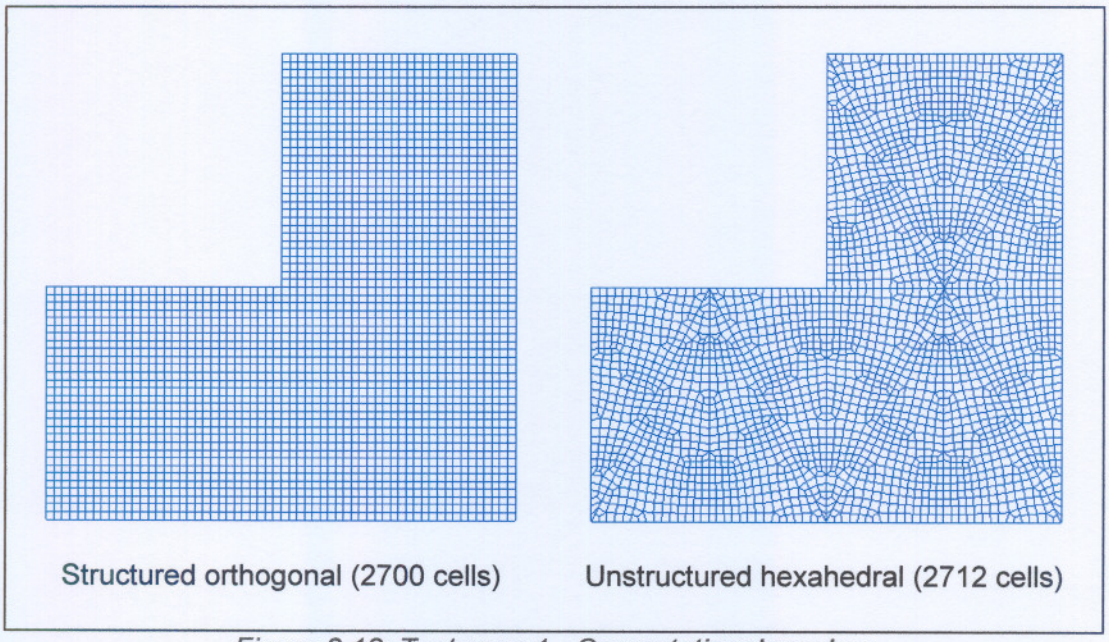


Figure 3.18: Test case 1 - Computational meshes

Figure 3.18 shows the two computational meshes that are used in this test case. The orthogonal mesh consists of 2700 cells while the arbitrary unstructured mesh consists of 2712 cells. The volume of each cell in the orthogonal mesh is 1.11e-4 m^3. The volume of the smallest and largest cells in the unstructured mesh is 4.27e-5 m^3 and 1.73e-4 m^3 respectively. The average cell value for the mesh is 1.106e-4 m^3. Figure 3.19 shows a histogram of the cell volume distribution of the unstructured mesh in ten equal volume increments between the minimum and maximum cell volumes.

The velocities for each cell in the two meshes are initialised by calculating the velocity components from the rotational velocity. The time required to complete 225° of rotation at a rotational speed of 1 rad/s, is 3.927 seconds. The Courant number for all the simulations for this test case is 0.2. For the orthogonal mesh the time step size for this Courant number equals 0.00344828 seconds, therefore requiring 1139 time steps to complete 225° rotation. For the unstructured mesh the time step size equals 0.0011392 seconds, requiring 3447 time steps to

complete 225° rotation. The smaller time step required for the unstructured mesh is due to the smaller cell sizes compared to the orthogonal mesh.

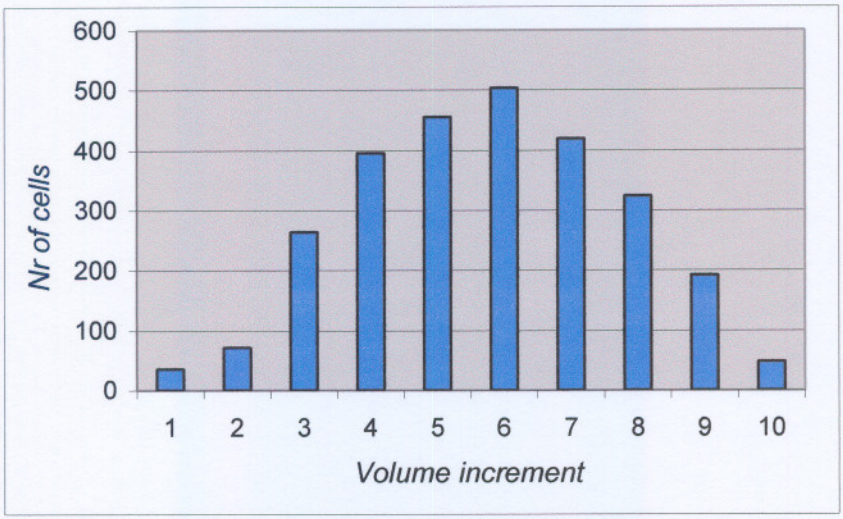


Figure 3.19: Cell volume distribution for unstructured mesh

The analytical solutions for the two meshes are shown in Figure 3.20.

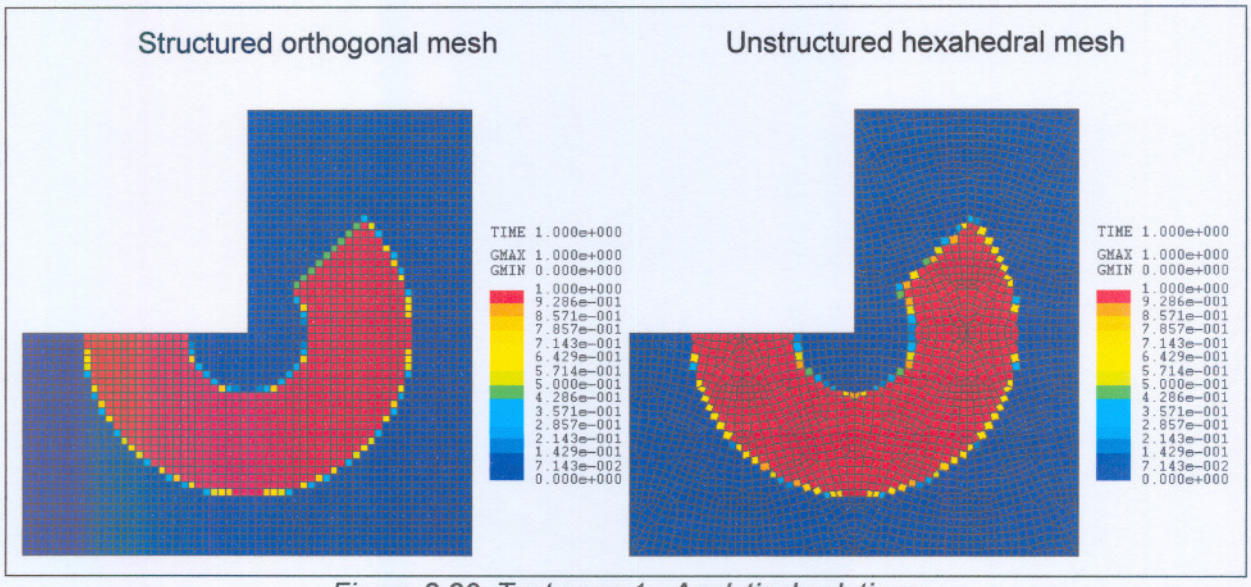


Figure 3.20: Test case 1 - Analytical solutions

The solutions shown in Figure 3.20 were obtained by mapping the final block wave position after 225° rotation onto the two meshes using the techniques described in Chapter 4. Cells with volume fraction values between zero and one indicate that the analytical solution passes through the cell. The volume fraction values of the analytical solution are plotted on the figure.

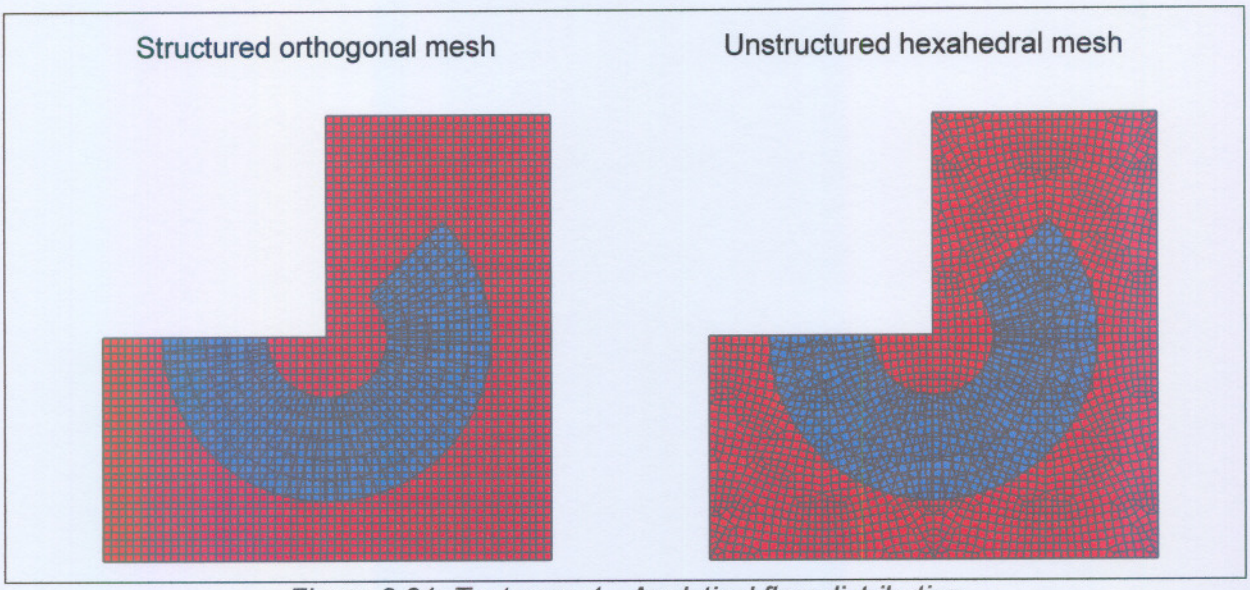


Figure 3.21: Test case 1 - Analytical flow distribution

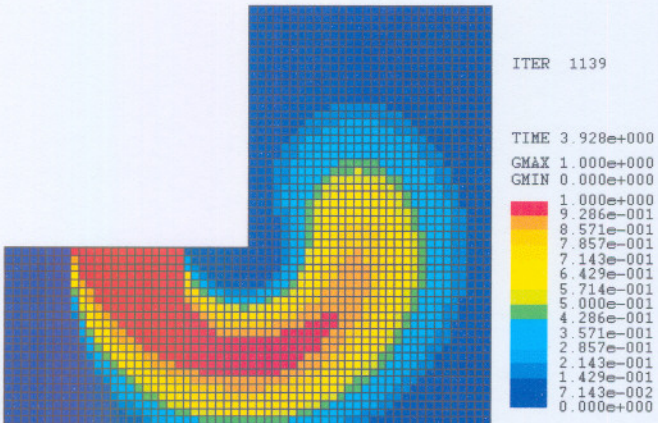
Figure 3.21 shows the analytical block wave distribution after 225° rotation represented by a cylindrical mesh overlapping the computational mesh. The analytical solutions allow error norms to be used to gauge the accuracy of the simulations. The solution error is defined as

$$E = \frac{\sum_{i}^{all\ cells} \left\| \alpha_i^n V_i - \alpha_i^a V_i \right\|}{\sum_{i}^{all\ cells} \alpha_i^a V_i} \tag{3.49}$$

where α^n is the solution after n time steps and α^a the analytical solution. V_i is the volume of cell i. The perfect convection scheme will result in a zero error computed with Eq. (3.49).

In the first part of this test case the upwind differencing, central differencing, QUICK, Gamma and Inter-Gamma schemes were used to model the block wave convection on the structured mesh to compare the characteristics of the different schemes and to compare the results with the analytical solution shown in Figure 3.20. The following figures show the results after 225° rotation. The error calculated for each simulation using Eq. (3.49) and comments on the results produced by each convection scheme are also shown. A blending control factor $\beta_m = 0.1$ is used for the Gamma scheme in Eq. (3.25).

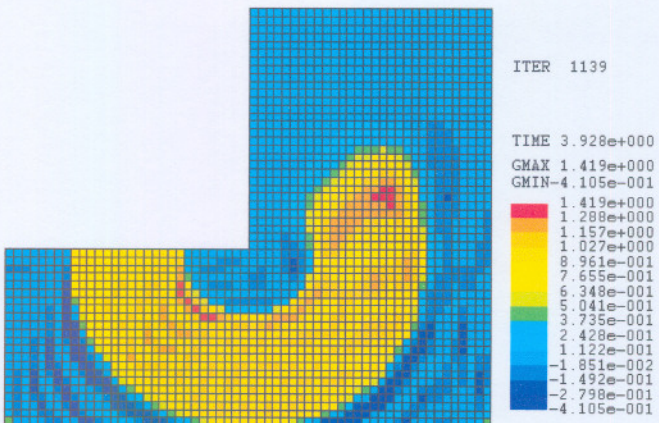
Upwind differencing (UD)



Solution error: 0.436

Although the solution is bounded between zero and one, significant levels of numerical diffusion are present. The block wave profile is diffused over several mesh cells, causing artificial mixing between the two fluids.

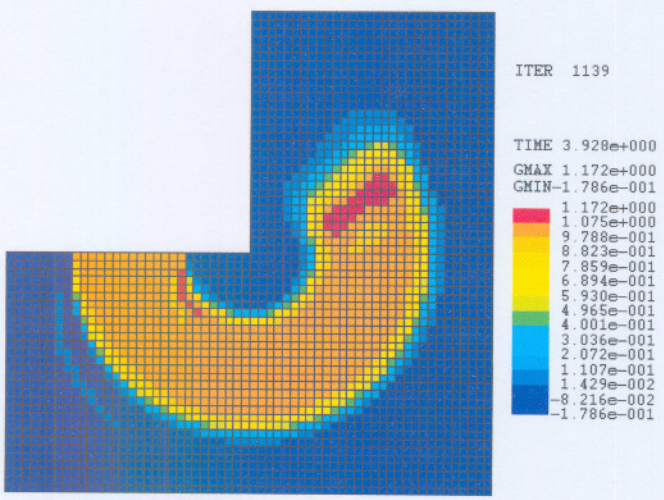
Central differencing (CD)



Solution error: 0.334

The solution error is smaller than that of the UD scheme with improved accuracy and less numerical diffusion. The solution is however unbounded with maximum and minimum values of 1.419 and -0.410 respectively. The numerical dispersion in the solution is clearly visible.

QUICK



Solution error: 0.124

The solution error is smaller than that of the CD scheme with improved accuracy and less numerical dispersion. The solution is still unbounded but a significant improvement over that of the UD and CD schemes.

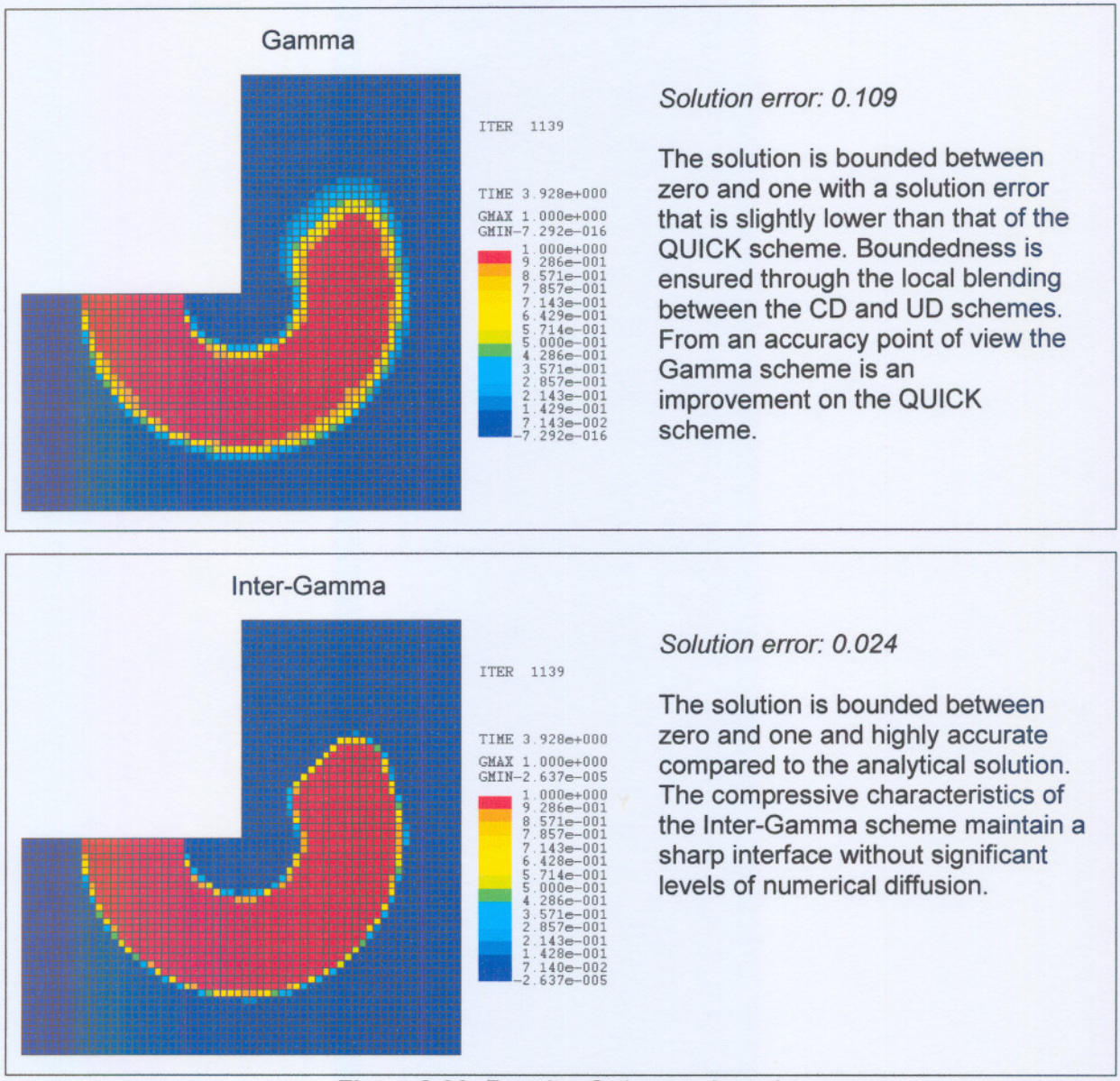


Figure 3.22: Results: Orthogonal mesh

Figure 3.22 shows the results obtained on the orthogonal mesh with the upwind differencing, central differencing, QUICK, Gamma and Inter-Gamma schemes. The results demonstrate the significant influence of the face value calculation for the convective term on the overall solution. The first order upwind differencing scheme (Courant *et al.*,1952) remains a popular choice in commercial CFD codes in spite of its highly diffusive characteristics. This is due to its stability and guaranteed boundedness. The upwind differencing scheme is usually the default selection in commercial CFD codes and therefore easily used without too much thought of the consequences by the inexperienced analyst. Although unconditionally bounded and stable, the upwind differencing scheme should be used with caution, and cannot be considered appropriate for any simulation.

The solution obtained with the central differencing scheme is clearly less diffusive than the solution with the upwind differencing scheme; however the unboundedness of the solution is evident from the peaks and troughs. The maximum and minimum cell values of the solution are 1.419 and -0.410 respectively. This unboundedness is unacceptable for two-phase flow simulations since volume fraction values larger than one and smaller than zero, is meaningless. The volume fraction indicates the portion of a cell filled with fluid one or two respectively. The results for the upwind, central and QUICK differencing schemes are in good agreement with the results of the one dimensional convection of a block wave shown in Figure 3.2.

The Gamma and Inter-Gamma schemes which are based on the normalised variable diagram, produces bounded solutions through the localised blending of different schemes. Eq. (3.41) is used to calculate the normalised donor cell value. It is evident that Eq. (3.41) works well for orthogonal meshes and maintains the boundedness of the solution. This can be explained by examining Eq. (3.41) in more detail. The calculation of the donor cell gradient directly involves the upwind cell value, which is bounded. The calculated upwind value will therefore not produce an unbounded value, thereby ensuring the boundedness of the solution. The Inter-Gamma scheme is highly compressive through the incorporation of downwind differencing into the face value calculation. The Gamma scheme is less compressive using a blend between the central and upwind differencing schemes. The results of this test case clearly demonstrate the advantages of using high-resolution schemes to maintain the boundedness and accuracy of the solution.

In the second part of this test case the Gamma and Inter-Gamma schemes are used to model the convection of the block wave on the arbitrary unstructured mesh. The results obtained when using Eq. (3.41) to calculate the normalised donor cell value are compared to the results obtained when using OPIS to construct three-point interpolation stencils.

Figure 3.23 shows the solutions obtained with the Gamma and OPIS-Gamma schemes on the unstructured mesh. The solution obtained with the Gamma scheme is unbounded whereas the solution obtained with OPIS, is bounded. OPIS ensures a bounded solution by calculating a bounded upwind cell value for the three-point stencil. The OPIS Gamma solution compares well with the bounded solution obtained on the orthogonal mesh with the Gamma scheme.

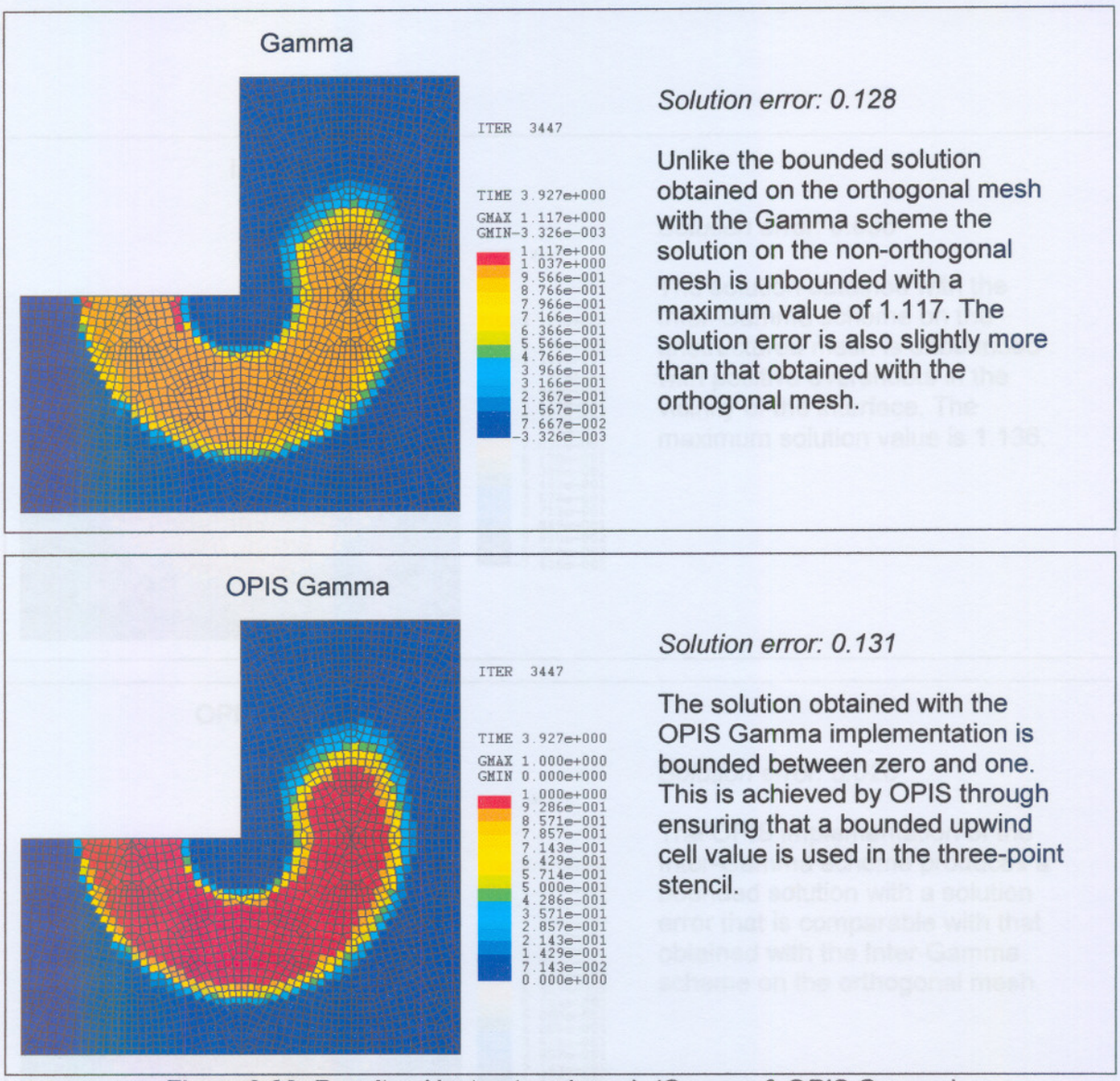
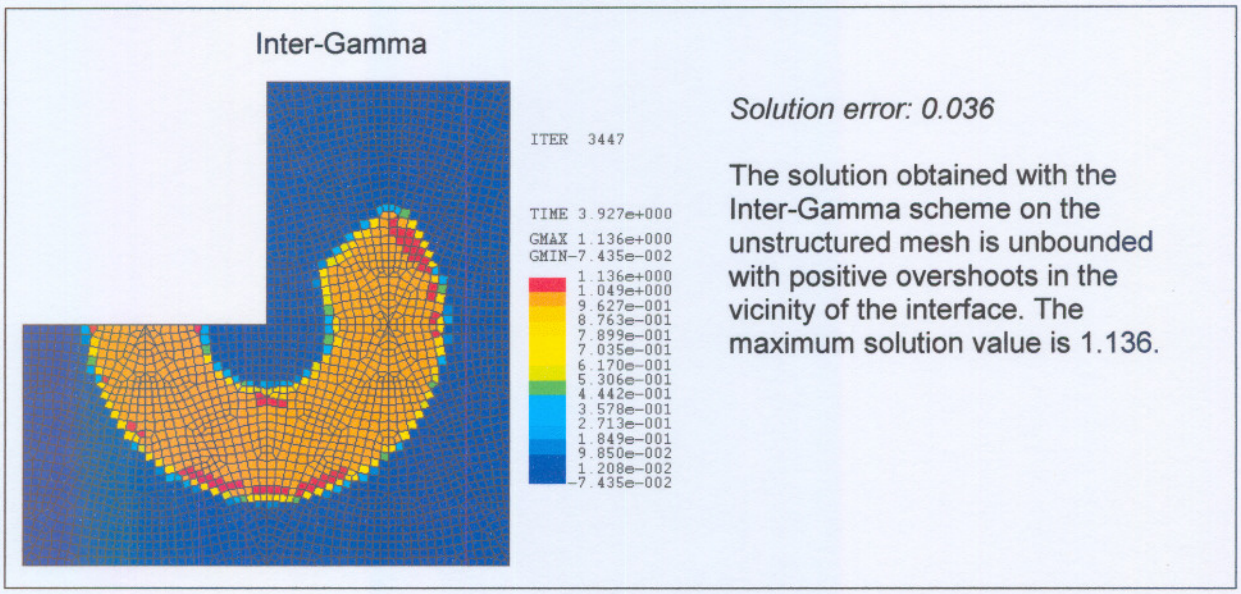


Figure 3.23: Results: Unstructured mesh (Gamma & OPIS Gamma)

Figure 3.24 shows the solutions obtained with the Inter-Gamma and OPIS Inter-Gamma schemes respectively. The solution obtained with the OPIS Inter-Gamma scheme is bounded. For the OPIS calculations the weighted upwind value and the original donor and acceptor values of the face were used in the calculation of the normalised donor cell value. OPIS ensures an exact upper bound of one for the convected profile with lower bounds consistently outperforming the orthogonal mesh results.

The results obtained with both the OPIS Gamma and OPIS Inter-Gamma schemes are promising; proving that volume weighted interpolation can effectively be used to construct the three-point stencils required for the implementation of high-resolution schemes.



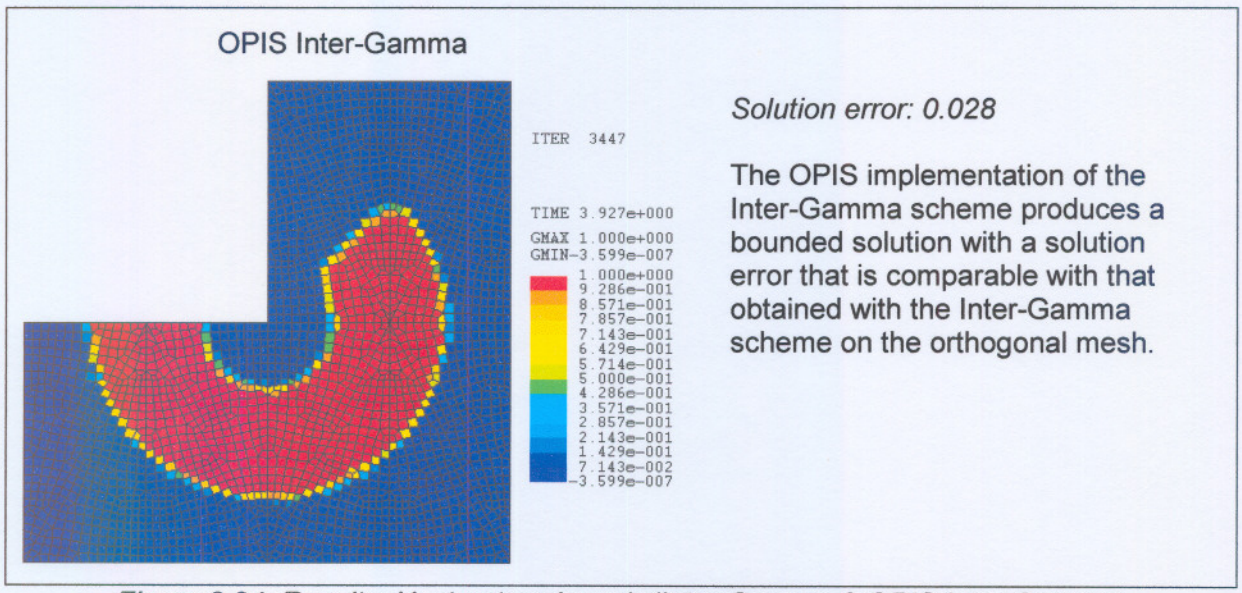


Figure 3.24: Results: Unstructured mesh (Inter-Gamma & OPIS Inter-Gamma)

3.7.3 Case 2: Shear flow

The shear flow test case was developed by Rider and Kothe (1995) to address the incomplete assessment of the integrity of volume tracking methods that can be made from simple translation and rotation tests as a result of the lack of topology change in the solution. In test case one the velocity field does not change the shape of the block wave profile during its movement across the mesh. In realistic problems the situation is more complicated, involving the stretching, shearing, break-up and merging of fluid regions (Rudman, 1997), yet even with restrictive constraints on the velocity field the accurate convection of step functions on an Eulerian mesh remains a difficult task. In this test case the presence of fluid shear is introduced into the velocity field.

A structured orthogonal and arbitrary unstructured mesh is used for the shear flow calculation. The structured mesh is identical to the mesh used by Rudman (1997). Both meshes have dimensions $x, z \in [0, \pi]$. The initial fluid distributions on both meshes is a circular shape with radius 0.2π and origin at $(0.5\pi, \pi - 0.2(1+\pi))$.

The structured mesh is divided into 100 x 100 cells in the x and z coordinate directions respectively giving a total cell count of 10000. The arbitrary unstructured mesh consists of 10143 cells with minimum, maximum and average cell sizes of 2.843e-5 m^3, 2.014e-4 m^3 and 9.730e-5 m^3 respectively. The two meshes together with the initial fluid distributions are shown in Figure 3.25 where the volume fraction field values are set to one inside and zero outside the circle.

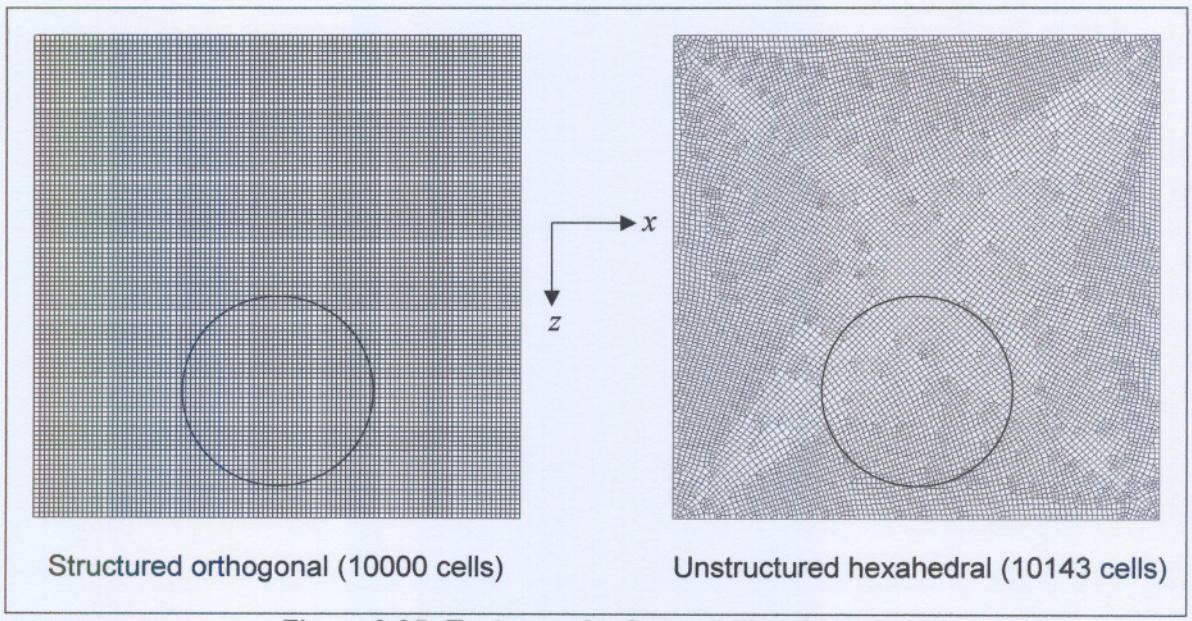


Figure 3.25: Test case 2 - Computational meshes

The cell volume distribution between the minimum and maximum cell volumes of the unstructured mesh are shown in Figure 3.26.

The velocity field for this test case is specified as:

$$\begin{aligned} u &= -\sin(x)\cos(z) \\ v &= 0 \\ w &= \cos(x)\sin(z) \end{aligned} \tag{3.50}$$

The simulation is performed for a number of time steps, N, where after the velocity field is reversed for the same number of time steps to return the fluid distribution to its original position as shown in Figure 3.25. Four cases are considered namely $N = 250, N = 500, N = 1000$ and

$N = 2000$. The results are compared with that of Rudman (1997). To ensure that the solution is at the correct position after a specified number of time steps, the structured mesh was used to calculate a time step size corresponding to a Courant number of 0.25. This time step size is 0.007855 seconds. The same time step size was used for the simulations on the unstructured mesh resulting in a higher maximum Courant number on this mesh.

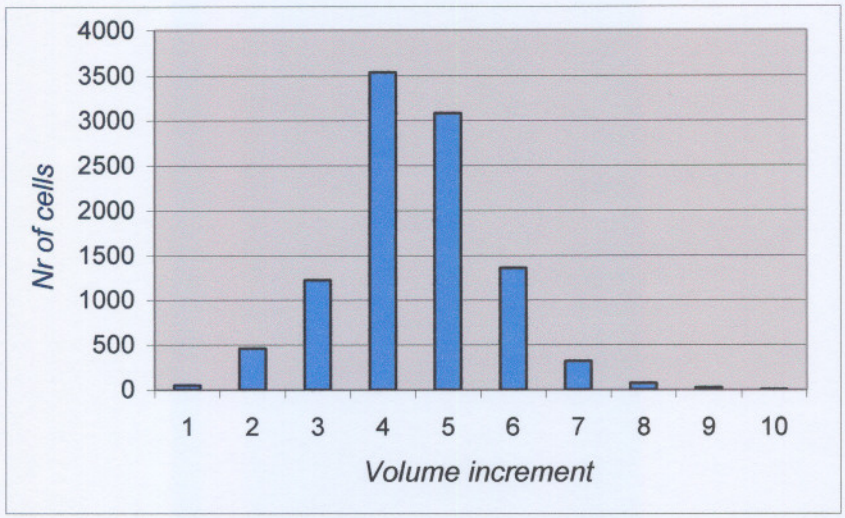


Figure 3.26: Cell volume distribution for unstructured mesh

The solution error is defined as:

$$E = \frac{\sum_{i}^{all\ cells} \left\| \alpha_i^n V_i - \alpha_i^a V_i \right\|}{\sum_{i}^{all\ cells} \alpha_i^o V_i} \tag{3.51}$$

where α^n is the solution after n time steps, α^a the analytical solution and α^o the original condition. V_i is the volume of cell i. The perfect convection scheme will result in a zero error computed with Eq. (3.51) as the fluid distribution returns to its original position after reversal of the velocity field.

The following figures show the results obtained with the Inter-Gamma scheme on the structured orthogonal and arbitrary unstructured mesh after 2000 time steps with a flow reversal after 1000 time steps. The initial fluid distribution, the distribution before the flow reversal and the final distribution are shown in each figure.

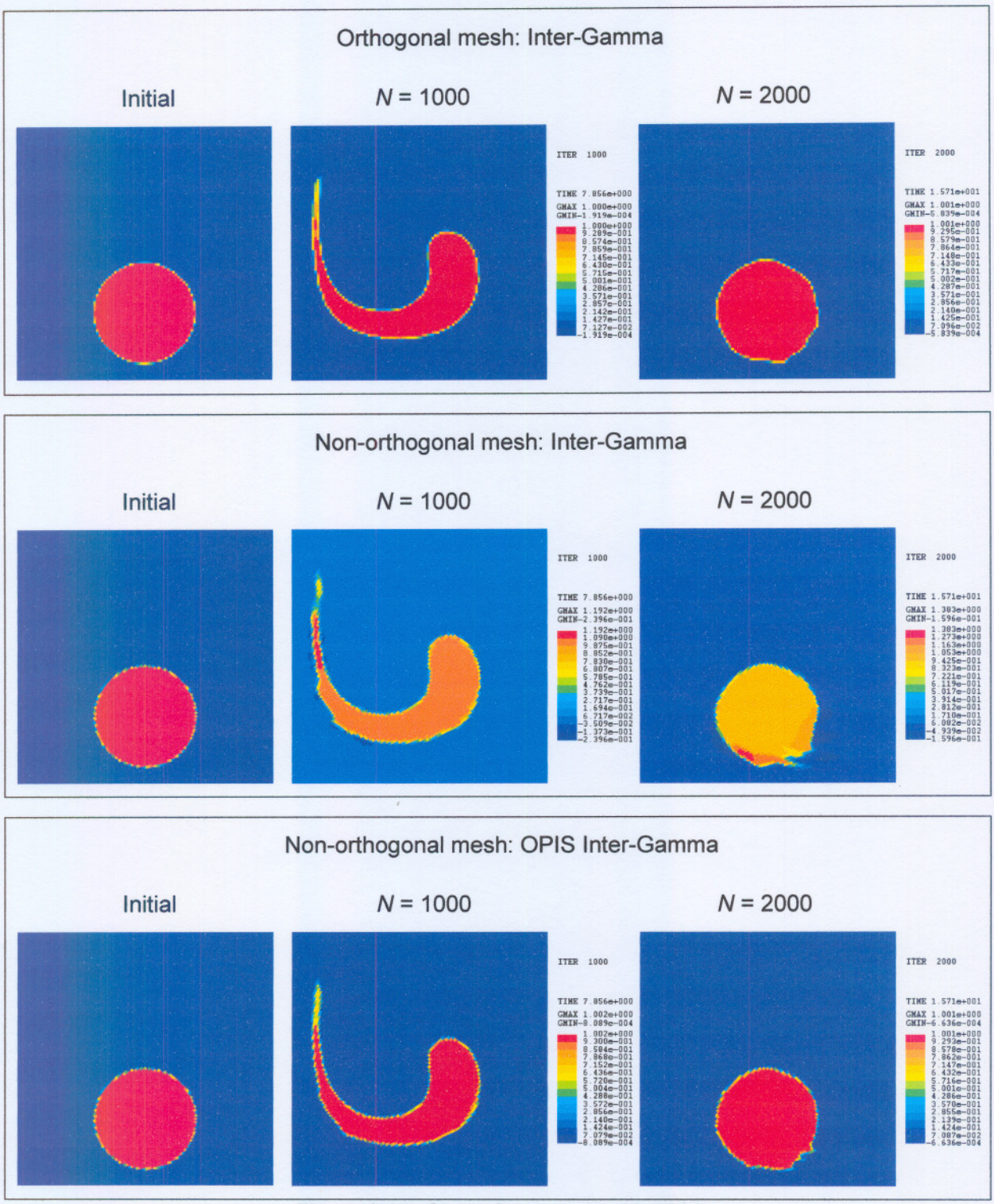


Figure 3.27: Shear flow results: N=1000

The following figures show the results obtained with the Inter-Gamma scheme on the structured orthogonal and arbitrary unstructured mesh after 4000 time steps with a flow reversal after 2000 time steps.

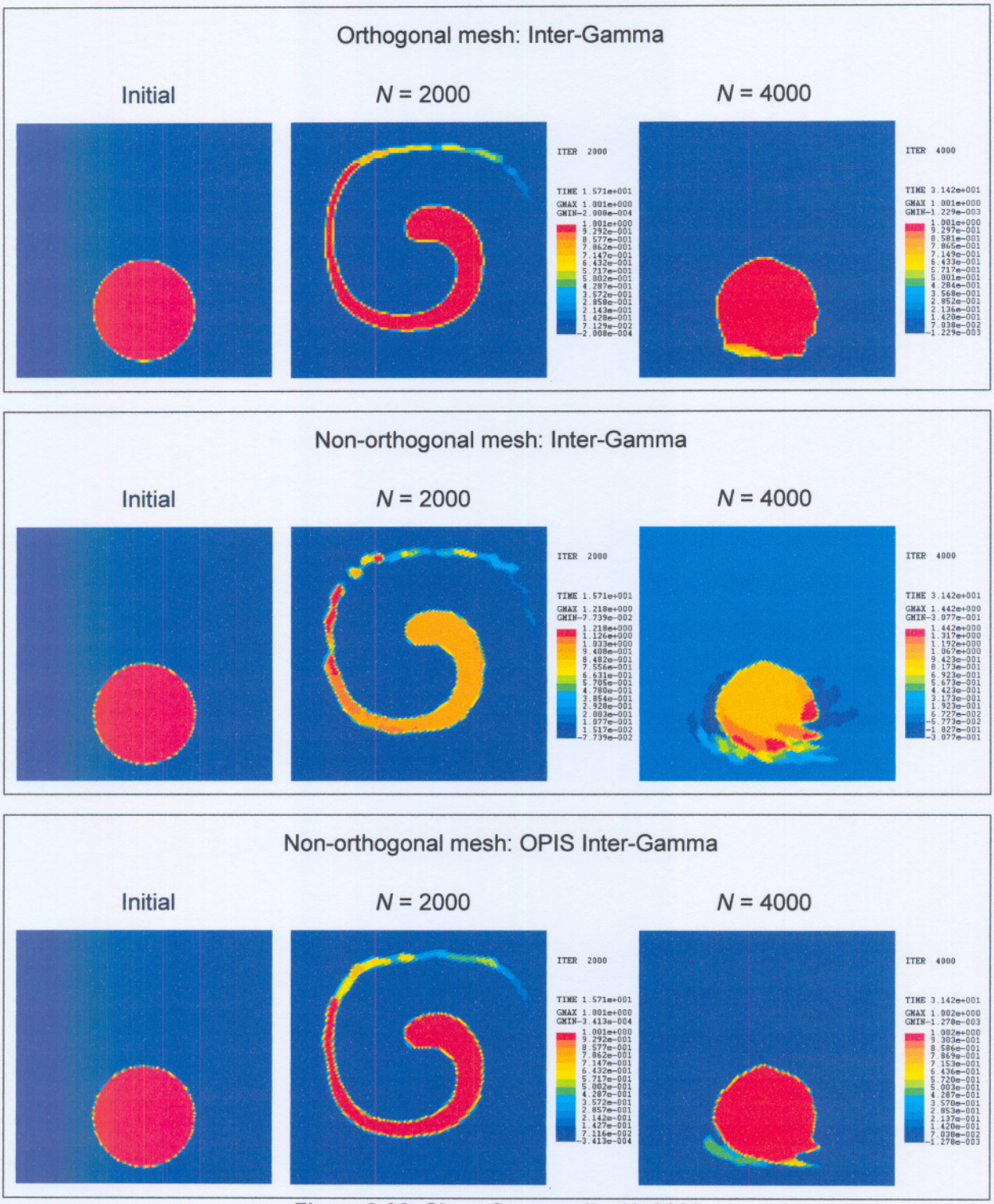


Figure 3.28: Shear flow results: N=2000

Figure 3.29 and Figure 3.30 show the results obtained by Rudman (1997) for the SLIC, Hirt-Nichols, FCT-VOF and Youngs schemes for $N = 1000$ and $N = 2000$ respectively. All the simulations by Rudman were performed on a 100 x 100 orthogonal mesh.

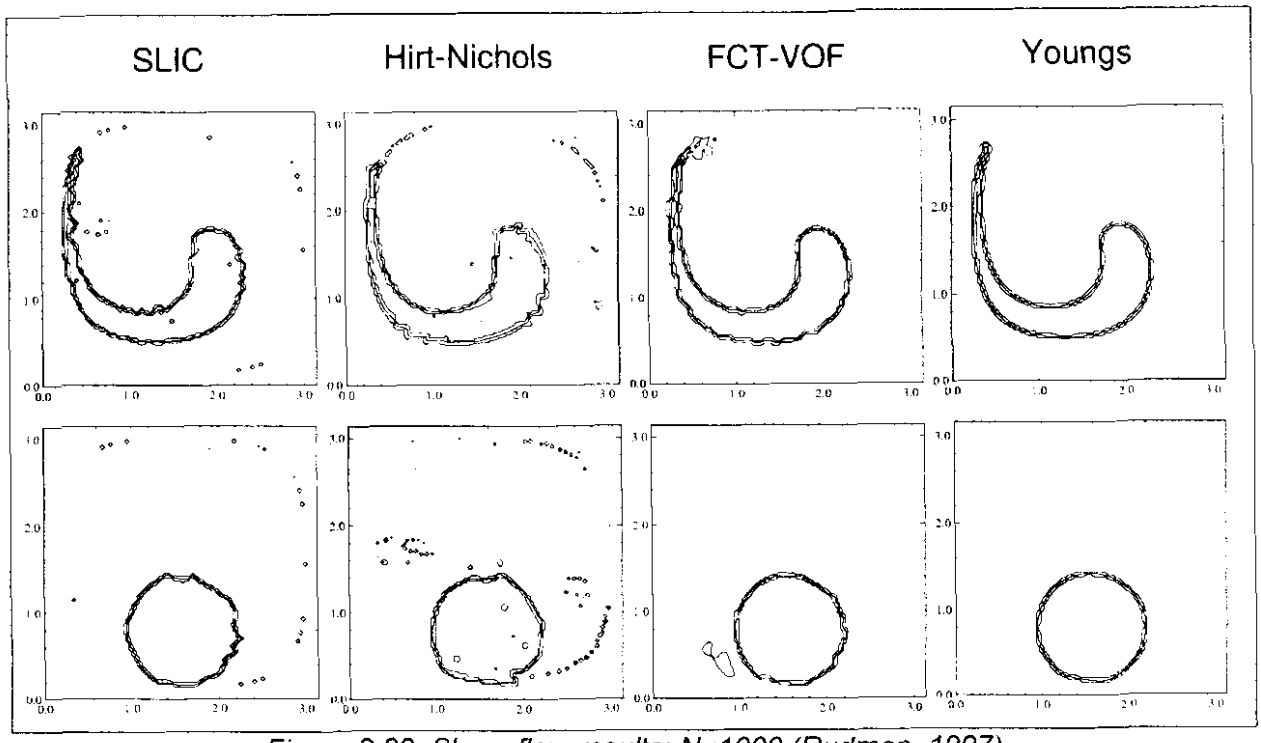


Figure 3.29: Shear flow results: N=1000 (Rudman, 1997)

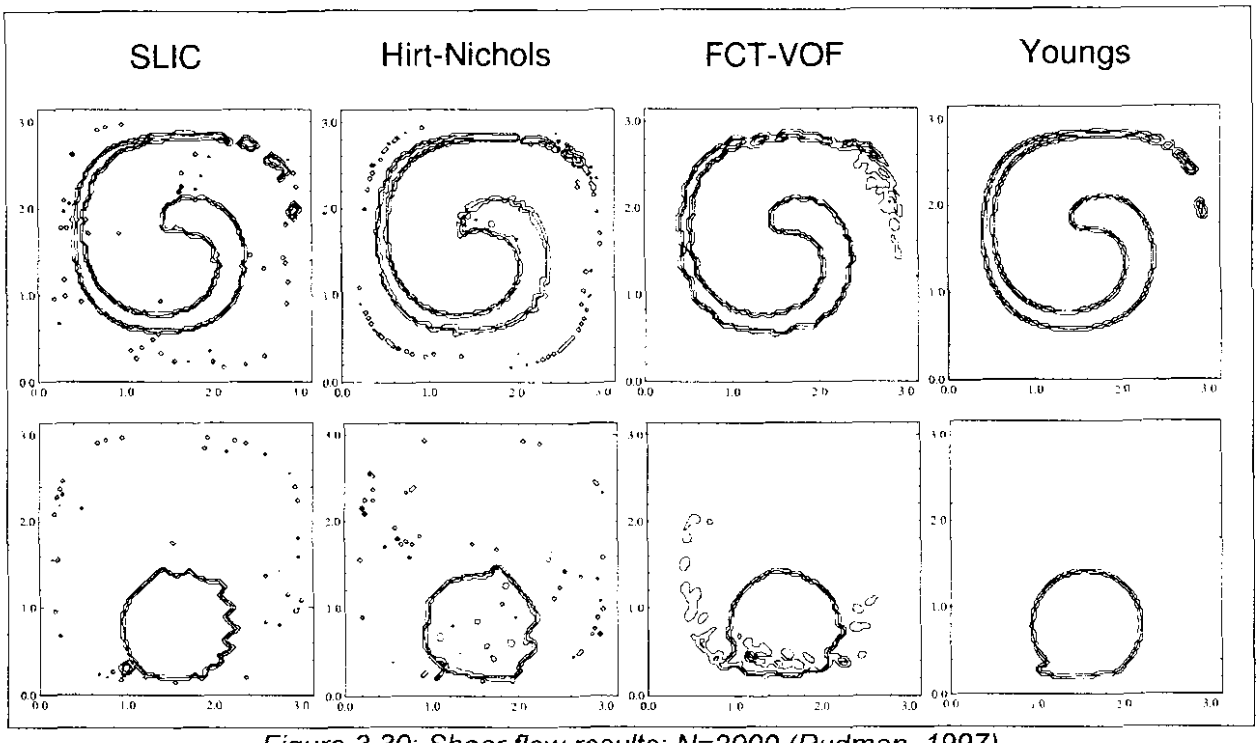


Figure 3.30: Shear flow results: N=2000 (Rudman, 1997)

Table 3.1 lists the solution errors obtained for the test runs performed with the Inter-Gamma and OPIS Inter-Gamma schemes as well as the errors obtained by Rudman (1997) on the structured mesh for the SLIC, Hirt-Nichols, FCT-VOF and Youngs schemes.

Scheme	$N = 250$	$N = 500$	$N = 1000$	$N = 2000$
Inter-Gamma (orthogonal)	2.14×10^{-2}	2.74×10^{-2}	3.05×10^{-2}	6.69×10^{-2}
Inter-Gamma (unstructured)	2.78×10^{-2}	4.00×10^{-2}	9.01×10^{-2}	2.76×10^{-1}
OPIS Inter-Gamma (unstructured)	2.08×10^{-2}	2.78×10^{-2}	3.91×10^{-2}	1.20×10^{-1}
SLIC	2.72×10^{-2}	3.30×10^{-2}	4.59×10^{-2}	9.02×10^{-2}
Hirt-Nichols	3.24×10^{-2}	4.00×10^{-2}	6.66×10^{-2}	1.09×10^{-1}
FCT-VOF	1.94×10^{-2}	2.35×10^{-2}	3.14×10^{-2}	1.44×10^{-1}
Youngs	2.61×10^{-3}	5.12×10^{-3}	8.60×10^{-3}	3.85×10^{-2}

Table 3.1: Solution errors for shear flow calculation

The Simple Line Interface Calculation (SLIC) method of Noh and Woodward (1976) and the method of Youngs (1982) form part of the family of *line techniques* (Ubbink, 1997). These methods were developed for multi-fluid flow simulations. The SLIC method approximates the interface in each cell with a line parallel to one of the coordinate axis and assumes different fluid distributions for movement in the different coordinate directions. The fluid distribution in a cell is determined using the volume fraction distribution of the neighbouring cells. The method of Youngs (1982) is a refinement of the SLIC method and uses oblique lines to approximate the interface in each cell. Both methods are therefore based on the principle of geometric reconstruction, using the local flow velocities to move the reconstructed fluid distribution in each cell in a Lagrangian manner (Ubbink, 1997). These methods are implemented on meshes consisting of orthogonal cells and the geometric reconstruction of the fluid distributions in each cell is a complex exercise. Extending these methods for arbitrary unstructured meshes will be a daunting task with major computational overheads.

The donor-acceptor method of Hirt and Nichols (1981) uses the acceptor cell volume fraction value to predict the volume fraction transported through the face between the donor and acceptor cells. Downwind differencing compresses the interface as seen from earlier examples using the Inter-Gamma scheme which has a downwind differencing component. Without careful control this approach will lead to unbounded volume fraction values. Boundedness is maintained by including information on the availability of fluid in the donor cell into the formulation. The method of Hirt and Nichols (1981) uses a nine cell stencil to calculate an

approximate interface normal vector. The interface is then approximated by a horizontal or vertical line based on the largest component of the interface normal. Fluxes in a direction parallel to the interface is calculated using the upwind differencing scheme while fluxes in a direction perpendicular to the interface is calculated using the donor-acceptor method. The FCT-VOF method (Rudman, 1997) is an improved version of the donor-acceptor method that applies a combination of the upwind- and downwind differencing schemes to eliminate the diffusiveness of the upwind differencing scheme as well as the instability of the downwind differencing scheme. All the simulations performed by Rudman (1997) apply structured orthogonal meshes with no indication that these schemes can be used on arbitrary unstructured meshes.

The solution errors of the Inter-Gamma scheme on the structured orthogonal mesh compares well with the other schemes listed in Table 3.1 except for the Youngs scheme which produces errors that are significantly smaller than the other schemes. While the solution errors for the Inter-Gamma scheme on the unstructured mesh differ significantly from the orthogonal mesh, there is a marked improvement in the solution when using the OPIS implementation of the Inter-Gamma scheme. The solution errors of the orthogonal mesh and the OPIS implementation on the unstructured mesh are in good agreement except for the $N = 2000$ case where the OPIS simulation breaks down. Even for this extreme case on the unstructured mesh, the OPIS implementation produces a solution error that is less than half that of the standard Inter-Gamma implementation using Eq. (3.41).

The improvement brought about by using OPIS is clearly seen from the results shown in Figure 3.27 and Figure 3.28. In Figure 3.28 the solution from the standard implementation of the Inter-Gamma scheme display sections where the tail breaks up into separate sections. In the OPIS Inter-Gamma simulation the tail stays intact and the solution compares well with that obtained on the structured orthogonal mesh. The OPIS Inter-Gamma solution is a significant improvement over the solutions obtained with the SLIC, Hirt-Nichols and FCT-VOF schemes on the orthogonal mesh. This is noticeable when comparing Figure 3.27 and Figure 3.28 with the results from Rudman (1997) shown in Figure 3.29 and Figure 3.30. The results from the SLIC, Hirt-Nichols and FCT-VOF show unacceptable levels of flotsam with severe distortion of the final fluid distributions. The OPIS Inter-Gamma solutions are free from any flotsam. Visually the results for the Inter-Gamma scheme on the structured mesh compares well with the results from the Youngs scheme.

3.8 Closure

The finite volume discretisation of the general transport equation for a scalar variable was presented in this chapter. The requirement for face value interpolation follows directly from the discretisation of the convection term of the transport equation. Since variables are stored and solved for at cell centres during a simulation, face values that are required in the discretised transport equations must be interpolated from cell values. Convection schemes are used for this purpose to approximate the variation of flow properties between cell centres in such a way that convection is accurately modelled.

Examples were presented to show the significant influence of different convection schemes on the convection of a complex block wave profile in one dimensional flow. It was shown that schemes such as upwind differencing produces bounded results but at the cost of accuracy, introducing severe levels of numerical diffusion into the solution. Other schemes such as central differencing generate results with improved accuracy, but causes numerical dispersion resulting in unphysical oscillations in the solution. The use of three-point stencils for convection modelling was described and applied to the QUICK scheme which produces more accurate results than central and upwind differencing.

The implementation of high-resolution schemes based on the normalised variable diagram was described and the issues involved in implementing these schemes on arbitrary unstructured meshes investigated. The concept of volume weighted interpolation was introduced as an alternative method to implement schemes based on the normalised variable diagram on arbitrary unstructured meshes. Volume weighted interpolation allows the use of convection schemes well suited for orthogonal meshes on arbitrary unstructured meshes. The OPIS interpolation stencil can be used with any convection scheme that requires a three-point stencil.

The advantages of volume weighted interpolation for arbitrary unstructured meshes were demonstrated by means of test cases involving rotational and shear flow. The volume weighted interpolation methodology is fully three-dimensional despite the fact that all the example calculations were performed on two dimensional meshes. In the following chapter the uses of volume weighted interpolation within the finite volume method is extended to include pre- and post-processing tasks.

4. Volume weighted interpolation for pre- and post-processing

4.1 Introduction

The use of volume weighted interpolation for the modelling of convective transport was described in Chapter 3. In this chapter additional uses of volume weighted interpolation are considered. Not only does volume weighted interpolation present a means to construct interpolation stencils for convection schemes, it is also a useful tool for pre- and post-processing. Some of the techniques described in this chapter were used in the case studies presented in Chapter 3. In some of the examples in this chapter, the dependent variable is the volume fraction used in two phase flow simulations to identify different fluids or fluid phases (Ubbink, 1997).

4.2 Pre-processing

The initialisation of scalar volume fraction fields is a problem that is often encountered during two-phase flow simulations.

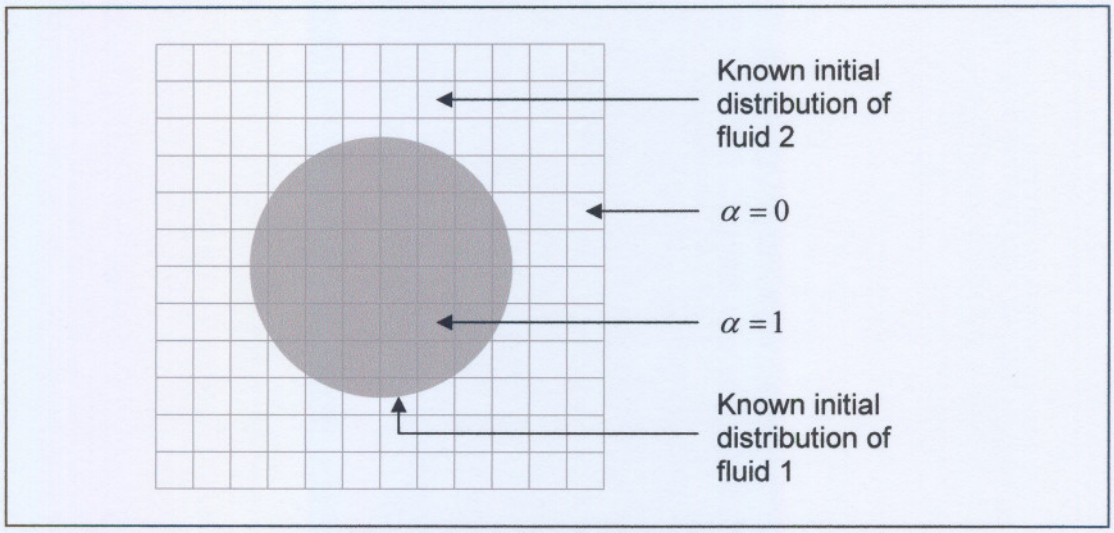


Figure 4.1: Initial fluid distribution

The initial location of the two fluids must be accurately specified, for example the location of a horizontal liquid gas interface or the location of a liquid droplet within a gas. This process requires the user to initialise a scalar volume fraction value for each cell in the mesh. When using orthogonal meshes and simple horizontal interfaces, this task is trivial, however the mesh is often complex and unstructured with complex fluid distributions. In order to initialise a fluid distribution on a mesh, the volume fractions of the cells that are *partially filled*, i.e. those cells

located at the interface between the two fluids or fluid phases must be accurately specified. Figure 4.1 shows an example of a circular fluid distribution on an orthogonal mesh which represents the initial location of one of the fluid phases, in this case fluid one.

One method to approximate the volume fraction field is to consider the locations of the cell centres in relation to the circular fluid distribution shown in Figure 4.1. When the cell centre is located within the circular distribution region of fluid one, its value is set equal to one, otherwise the value is set to zero. The result is a field of ones and zeros as shown in Figure 4.2. This initialisation is not conservative. The initialised field will therefore not necessarily contain the same volume of fluid one as the initial *analytical* fluid distribution.

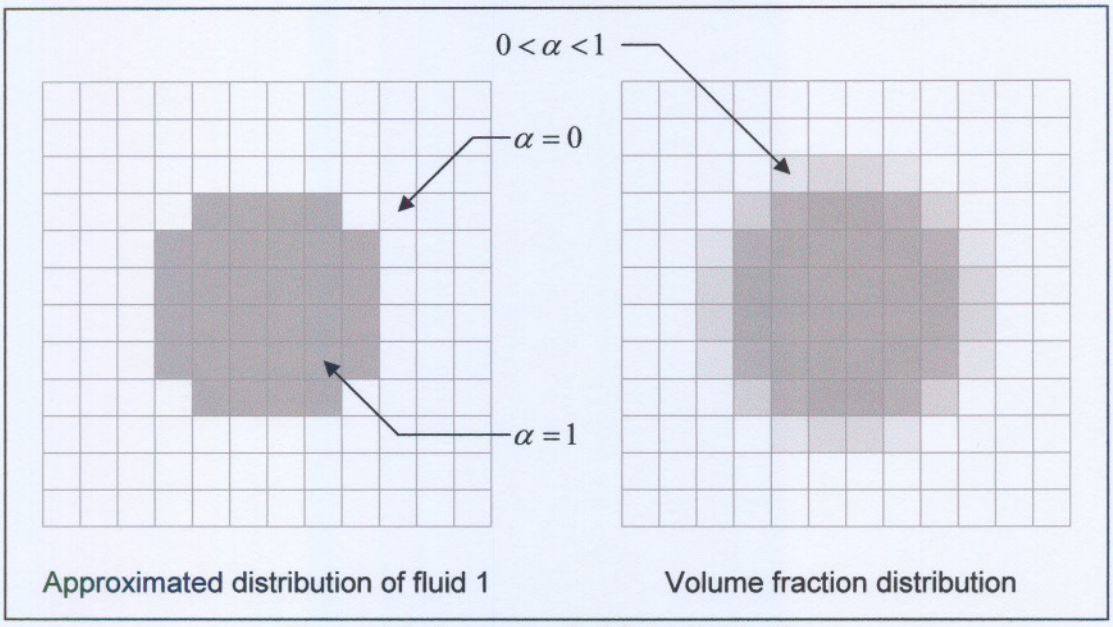


Figure 4.2: Approximated fluid distributions

A more appropriate initialisation method is to calculate the volume fractions of the cells located in the vicinity of the interface between the two fluids or fluid phases using the available geometric information of the computational mesh and the initial fluid distribution. In doing so the correct volume fractions are calculated for cells containing a mixture of fluid one and two. Figure 4.2 shows how the interface between fluid one and two is approximated in accordance with the volume fractions values of fluid one in cells containing both fluids. This method of initialisation is considered to be the most appropriate since it is both conservative, i.e. no loss of fluid and consistent with the finite volume method where the cell value represents the average value of the variable for that cell. A problem with this approach is the difficulties associated with the calculation of the volume fractions in the vicinity of the interface.

Volume weighted interpolation addresses this problem effectively. Apart from the computational mesh generated for the flow solution, a second mesh is created which represents the shape of

the initial fluid distribution. The boundaries of this second mesh, referred to as mesh A therefore coincides with the boundaries of the fluid distribution of fluid one. A conservative volumetric mapping is performed from mesh A to the mesh used for the flow simulation, referred to as mesh B. The interpolated cell values of mesh B then equals the overlapped volumes of mesh A onto mesh B; therefore, if a cell in mesh B is not overlapped by mesh A, the cell value will be zero. When a cell in mesh B is completely overlapped by mesh A, the cell value will be one. A partial overlap of a cell in mesh B with mesh A results in a cell value between zero and one, consistent with the volume fraction overlapped between the cell and mesh A.

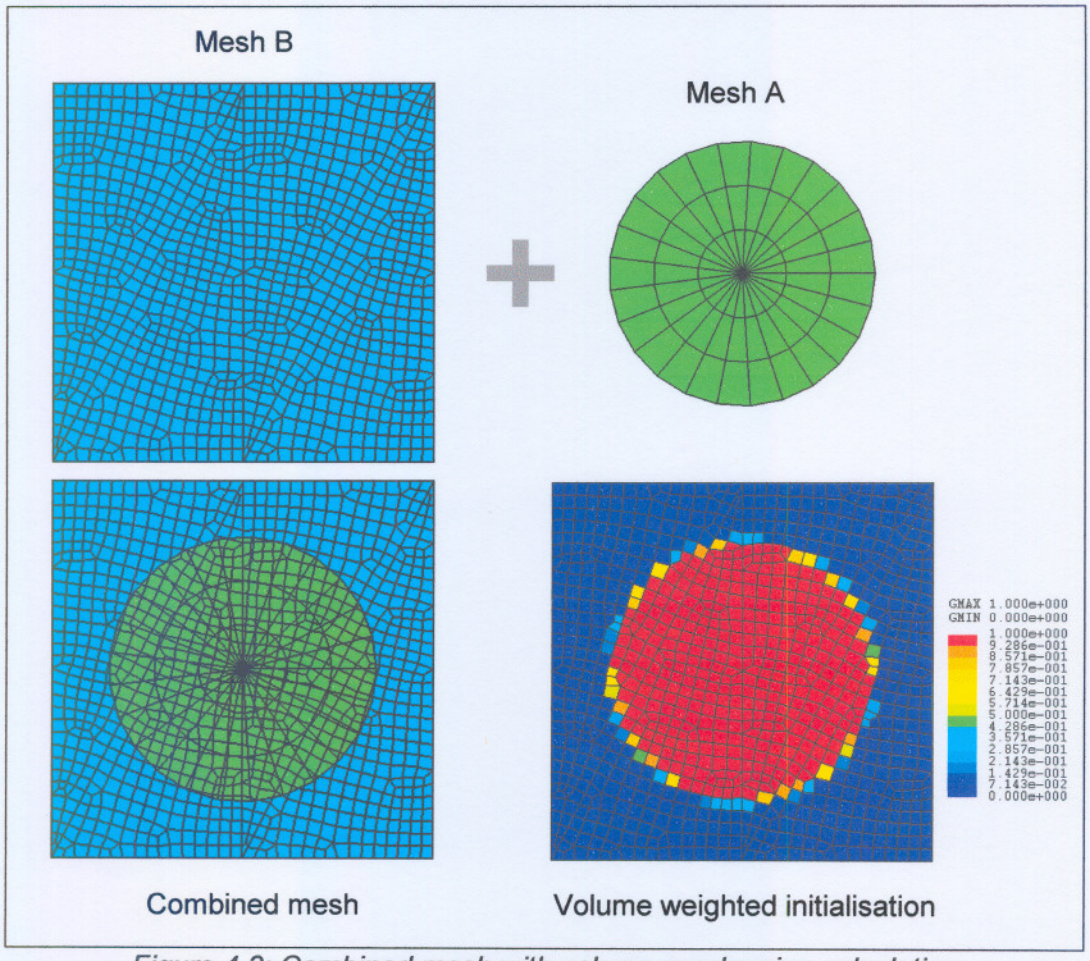


Figure 4.3: Combined mesh with volume overlapping calculation

As an example a circular droplet is mapped onto an unstructured hexahedral mesh. The geometry of the droplet is defined by creating mesh A in the shape of the droplet. Mesh A may have an arbitrary shape to accommodate the initial distribution of fluid one. The result of the volume overlapping calculation is the field of volume fraction values shown in Figure 4.3. A calibrated colour scale indicating the volume fraction values between zero and one is also shown in the figure.

The properties of convection schemes are often tested using scalar convection test cases (Rider and Kothe, 1998), (Rudman, 1998). In many of these cases the analytic solution is known because of the simple nature of the problem. Volume weighted interpolation is therefore an ideal method to map known analytical fluid distributions onto computational meshes in order to compare simulation results with analytical solutions. Figure 4.4 and Figure 4.5 show additional examples of volume fraction initialisations on structured orthogonal and unstructured meshes. In each example mesh A is superimposed on to mesh B, showing the overlapping regions.

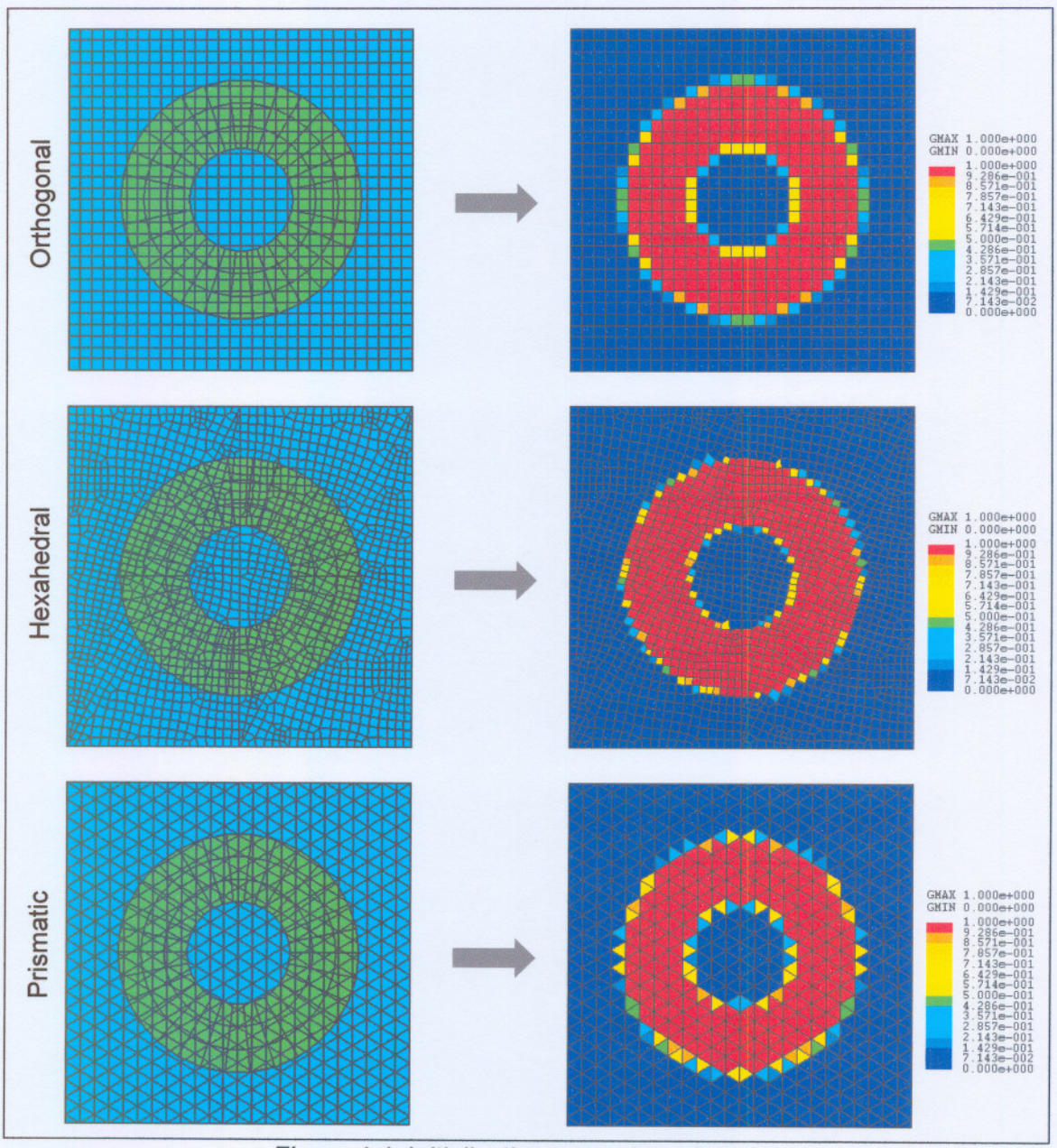


Figure 4.4: Initialisation examples - Annulus

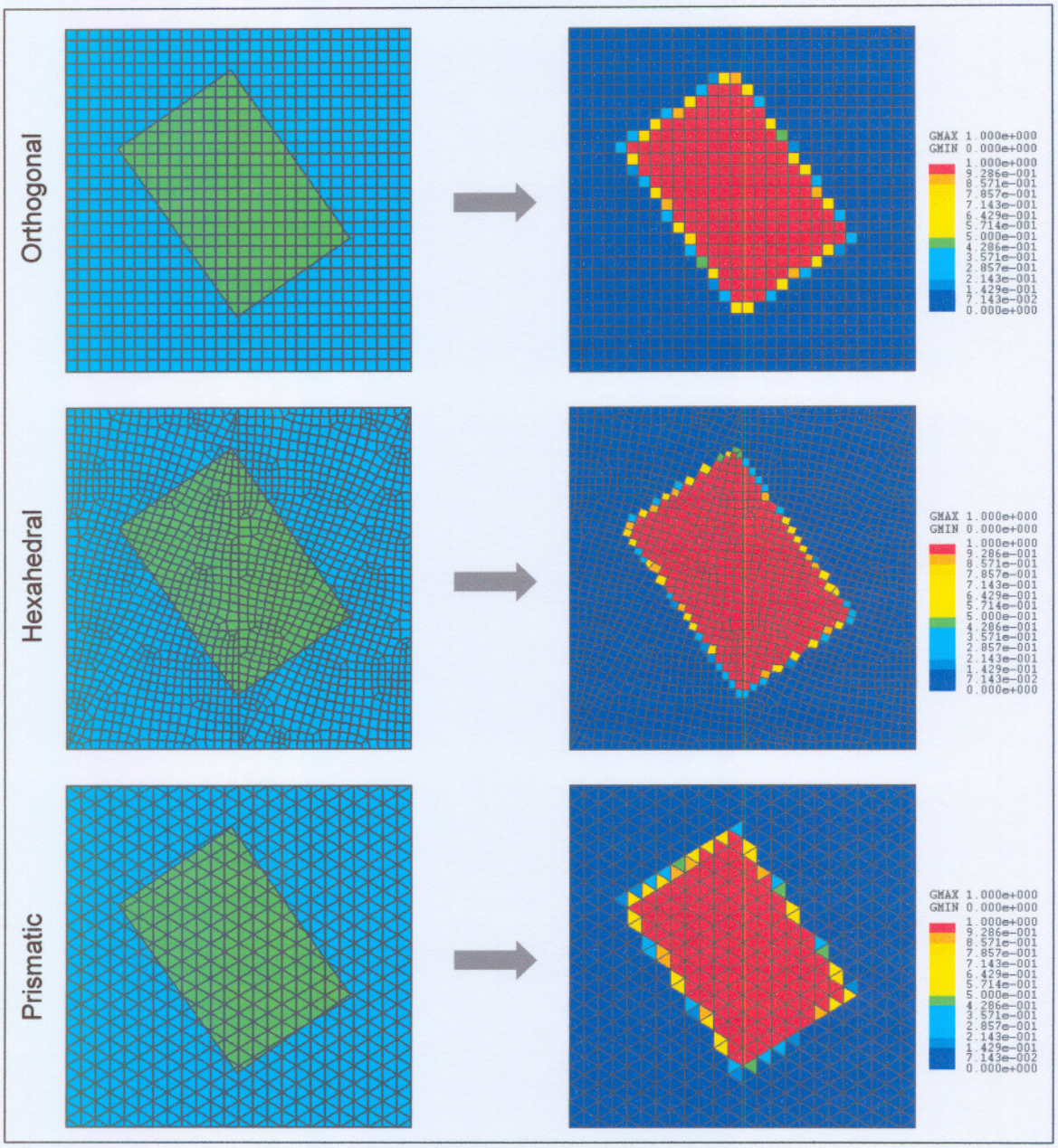


Figure 4.5: Initialisation examples - Rectangle

Any shape constructed using mesh A can be mapped onto mesh B, thereby greatly simplifying the initialisation of complex volume fraction fields for structured and unstructured meshes. Although the examples shown are volume fraction initialisations, other variables, for example momentum, can also be mapped conservatively from one mesh to another through volume weighted interpolation. Volume weighted interpolation to a great extent simplifies the process of mapping variables onto complex meshes.

4.3 Post-processing

4.3.1 Mapping solutions onto different meshes

Volume weighted interpolation can also be utilised as a post-processing tool. It is often required to produce graphs of simulation results along straight lines across a mesh. It is a simple task to create graphs of results along mesh lines when modelling problems on structured orthogonal meshes where cell centres are evenly spaced. This task becomes more challenging when unstructured meshes are used since the cell centres are not generally located in straight lines or evenly spaced. Volume weighted interpolation can be used for the post-processing of results on structured and unstructured meshes. With a solution available on a structured or unstructured mesh, a conservative mapping onto another mesh, which is more suitable for post-processing purposes, can be performed.

There are no requirements for the mesh onto which the results are mapped with regard to the cell size and location, but as a rule of thumb it is recommended that this mesh consist of cells of approximately the same size as the cells of the mesh that was used in the calculation. The use of cells that are too large will cause the mapped solution to smear across the mesh while cells that are too small will result in stepped mapping results. The volume weighted interpolation algorithm is robust and guarantees a conservative mapping between the meshes. Two examples are presented to illustrate this concept for post-processing.

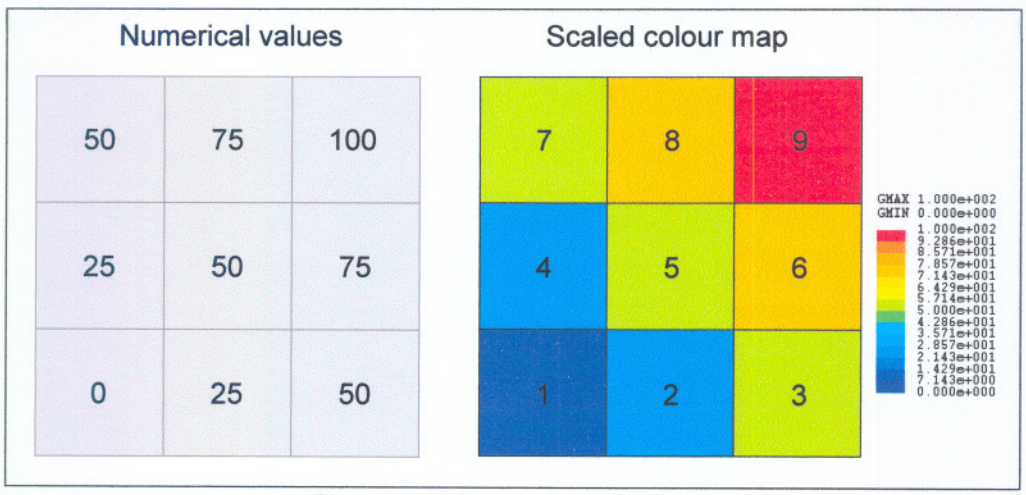


Figure 4.6: Example 1- Solution

Figure 4.6 shows a mesh consisting of nine cells. The cell values are initialised with the numerical values as shown. These values could for example represent a temperature distribution. The numerical values represent the *cell values* per unit volume. The representative value for the mesh, ψ, is calculated by multiplying the cell value with the cell volume and adding together the values for all the cells. For this example the cell volume equals 1/9 with the

total mesh volume equal to one. The representative value for the mesh is 50 as shown in the following calculation.

$$\psi = \frac{0+25+25+50+50+50+75+75+100}{9} \qquad (4.1)$$
$$= 50$$

The results were mapped onto a 2 x 2 and 4 x 4 mesh using volume weighted interpolation. The results are shown in Figure 4.7 with variable values coloured according to the calibrated colour scale for each mesh.

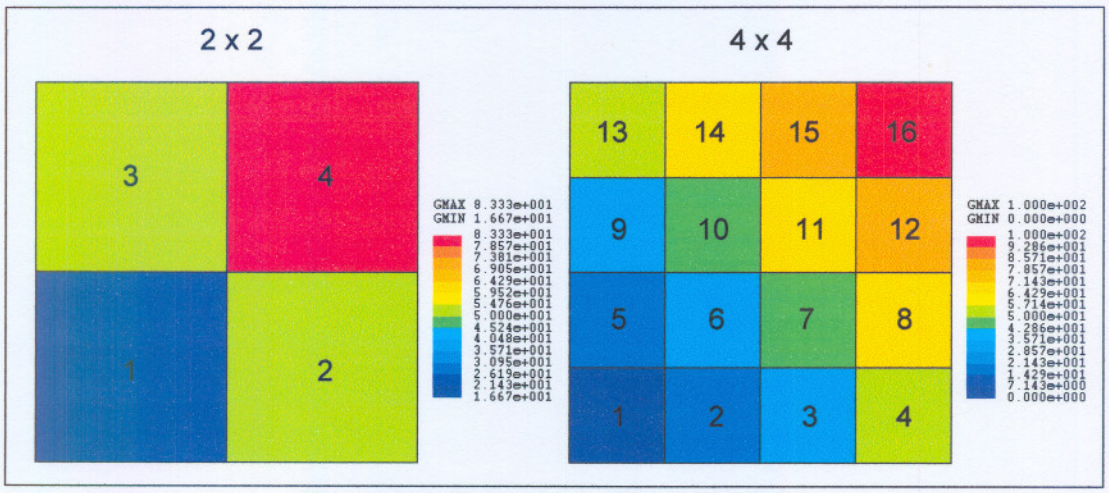


Figure 4.7: Example 1 - Mapped results

The interpolated results for the 2 x 2 mesh are as follows:

Cell nr	Mapped value
1	16.66666
2	49.99999
3	49.99999
4	83.33333

Table 4.1: Example 1 - 2 x 2 mesh

From Table 4.1 the representative mesh value can be calculated using Eq. (4.1). The cell volume equals 1/4.

$$\psi = \frac{16.66666+49.99999+49.99999+83.33333}{4} \qquad (4.2)$$
$$= 50$$

It should be noted that conservation is satisfied; however, since the 2 x 2 mesh used in the mapping is coarser than the original 3 x 3 mesh, the available information is *smeared* across the mesh. The results for the 4 x 4 mesh are as follows:

Cell nr	Mapped value	Cell nr	Mapped value
1	0.00000	9	33.33333
2	16.66666	10	49.99999
3	33.33333	11	66.66666
4	50.00000	12	83.33333
5	16.66666	13	50.00000
6	33.33333	14	66.66666
7	49.99999	15	83.33333
8	66.66666	16	100.00000

Table 4.2: Example 1 - 4 x 4 mesh

Again the representative mesh value is 50. Conservation is therefore satisfied. In this example the boundaries of the original and mapped meshes coincides. This is however not a requirement. Any two overlapping meshes may be used. It is therefore possible to create meshes in regions where results are required for post-processing purposes and interpolate to those meshes.

The second example considers the scalar convection test case that was described in Chapter 3. A block wave profile is convected in a rotational velocity field using a time dependent scalar convection equation without any physical diffusion. Convection is modelled using the Gamma differencing scheme. An arbitrary unstructured computational mesh was used in the calculation.

In the first part of this example the solution is interpolated from the hexahedral mesh onto progressively finer orthogonal meshes. Figure 4.8 shows the flow solution obtained on the unstructured hexahedral mesh as well as the interpolated results for the three orthogonal meshes. Map A consists of three blocks of 10 x 10 cells. Map B consists of three blocks of 20 x 20 cells and map C consists of three blocks of 40 x 40 cells. In all three examples the interpolation is conservative. One example of an application where this type of interpolation is useful is in the construction of three-dimensional contour maps of results using applications suitable for structured data sets. The results are interpolated from the mesh used in the simulation to the mesh or meshes suitable for post-processing purposes.

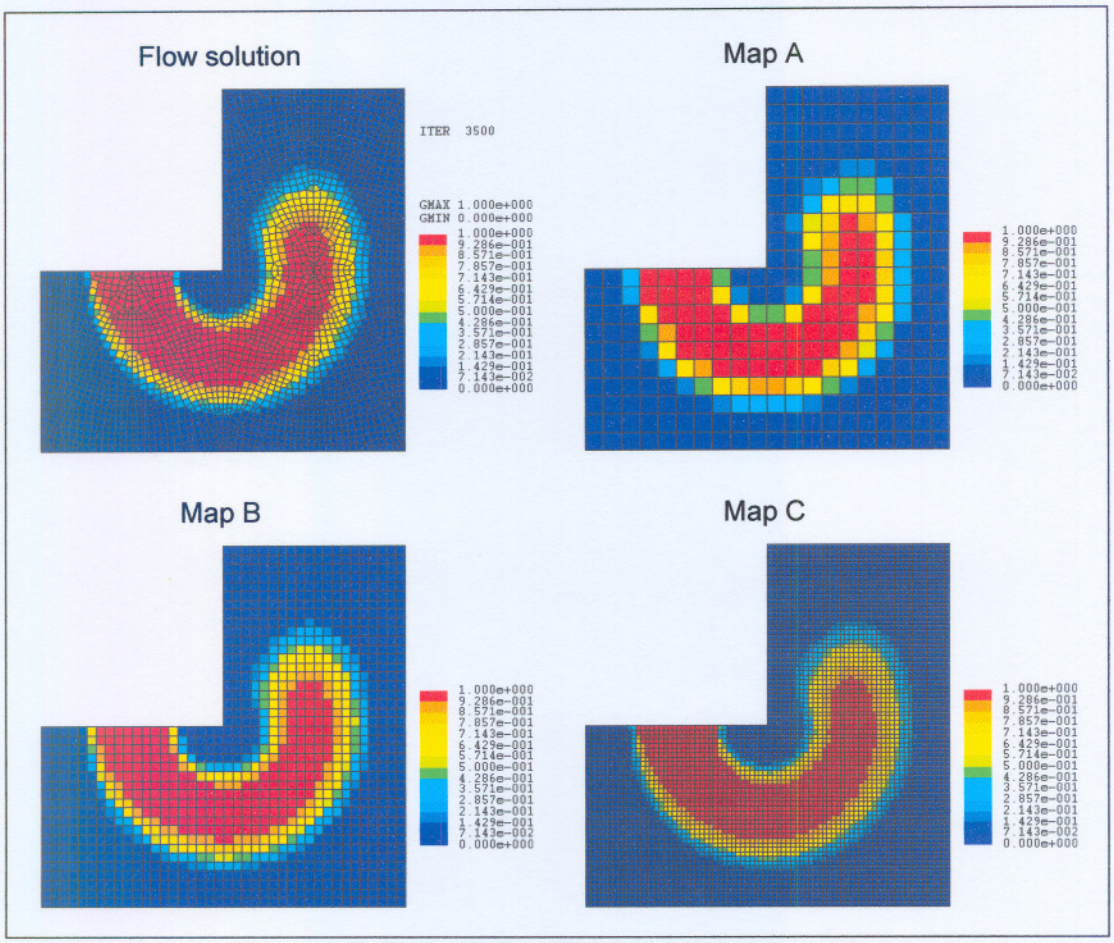


Figure 4.8: Example 2 - Orthogonal mesh mapping

In the second part of this example graphs are created along several radial stations across the mesh. This is achieved by constructing orthogonal meshes consisting of rows of orthogonal cells in the locations where the results should be graphed.

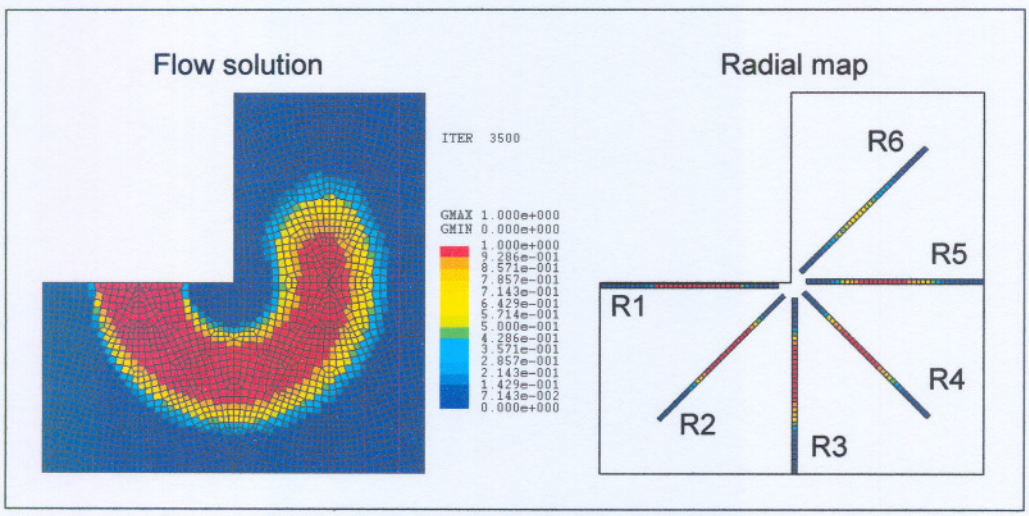


Figure 4.9: Example 2 - Radial map with 37 cell stencils

Figure 4.9 and Figure 4.10 show how these cell blocks are created along the radial directions. The results of the flow solution are interpolated to this radial mesh and the interpolated results used to construct graphs of the solution along the radial lines. For reference purposes the radial mesh blocks are numbered from R1 to R6. The mesh in Figure 4.9 consists of 37 cells along each radial cell block, while the mesh in Figure 4.10 consists of 19 cells along each radial cell block.

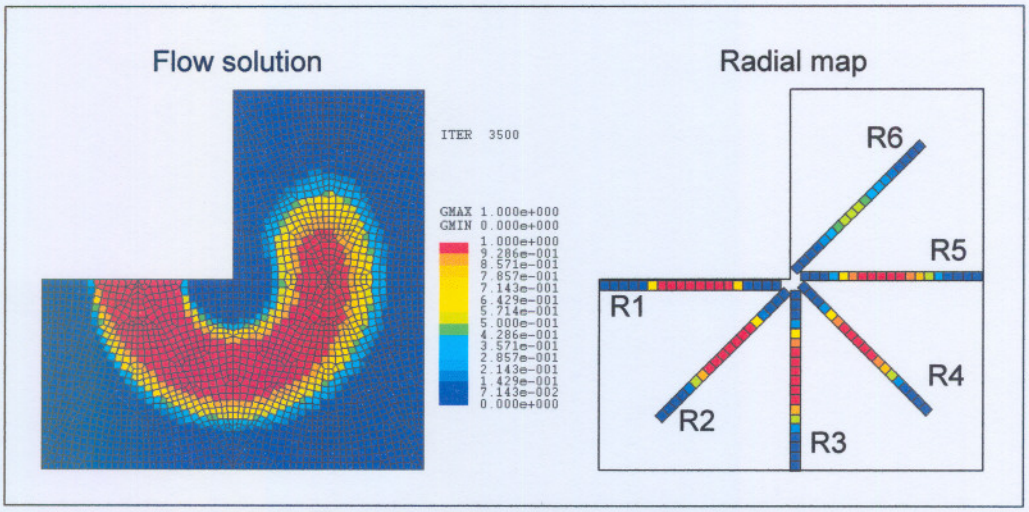


Figure 4.10: Example 2 - Radial map with 19 cell stencils

Figure 4.11 shows the graphs generated from the mapped results along the radial mesh lines. Graphs such as these effectively show trends in solutions along any chosen line or curve. Note how the radial mesh, consisting of 19 cells per radial cell block, smoothes the solution in comparison with the 37 cells per radial cell block, that consists of smaller cells.

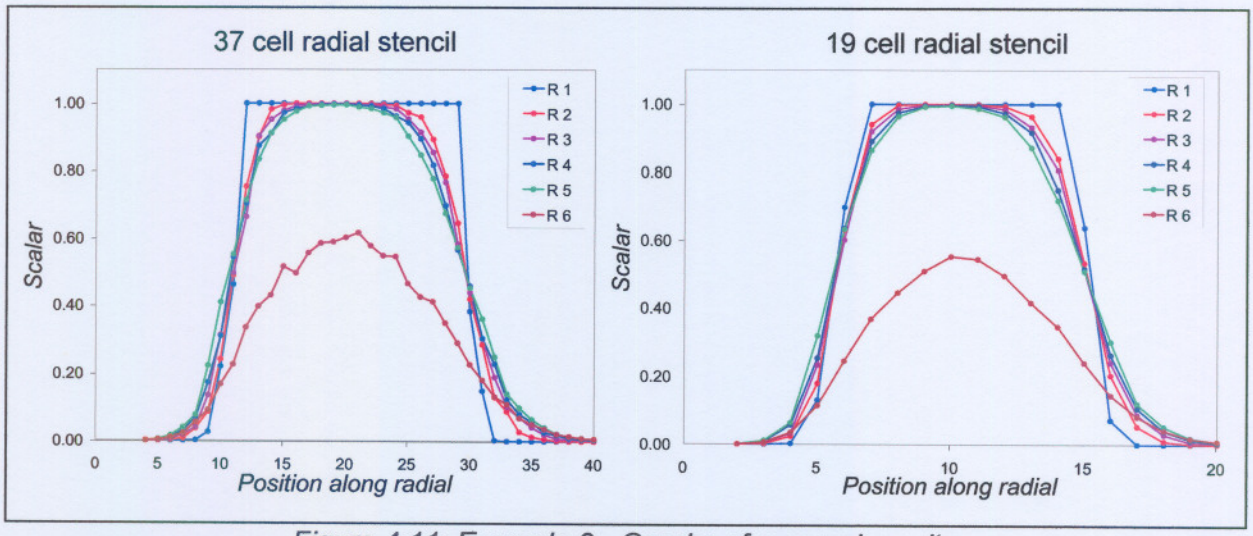


Figure 4.11: Example 2 - Graphs of mapped results

Volume weighted interpolation for post-processing addresses both the problems of unevenly spaced data points and cell centres that are not located in straight lines. Volume weighted

interpolation can also be used on Cartesian meshes when results are required along lines that are not aligned with the main coordinate directions, for example oblique lines.

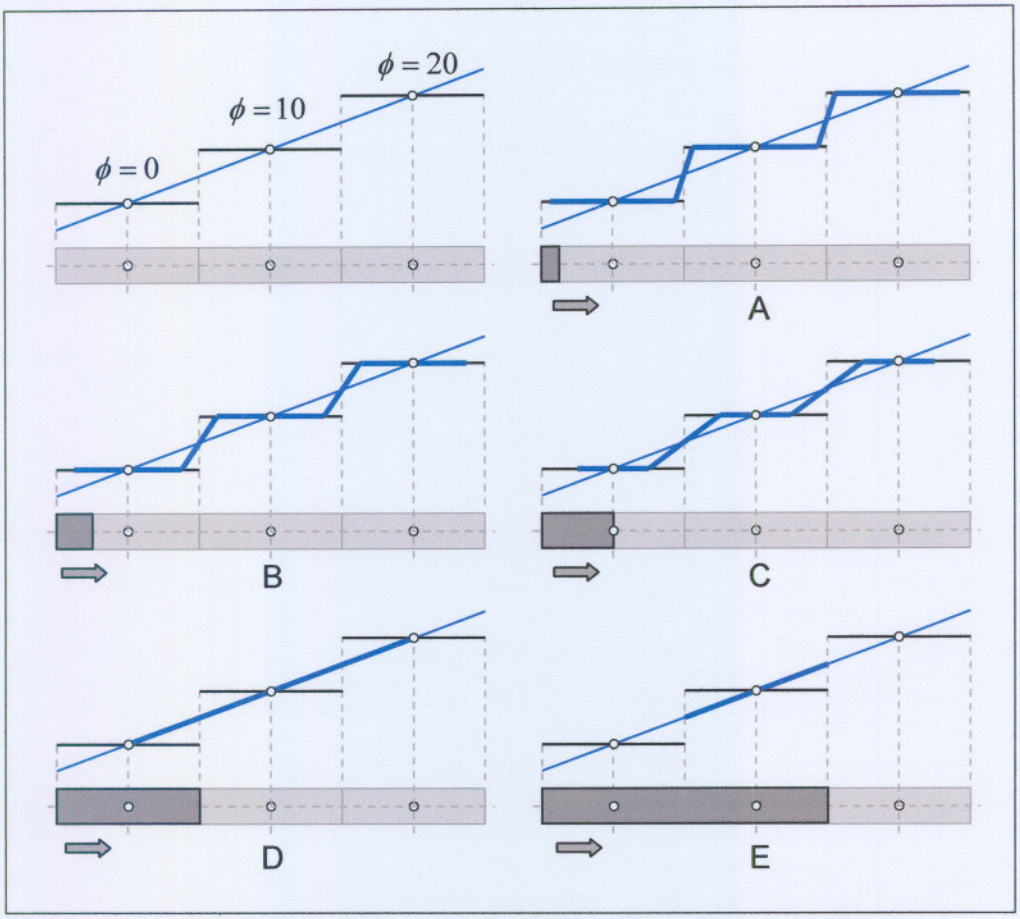


Figure 4.12: Influence of cell size on mapped results

Figure 4.12 demonstrate how the size of a mapped cell influences the result of a volume weighted interpolation. Three cells are shown with a linear variable distribution. Also shown are horizontal lines indicating the average value of each cell. Five examples are shown in the figure where different mapped cell sizes were used to map results from the base mesh, each with an increased size as indicated. In each of these examples the interpolated mapped cell value is shown as obtained when sliding the mapped cell in the direction indicated and at each location plotting the interpolated mapped cell value superimposed over the original solution. The mapping remains conservative regardless of the chosen mapped cell size, however, the profile of the mapped solution changes. In example D and E, the mapped solution coincides with the original distribution. In the other examples the mapped solution consists of partly the average cell value of the original base cell as well as a *mixed value* consisting of a combination of the neighbouring base cell values.

The example illustrates the importance of selecting an appropriate mapped cell size when using volume weighted interpolation for post-processing purposes. To achieve this, the sizes of the

base mesh cells in the vicinity of the point where the weighted value is sought, should be used in the construction of the mapped cell. This process can be automated to relieve the user from having to construct appropriate mapped cells for post-processing purposes.

As a final example of volume weighted interpolation for post-processing purposes, the solution on the hexahedral mesh is mapped onto a cylindrical mesh. The result is shown in Figure 4.13.

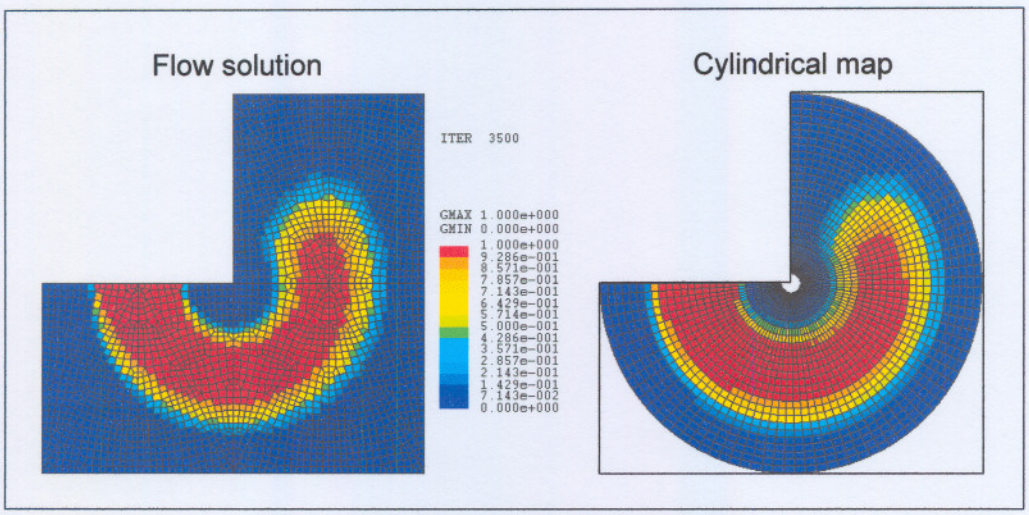


Figure 4.13: Example 2 - Cylindrical map

The examples presented in this section demonstrate the versatility of volume weighted interpolation as a tool for pre- and post-processing in the finite volume method.

4.3.2 Calculation of mass flux across surfaces

It is often required to calculate mass flux through a surface as part of the post-processing that is performed on the results of a simulation. With the collocated variable arrangement, all the dependent variables including velocity vectors are calculated at cell centres while mass flux is calculated across the control volume faces in order to ensure the local conservation of a flow quantity on a control volume basis.

To be able to use control volume faces for mass flux calculations during post-processing, the CFD code must provide access to this information and the control volume faces must be located at the positions in the mesh where mass flux results are required. When using orthogonal meshes the control volume faces are aligned to form surfaces through the mesh that are ideal for mass flux calculations. On complex non-orthogonal meshes, *surfaces* for mass flux calculations are not generally available.

Volume weighted interpolation can be used for accurate *approximations* of mass flux through surfaces. *Surface cells* are created which represents the surfaces where mass flux must be

calculated. Surface cells are thin three-dimensional cells representing two dimensional surfaces that overlap the computational mesh.

During post-processing the velocity vectors at the centres of the surface cells are calculated through a process of conservative momentum mapping. The flow areas of the surface cells together with the velocity vectors at the centres of the cells, are then used to calculate an *approximate* mass flow through the surfaces by means of Eq. (4.3).

$$\dot{m} = \rho \vec{u} \cdot \vec{A} \tag{4.3}$$

Mass flow calculations based on cell centre velocities are only approximate since conservative fluxes are computed across control volume faces in the finite volume method. When thin surface cells are used the error in calculation will be small and the accuracy acceptable.

After a simulation has been completed on a given computational mesh, surface cells are created for the purpose of mass flow calculations. The momentum from the solution on the original mesh is mapped to the surface cells by means of conservative volume weighted interpolation. The velocities at the centres of the surface cells are calculated from the interpolated momentum values. Momentum is interpolated on a component basis using the three Cartesian velocity components, u, v and w from the solution.

$$\begin{aligned} (P_u)_{surface\ cell} &= \sum_{n=1}^{N} (\alpha \rho V u)_n \\ (P_v)_{surface\ cell} &= \sum_{n=1}^{N} (\alpha \rho V v)_n \\ (P_w)_{surface\ cell} &= \sum_{n=1}^{N} (\alpha \rho V w)_n \end{aligned} \tag{4.4}$$

In Eq.(4.4) P_u, P_v and P_w are the volume weighted momentum values of the surface cell computed using the u, v and w velocity components respectively. α is the ratio of volume overlapping between the base mesh cell n and the surface cell. N is the total number of cells overlapped by the surface cell. V is the volume of the base mesh cell n.

A volume weighted density must also be computed for flows with variable density in order to calculate the velocity components from the volume weighted momentum components.

$$\rho_{surface\ cell} = \sum_{n=1}^{N} (\alpha \rho)_n \tag{4.5}$$

The velocity components of the surface cells are obtained by dividing the momentum component for that velocity component by the surface cell volume and density.

$$u_{surface\ cell} = \left(\frac{P_u}{\rho V}\right)_{surface\ cell}$$
$$v_{surface\ cell} = \left(\frac{P_v}{\rho V}\right)_{surface\ cell} \qquad (4.6)$$
$$w_{surface\ cell} = \left(\frac{P_w}{\rho V}\right)_{surface\ cell}$$

Once the velocities are available the mass flux can be calculated using Eq.(4.3). The concept is demonstrated in the following examples:

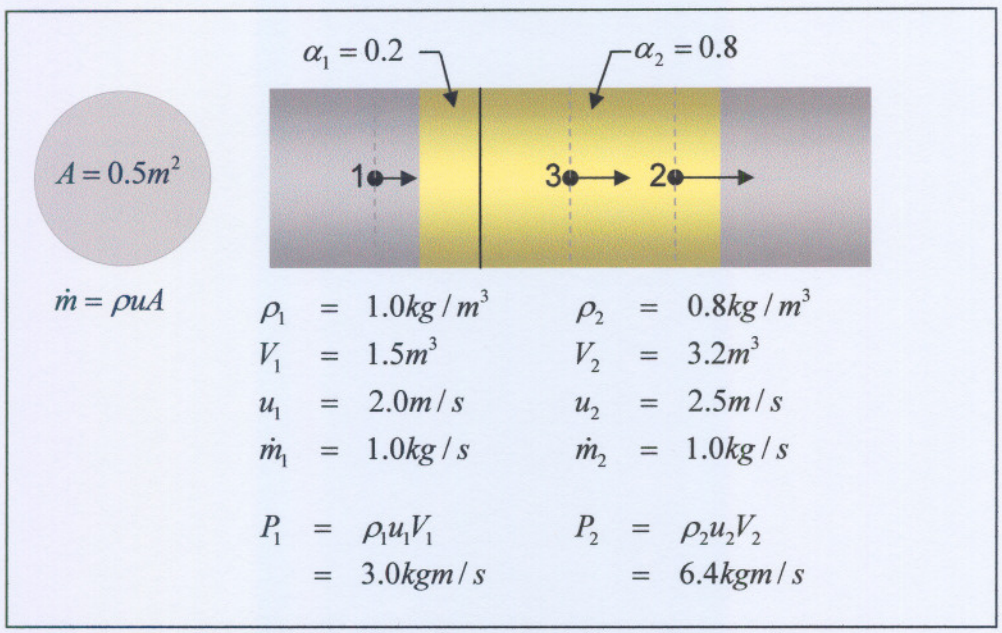


Figure 4.14: Mass flow calculation - Pipe section

Figure 4.14 shows a section of a pipe that is divided into two base cells. The velocity, density and volume are shown for each cell. The velocity for cell number two was calculated so that the mass flow through the pipe is constant. The momentum for each cell is calculated by multiplying the cell mass with the cell velocity.

$$P = mu = (\rho V)u \qquad (4.7)$$

Cell number three overlaps cell one and two with volume fractions of 0.2 and 0.8 respectively. These volume fractions are used in Eq. (4.4) to calculate the volume weighted momentum of cell three.

$$P_3 = \alpha_1 P_1 + \alpha_2 P_2$$
$$= 5.72 kgm/s$$

Eq. (4.5) is used to calculate the volume weighted density of cell three.

$$\rho_3 = \alpha_1 \rho_1 + \alpha_2 \rho_2$$
$$= 0.84 kg/m^3$$

The volume of cell three can either be calculated in the same way as the density or will be known beforehand from the dimensions that were used to create the cell.

$$V_3 = \alpha_1 V_1 + \alpha_2 V_2$$
$$= 2.86 m^3$$

The velocity for cell three is calculated using Eq. (4.6).

$$u_3 = \frac{P_3}{\rho_3 V_3}$$
$$= 2.38095 m/s$$

As a check, the velocity of cell three is used to calculate the mass flow by means of Eq. (4.3).

$$\dot{m}_3 = \rho_3 u_3 A$$
$$= 1.0 kg/s$$

The calculated mass flow satisfies mass conservation.

Incompressible flow through a nozzle is considered in the following example. Figure 4.15 shows a section of a convergent divergent nozzle, meshed with an arbitrary unstructured mesh consisting of hexahedral cells. The inflow and outflow areas of the nozzle are 0.8 m x 0.1 m and the throat area 0.4 m x 0.1 m. Since the flow is symmetrical around the centreline of the nozzle, only the bottom half of the nozzle is used in the simulation with the centreline represented by a symmetry boundary condition.

Water enters the domain at a speed of 1.3 m/s and a mass flow of 52 kg/s (bottom half of nozzle). The water density is a constant 1000 kg/m^3. The boundary conditions as well as a contour plot of the velocity magnitude calculated during the simulation are shown in Figure 4.15. The solution converged after 107 iterations with a specified convergence level of 0.0001.

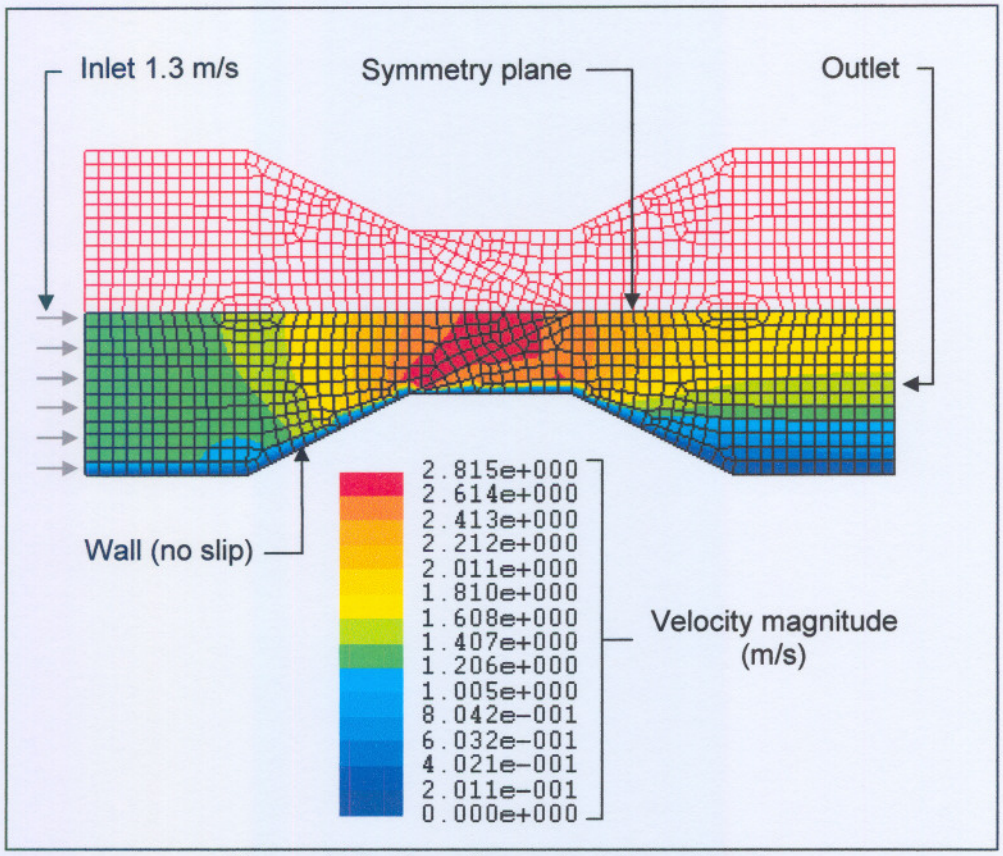
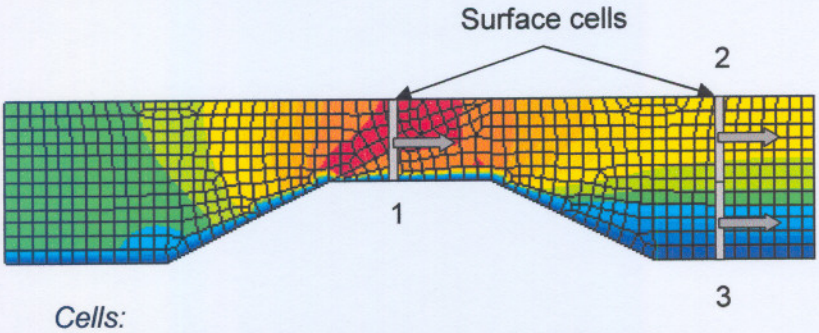


Figure 4.15: Mass flow calculation - Nozzle

Three surface cells are created in the flow domain to demonstrate the mass flow calculation procedure. Surface cell one is located in the throat area of the nozzle while cell two and three are located in the outlet area, together covering the entire cross sectional area of the outlet as shown in Figure 4.16. Each surface cell is created with a thickness of 0.0001 m.

From inspection the mass flow through surface cell one must be 52 kg/s. The sum of the mass flows through cell two and three must also equal 52 kg/s. The mass flow through the three surface cells is obtained by multiplying the cross sectional area of the cells with the velocity calculated from the interpolated momentum values of each surface cell.

Figure 4.16 shows the results obtained for each surface cell. From these results it is evident that the mass flow calculation procedure, based on volume weighted interpolation, is accurate. The method is not limited to single surface cells. Surface cells can be created to form any surface across a domain. The mass flow through the surface is obtained by summing the mass flows through the surface cells defining the surface. In the following example a closed surface is used for mass flow calculations. In this example the surface area vector for each surface cell is directed outwards. In the absence of mass sources within the closed surface the net mass flux across such a closed surface should be zero.



Cells:

Area for mass flow calculation : 0.2m x 0.1m

Thickness of cells in flow direction : 0.0001m

Results:

$u_1 = 2.599 m/s \quad \dot{m}_1 = 51.996 kg/s$

$u_2 = 1.677 m/s \quad \dot{m}_2 = 33.540 kg/s$

$u_3 = 0.927 m/s \quad \dot{m}_3 = 18.540 kg/s$

$\dot{m}_2 + \dot{m}_3 = 52.08 kg/s$

Figure 4.16: Results of nozzle mass flow calculations

Table 4.3 lists the 24 surface cells forming the closed annular surface shown in Figure 4.17. Also listed in the table are the velocity components of the surface cells, calculated by means of a conservative momentum interpolation from the solution mesh to the surface cells. The components of the area vectors of each surface cell are also shown.

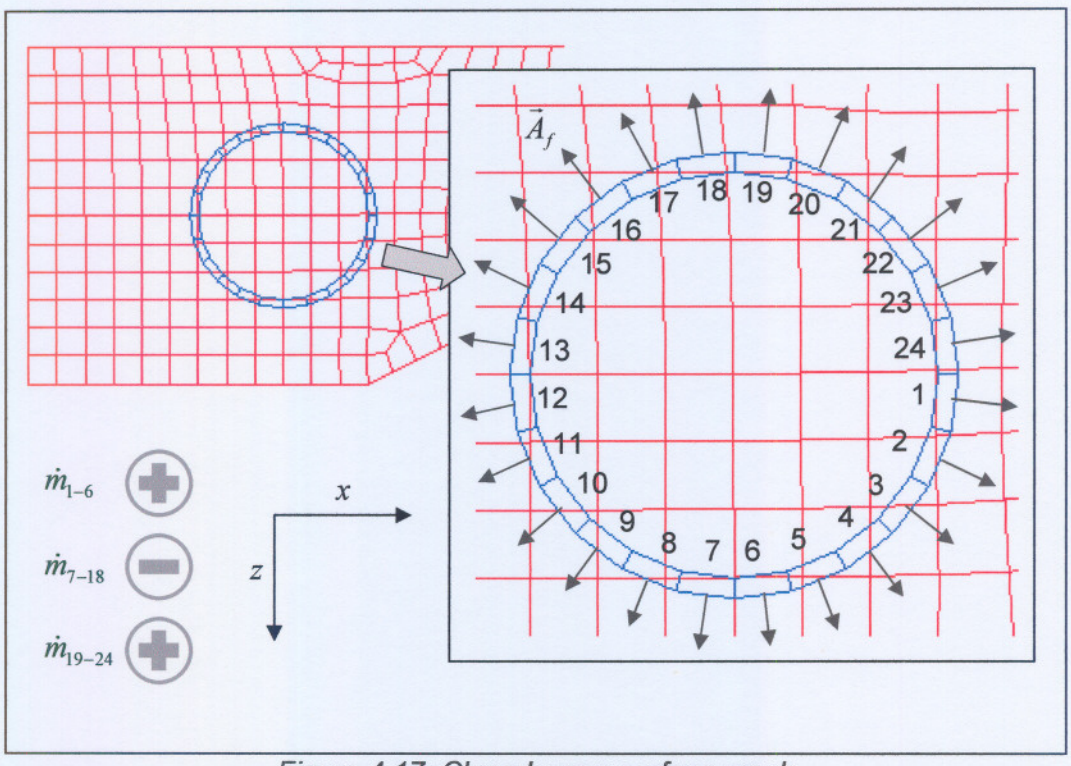


Figure 4.17: Closed curve surface mesh

The mass flow through each surface cell is calculated using Eq. (4.3) and added together to give the net mass flow through the closed surface. The mass flows shaded in grey in the table for cells 7 to 18 is negative, indicating an inflow through the surface, while the positive mass flows of the remaining cells indicates an outflow across the surface. The sign of the calculated mass flows originates from the outward oriented surface area vectors for the closed surface. The sum of the mass flows through the surface cells is approximately zero as required for mass conservation.

Nr	u	w	A_x	A_z	$\dot{m}$
1	1.3624	-0.2496	0.0028470095	0.0003748159	3.7851
2	1.3054	-0.2363	0.0026529905	0.0010989047	3.2035
3	1.2694	-0.2301	0.0022781746	0.0017481048	2.4898
4	1.2401	-0.1981	0.0017481048	0.0022781746	1.7164
5	1.2310	-0.1622	0.0010989047	0.0026529905	0.9225
6	1.2142	-0.1170	0.0003748159	0.0028470095	0.1220
7	1.2354	-0.0845	-0.0003748159	0.0028470095	-0.7037
8	1.2606	-0.0756	-0.0010989047	0.0026529905	-1.5857
9	1.2706	-0.0659	-0.0017481048	0.0022781746	-2.3712
10	1.2841	-0.0575	-0.0022781746	0.0017481048	-3.0259
11	1.2914	-0.0570	-0.0026529905	0.0010989047	-3.4888
12	1.2997	-0.0487	-0.0028470095	0.0003748159	-3.7186
13	1.3078	-0.0470	-0.0028470095	-0.0003748159	-3.7058
14	1.3168	-0.0507	-0.0026529905	-0.0010989047	-3.4378
15	1.3220	-0.0477	-0.0022781746	-0.0017481048	-2.9283
16	1.3350	-0.0506	-0.0017481048	-0.0022781746	-2.2184
17	1.3445	-0.0553	-0.0010989047	-0.0026529905	-1.3307
18	1.3612	-0.0538	-0.0003748159	-0.0028470095	-0.3570
19	1.3787	-0.0646	0.0003748159	-0.0028470095	0.7006
20	1.3928	-0.0946	0.0010989047	-0.0026529905	1.7814
21	1.4054	-0.1120	0.0017481048	-0.0022781746	2.7120
22	1.4094	-0.1452	0.0022781746	-0.0017481048	3.4647
23	1.4023	-0.1736	0.0026529905	-0.0010989047	3.9112
24	1.3983	-0.2218	0.0028470095	-0.0003748159	4.0641
				Total	0.0012

Table 4.3: Results of closed surface mass flow calculation

4.4 Closure

In this chapter the versatility of volume weighted interpolation for pre- and post-processing were demonstrated. As a pre-processing tool volume weighted interpolation greatly simplifies the initialisation of variable fields onto complex meshes and as a post-processing tool, volume weighted interpolation allows data to be interpolated from complex meshes onto simplified meshes for the purpose of data analysis. Volume weighted interpolation for mass flow calculation was described as well as examples presented to demonstrate how this tool can be utilised to enhance the capabilities of the finite volume method. The research is summarised and concluded in Chapter 5.

5. Conclusions

5.1 Summary

The work presented in this thesis describes a novel interpolation technique, applying the finite volume discretisation method and volume weighted interpolation. This technique allows the numerical simulation of convective transport using high-resolution convection schemes on arbitrary unstructured meshes. The need for and importance of interpolation methods for the finite volume method was explained in Chapter 1. The need for interpolation originates from the fact that variables are located at discrete locations throughout the computational domain. While variables are located and solved at cell centres, the discretised transport equations require the availability of variable values at other locations for example cell faces for the evaluation of flux terms.

The concept of volume weighted interpolation was introduced in Chapter 2 as an alternative method of interpolation for the finite volume method. The most notable advantage of using volume weighted interpolation is that the interpolation is conservative and therefore consistent with the finite volume method where conservation is maintained at a control volume level. In order to perform volume weighted interpolation, the calculation of the common volume between two polyhedral cells is required. An algorithm to perform this calculation was described in Chapter 2. It was demonstrated by means of examples how variables can be interpolated conservatively between various types of overlapping meshes.

In Chapter 3 the application of volume weighted interpolation was extended to the modelling of convective transport on arbitrary unstructured meshes. High-resolution convection schemes were implemented using the Orthogonal Projection Interpolation Stencil (OPIS). Test cases were presented to demonstrate the capabilities of OPIS where three-point stencils were constructed to implement high-resolution convection schemes on arbitrary unstructured meshes. OPIS provides a natural means to calculate bounded upwind cell values required for three-point stencils.

In Chapter 4 the focus was on the application of volume weighted interpolation for pre- and post-processing tasks on complex meshes. Complex field initialisations can be performed by mapping variables from one mesh onto another. Volume weighted interpolation can also be applied effectively to calculate mass flows across surfaces during post-processing.

Volume weighted interpolation is a powerful, versatile and robust tool with extensive application capabilities within the finite volume method.

5.2 Opportunities for future research

The application of volume weighted interpolation for the finite volume method should be explored further to investigate and harness all the benefits that it has to offer. The following areas were identified for future research opportunities in this field.

1.

In principle volume weighted interpolation can be used to calculate values at any position within a mesh. This is achieved by creating a mapped cell around the point where the value is required. The shape of the mapped cell can for example be a sphere in a three-dimensional mesh. The solution is then interpolated to the mapped cell by means of volume weighted interpolation. The challenge is to automate the process of creating an optimum sized mapped cell around the point where the value is sought, in order to perform an optimum weighting from the underlying base mesh cells. The algorithm should be able to calculate the optimum mapped cell size from the underlying base cell sizes to produce an accurate interpolated value at that specific location in the mesh. With a tool like this the value of a variable at any point in the domain can then easily be calculated during post-processing by simply specifying the location where the variable value is required.

2.

In the calculations presented in Chapter 3, the OPIS scheme was applied to all the inner faces in the mesh. To improve the computational efficiency, it is possible to develop an algorithm that can check the mesh construction so that volume weighted interpolation is applied only in those regions where it is required, i.e. in those regions of a mesh that is non-orthogonal or where the skewness error creates a problem due to the presence of highly deformed cells.

3.

The OPIS scheme constructs regular cells orthogonal to a face, yet other possibilities exist and should be investigated. It is for example possible to construct regular meshes that are aligned with velocity vectors to allow interpolation along streamlines to be performed. The main issue with this method is the re-meshing required with changes in the velocity field during a simulation.

4.

OPIS should be implemented and tested on tetrahedral meshes. The concept of using volume weighted interpolation to construct three-point stencils on these meshes should work well by

involving additional cells for face value interpolations. As described in Chapter 3 the method of projecting an upwind cell by using the gradient of the dependent variable of the donor cell of a face does not work well for this type of mesh. The reason for this is the calculation of the donor cell gradient which involves only the information from the direct neighbour cells of the donor cell. By taking the values of additional cells into consideration through volume weighted interpolation, improved three-point stencils can be constructed for modelling convection.

5.

Volume weighted interpolation can also be applied to other areas of the finite volume method, for example, in gradient calculations. The different application areas should be thoroughly investigated and tested to ensure that the full benefit of the technique is harnessed.

5.3 Conclusions

From the work presented in this thesis it is evident that volume weighted interpolation has a very important role to play in the finite volume method. The ability to interpolate variables conservatively between meshes provides many possibilities, for example, the coupling of finite volume codes with other software packages that requires variable values at locations other than where they are available in the finite volume code. The advantage of the finite volume method over other methods, such as the finite element method, is that the finite volume method is conservative on a control volume level. Volume weighted interpolation is consistent with this conservative characteristic when it is applied to interpolate variables between overlapping meshes. The volume weighted interpolation methodology therefore fits well within the framework of the finite volume method. The tests cases and examples presented in this thesis is confirmation that volume weighted interpolation is a valuable tool for the finite volume method that should be added to the knowledge pool, further developed and utilised for the benefit of computational fluid dynamics.

References

Anderson, J.D. 1995. **Computational fluid dynamics. The basics with applications.** New York: McGraw-Hill International.

Anton, H. 1995. **Calculus with analytic geometry, 5th edition**. New York: Wiley.

Bird, R.B., Stewart, W.E. and Lightfoot, E.N. 2002. **Transport Phenomena, 2nd edition.** New York: Wiley.

Caretto, L.S., Curr, R.M. and Spalding, D.B. 1972. **Two numerical methods for three-dimensional boundary layers.** *Computer methods in applied mechanics and engineering.* 1: 39-57.

Chakravarthy, S.R. and Osher, S. 1983. **High resolution applications of the Osher upwind scheme for the Euler equations.** *AIAA Paper* 83-1943.

Choi, S.K., Nam, H.Y., Lee, Y.B. and Cho, M. 1993. **An efficient three-dimensional calculation procedure for incompressible flows in complex geometries.** *Numerical heat transfer, Part B*, 23: 387-400.

Chow, P., Cross, M. and Pericleous, K. 1996. **A natural extension of the conventional finite volume method into polygonal unstructured meshes for CFD applications.** *Applied mathematical modelling*, 20: 170-183.

Coelho, P. and Peraira, J.C.F. 1993. **Calculation procedure for 3-D laminar flows in complex geometries using a nonstaggered non-orthogonal grid system.** *Applied mathematical modelling*, 17: 562-576.

Courant, R., Isaacson, E. and Rees, M. 1952. **On the solution of nonlinear hyperbolic differential equations by finite difference.** *Communications in pure and applied mathematics*, 5: 243-255.

Croft, N. 1998. **Unstructured mesh – Finite volume algorithms for swirling, turbulent, reacting flows.** *PhD Thesis*, University of Greenwich, London.

Darwish, M.S. and Moukalled, F. 1996. **The normalized weighting factor method: A novel technique for accelerating the convergence of high-resolution convective schemes.** *Numerical Heat Transfer, Part B*, 30: 217-237.

Du Toit, P.S. 2005. Personal communication. *Flamengro Armscor Business.*

Ferziger, J.H. and Peric, M. 1999. **Computational methods for fluid dynamics, 2nd edition.** Springer.

Gaskell, P.H. and Lau, A.K.C. 1988. **Curvature compensated convective transport: SMART, a new boundedness preserving transport algorithm.** International *journal for numerical methods in fluids*, 8: 617-641.

Gooch, C.O. 2002. **A toolkit for numerical simulation of PDEs:I. Fundamentals of generic finite-volume simulation.** *Computer methods in applied mechanics and engineering.* 192(2003): 1147-1175.

Hirsch, C. 1988. **Numerical computation of internal and external flows, Volume 1, Fundamentals of numerical discretisation.** John Wiley & Sons. New York.

Hirt, C.W. and Nichols, B.D. 1981. **Volume of fluid (VOF) method for the dynamics of free boundaries.** *Journal of computational physics*, 39: 201-225.

Issa, R.I. 1986. **Solution of the implicitly discretised fluid flow equations by operator splitting.** *Journal of computational physics*, 62: 40-65.

Jasak, H. 1996. **Error analysis and estimation for the finite volume method with applications to fluid flows.** *PhD Thesis*, University of London.

Jasak, H. and Weller, H.G. 1995. **Interface-tracking capabilities of the Inter-Gamma differencing scheme.** *Internal report, CFD research group*, Imperial College, London.

Jasak, H., Weller, H.G. and Gosman, A.D. 1999. **High resolution NVD differencing scheme for arbitrary unstructured meshes.** *International journal for numerical methods in fluids*, 31:431-449.

Khosla, P.K. and Rubin, S.G. 1974. **A diagonally dominant second-order accurate implicit scheme.** *Computers and fluids*, 2:207-209.

Kruger, J. 2005. **Object-oriented software development applied to adaptive resolution control in two-fluid models.** *Potchefstroom: North-West University. PhD Thesis.*

Leonard, B.P. 1979. **A stable and accurate convective modelling procedure based on quadratic upstream interpolation.** *Computer methods in applied mechanics and engineering*, 19: 59-98.

Leonard, B.P. 1991. **The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection.** *Computer methods in applied mechanics and engineering.* 88:17-74.

Leonard, B.P. and Mokhtari, S. 1990. **Beyond first-order upwinding: The ultra sharp alternative for non-oscillatory steady-state simulation of convection.** *International journal for numerical methods in engineering.* 30: 729-766.

Muzaferija, S. and Peric, M. 1998. **Computation of free surface flows using interface-tracking and interface-capturing methods.** Chapter 2 in Nonlinear water wave interaction, Mahrenholtz, O. and Markiewicz, M. (eds.), *Computational mechanics publications*, Southhampton.

Noh, W.F. and Woodward, P. 1976. **SLIC (Simple Line Interface Calculation).** *Lecture notes in physics.* 59: 330-340.

Patankar, S.V. 1980. **Numerical heat transfer and fluid flow (Series in computational methods in mechanics and thermal sciences).** New York : McGraw-Hill.

Patankar, S.V. and Spalding, D.B. 1972. **A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows.** *International journal of heat and mass transfer*, 15: 1787-1806.

Peric, M. 1985. **A finite volume method for the prediction of three-dimensional flow in complex ducts.** *Ph.D Thesis*, University of London.

Peric, M. 2002. **News from Star-CD developers.** *Star-CD dynamics.* 18: 11.

Reddy, J.N. 1993. **An introduction to the finite element method.** New York: McGraw-Hill International editions, Engineering mechanics series.

Rhie, C.M. and Chow, W.L. 1983. **A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation.** AIAA Journal, 21(11): 1525-1532.

Rider, W.J. and Kothe, D.B. 1995. **Stretching and tearing interface tracking methods.** *AIAA Paper 95-1717.*

Rider, W.J. and Kothe, D.B. 1998. **Reconstructing volume tracking.** *Journal of computational physics*, 141: 112-152.

Rudman, M. 1997. **Volume-tracking methods for interfacial flow calculations.** *International journal for numerical methods in fluids*, 24: 671-691.

Rudman, M. 1998. **A volume-tracking method for incompressible multi-fluid flows with large density variations.** *International journal for numerical methods in fluids*, 28: 357-378.

Stroustrup, B. 1997. **The C++ programming language, 3rd edition.** Addison Wesley.

Tao, W.Q., He, Y.L., Li, Z.Y., and Qu, Z.G. 2004. **Some recent advances in finite volume approach and their applications in the study of heat transfer enhancement.** *Proceedings of ICHMT. International symposium on advances in computational heat transfer. April 19 - 24, Norway.*

Ubbink, O. 1997. **Numerical prediction of two fluid systems with sharp interfaces.** *PhD Thesis*, University of London.

Ubbink, O. and Issa, R.I. 1999. **A method for capturing sharp fluid interfaces on arbitrary meshes.** *Journal of computational physics*, 153: 26-50.

Van Leer, B. 1997. **Towards the ultimate conservative difference scheme. A second–order sequel to Godunov's method.** *Journal of computational physics*, 135: 229-248.

Versteeg, H.K. and Malalasekera, W. 1995. **An introduction to computational fluid dynamics, the finite volume method.** Addison Wesley Longman.

Weller, H.G., Tabor, G., Jasak, H. and Fureby, C. 1998. **A tensorial approach to computational continuum mechanics using object-oriented techniques.** *Computers in physics.*, 12(6): 620-631.

Youngs, D.L. 1982. **Time-dependent multi-material flow with large fluid distortion.** In *Numerical methods for fluid dynamics*, London: Academic press: 273-285.

Zhu, J. and Rodi, W. 1991. A **low dispersion and bounded convection scheme.** *Computer methods in applied mechanics and engineering*, 92: 87-96.