UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Numerical Computation of Incompressible Flows on Adaptive and Unstructured Grids

Duarte Manuel Salvador
Freire Silva de Albuquerque

Supervisor: Doctor José Carlos Fernandes Pereira
Co-Supervisor: Doctor José Manuel da Silva Chaves Ribeiro Pereira

Thesis approved in public session to obtain the PhD Degree in
Computational Engineering
Jury final classification: Passed with Merit

Jury
Chairperson: Chairman of the IST Scientific Board

Members of the Committee:
Doctor Paulo Jorge dos Santos Pimentel de Oliveira
Doctor José Carlos Fernandes Pereira
Doctor António Manuel Gameiro Lopes
Doctor Luís Rego da Cunha de Eça
Doctor André Calado Marta

2013

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Numerical Computation of Incompressible Flows on Adaptive and Unstructured Grids

Duarte Manuel Salvador Freire Silva de Albuquerque

Supervisor: Doctor José Carlos Fernandes Pereira
Co-Supervisor: Doctor José Manuel da Silva Chaves Ribeiro Pereira

Thesis approved in public session to obtain the PhD Degree in
Computational Engineering
Jury final classification: Passed with Merit

Jury
Chairperson: Chairman of the IST Scientific Board

Members of the Committee:
Doctor Paulo Jorge dos Santos Pimentel de Oliveira, Professor Catedrático da Universidade da Beira Interior
Doctor José Carlos Fernandes Pereira, Professor Catedrático do Instituto Superior Técnico, da Universidade de Lisboa
Doctor António Manuel Gameiro Lopes, Professor Auxiliar da Faculdade de Ciências e Tecnologia, da Univerisade de Coimbra
Doctor Luís Rego da Cunha de Eça, Professor Auxiliar do Instituto Superior Técnico, da Universidade de Lisboa
Doctor André Calado Marta, Investigador Auxiliar do Instituto Superior Técnico, da Universidade de Lisboa

Funding Institutions
Fundação para a Ciência e a Tecnologia (FCT)
Doctoral Scholarship with the reference SFRH/BD/48150/2008

2013

A todos aqueles que sempre me apoiaram, o meu muito obrigado.

Abstract

This Thesis explains how to implement a Computational Fluid Dynamics (CFD) code on the unstructured grid framework, covering different topics from: computational geometry, numerical analysis, Finite-Volume (FV) discretization, least squares methods, parallel computing and other important topics.

The majority of the applications presented in this work concerns a substantial review of FV schemes for unstructured grids, the development of a new Total Variation Diminishing (TVD) convective scheme on unstructured grids and the creation of two automatic h-adaptive refinement algorithms, which use the information provided by an error estimator in order to increase the number of computational points at the zones of higher numerical error, creating more efficient grids without user intervention.

The developed relative error estimator dR allowed to capture the sequence of different sharp corner vortices introduced analytically by Moffatt (1963), in the square lid cavity flow problem. The smallest corner vortex detected is $O(10^{-16})$ smaller in velocity magnitude compared with the cavity primary vortex.

The Residual Least Squares (RLS) error estimator is also implemented with an adaptive refinement procedure that allows the application to different cases without requiring the change of a constant that drives the adaptive algorithm, unlike other methods presented in the literature. The adaptive algorithm is applied to a series of test cases with a known analytical solution in order to evaluate its performance and efficiency.

Keywords

incompressible flows, finite volume method, unstructured grids, adaptive grids, error estimators, least squares method, TVD schemes, coupling algorithms, code verification, computational geometry

Resumo

Esta Tese explica como implementar um código CFD no contexto de malha não-estruturada, cobrindo diversos tópicos: como geometria computacional, análise numérica, discretização em volume finito, métodos de mínimos quadrados, cálculo paralelo e outros tópicos importantes.

As principais aplicações desenvolvidas no código dizem respeito: à revisão de diversos esquemas de volume finito para malha não-estruturada, à proposta dum novo esquema convectivo TVD para malha não-estruturada e à criação de dois algoritmos de refinamento adaptivo-h, que usam a informação fornecida por um estimador de erro de modo a aumentar o número de pontos computacionais em zonas de maior erro numérico, criando malhas mais eficientes e sem intervenção do utilizador.

O estimador de erro relativo dR permitiu capturar em detalhe uma sequência de pequenos vórtices localizados nos cantos inferiores duma cavidade quadrada, cujo o resultado analítico foi introduzido por Moffatt (1963). O vórtice mais pequeno tem uma velocidade $O(10^{-16})$ mais pequena em relação ao vórtice principal da cavidade.

Um estimador de erro absoluto RLS é também implementado com um procedimento de refinamento adaptativo que permitiu a sua aplicação a diferentes casos. Não necessitando a intervenção do utilizador, visto não ser necessário mudar valores empíricos ao contrário de outros algoritmos presentes na literatura. O procedimento adaptativo é aplicado para uma série de casos testes de modo a avaliar o seu desempenho.

Palavras-chave

escoamentos incompressíveis, métodos de volume finito, malhas não-estruturadas, malhas adaptativas, estimadores de erro, método de mínimos quadrados, esquemas TVD, algoritmos de acoplamento, verificação de código, geometria computacional

Agradecimentos

O meu primeiro agradecimento é feito aos meus pais e ao meu irmão, pela paciência e pelo apoio que sempre me deram ao longo da vida. Especialmente por me terem ensinado os valores da responsabilidade e do trabalho árduo.

Um agradecimento muito especial à Inês Martins, o seu apoio e carinho foram fundamentais para que este trabalho fosse finalizado.

Agradeço ao meu orientador Prof. José Carlos Fernandes Pereira pela sua orientação, interesse e procura de avanços científicos nesta área. Beneficiei muitas vezes dos seus conselhos e da sua vasta experiência, a quando da disciplina de Mecânica dos Fluidos Computacional, durante a minha tese de Mestrado e especialmente durante a minha Tese de Doutoramento.

Agradeço ao meu co-orientador Prof. José Manuel Chaves Pereira por toda a sua ajuda. Aprendi bastante com o professor, visto o seu Doutoramento ter sido nesta área científica. Aprendi sobre ferramentas LINUX e software comercial CFD STAR. Durante este trabalho, beneficiei bastante das suas sugestões e comentários.

Agradeço aos meus orientadores a liberdade de escolha no meu plano de trabalhos. Por um lado devido ao gosto que adquiri pelas áreas de computação numérica, por outro lado devido a necessidade, por parte do mercado de trabalho internacional, de doutores especializados no desenvolvimento de códigos CFD.

Um agradecimento ao Dr. João Paulo Magalhães pela a sua transmissão de conhecimentos do código SOL e de literatura sobre esta área, sem a qual, a realização deste trabalho teria sido mais árdua. Agradeço aos vários colegas que tive e tenho no LASEF que sempre me ajudaram e apoiaram: Rodrigo Taveira, Jorge Navalho, Pedro Neto, Joaquim Simas, Rita Ervilha, Flávio Sousa, Rafael Vicente, João Miranda e António Amador. A respectiva lista não é maior devido a constrangimentos de espaço, pelo que peço desculpa aos actuais e antigos colegas não referidos.

Um agradecimento ao apoio financeiro da Fundação para a Ciência e Tecnologia (FCT), pela bolsa de Doutoramento SFRH/BD/48150/2008, e que provavelmente sem ela a oportunidade de realizar este trabalho nunca teria sido possível.

Contents

List of Figures

List of Tables

List of abbreviations

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
AVG	Arithmetic Average interpolation scheme
BC	Boundary Conditions
CAD	Computer-Aided Design
CFD	Computational Fluid Dynamics
CDS	Central Differencing Scheme
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
GMRES	Generalized Minimal Residual
ILU	Incomplete LU factorization
LES	Large Eddy Simulation
MMS	Method of Manufactured Solutions
PISO	Pressure Implicit with Split Operator
RLS	Residual Least Squares
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SVD	Singular Value Decomposition
TC	Tangential Correction
TRI	Triangular interpolation scheme
TSTE	Taylor Series Truncation Error
TVD	Total Variation Diminishing
UDS	Upwind Differencing Scheme
WLS	Weighted Least Squares

List of symbols

Please note that some symbols may have different meanings, depending of the context they refer to:

Common symbols

A, B	analytic constants
a_p, a_l	momentum matrices entries
$\mathbf{b}$	coefficient vector (WLS)
C	contribution of different schemes
D, L	reference diameter, length
$\mathbf{d}$	distance vector between cells P and P_n
E	numerical error
E_T	Taylor series error estimator
$\mathbf{E}_R$	residual least squares error estimator
$\mathbf{e}$	least squares residual vector (WLS)
$\mathcal{F}(P)$	the set of faces of cell P
f	face
$\mathbf{f}$	coordinates of the face centroid f
h	hydraulic diameter
K_0	modified Bessel function of second kind and zero order
M_j	jet momentum
P	cell or volume control (mesh)
$\mathbf{P}$	coordinates of the cell centroid P
p	pressure
p'	pressure correction
$\mathbf{R}_M$, R_C	momentum and continuity residuals

Re	Reynolds number
r, γ	polar coordinates
S_f	face normal or surface vector (mesh)
S	source magnitude
U_f	face velocity
$\mathbf{u}$	velocity vector
u, $v\,w$	velocity components
V_P	cell volume
v	vertex
$\mathbf{W}$	weight matrix (WLS)
w_P	weighted function of cell P
$\mathbf{X}$	least squares rectangular matrix (WLS)
$\mathbf{x}_p$	cell P centroid
x, y	cartesian coordinates
$\mathbf{y}$	observation vector (WLS)
Ω	arbitrary domain or a cell
$\partial\Omega$	boundary of the respective Ω
$\alpha(f)$	warp angle of face f
α_u, α_p	momentum and pressure relaxation factors
Δt	time step
λ_c	limit for the convective stability criterion
λ_d	limit for the diffusive stability criterion
ϕ	transported variable or computational variable
ρ	density
μ	dynamic viscosity
ν	kinematic viscosity
Γ	diffusion coefficient
$\sigma(f)$	skewness of face f

Operators

$\nabla\phi$	gradient
$\nabla \cdot \mathbf{u}$	divergence
$\nabla^2\phi$	Laplacian
$\mathbf{u} \otimes \mathbf{v}$	dyadic product

$\|\mathbf{u}\|$	Euclidean norm
vers($\mathbf{u}$)	versor
$\mathbf{A}'$	transpose

Chapter 1

Introduction

1.1 Theme

This Thesis documents the improvements of an in-house Computational Fluid Dynamics (CFD) code developed at the laboratory of fluid simulation in energy and fluids (LASEF), that was coined or named SOL, as an abbreviation of the word solver. During this Thesis, several numerical techniques to solve the Navier-Stokes equations are implemented, using the Finite Volume (FV) method on adaptive and unstructured grids with arbitrary topology. This last part means that 3D polyhedral grids can be imported from the different software packages into the SOL code for several objectives like computing a laminar flow in a complex geometry or checking the implemented numerical schemes or even comparing and verifying the obtained results of other CFD codes.

The majority of the applications of this Thesis are done in the framework of the automatic adaptive refinement algorithms driven by *a posteriori* error estimator, which tries to estimate the different error scales in the computational grid, providing which cells have higher numerical error than others. These cells are refined[1] in order to reduce the local error and improve the solution accuracy. The adaptive algorithm makes this cell selection without user intervention and, if an universal rule is possible, the algorithm will save time to the CFD engineer at the meshing process which is the bottleneck in an industrial CFD simulation.

In the SOL code, the FV discretization of the adaptive grids is done in the same way as for the unstructured ones. So, as a consequence, non specific methods are required in the discretization of the governing equations near the interfaces created by the cells

[1]Refining a cell, means dividing it in smaller cells and consequently decreasing the local mesh size.

with different levels of refinement. This requires that the author has knowledge of the different procedures on unstructured grids, which have several difficulties when compared to a structured grid code. The main differences are the arbitrary number of neighbors that a cell can have and that the grid does not have any preferential direction: due to this, several numerical techniques have to rearranged in a different way in an unstructured grid code.

In the framework of unstructured grids a wide range of topics must be covered during this Thesis from computational geometry, numerical analysis, differential equations discretization techniques, code programming, parallel computing, optimization algorithms and non-linear analysis. This framework is also important because CFD applications require complex geometry capability, which in order to obtain an acceptable accuracy is important to use an unstructured grid approach.

The unstructured grid capability is highly presented in commercial CFD code packages where Computer-Aided Design (CAD) models can be imported to create a grid of the intended geometry as realistic as possible. The problem with these packages is that they tend to be treated as black boxes to the common user and questions like numerical accuracy and stability are not easily answered. On the other hand, academic codes do not have the same resources as some companies, due to financial constrains, but specially by the different number of code developers in their team.

In the context of unstructured grids with the Finite Volume Method (FVM) there is a need in the scientific community for the development of new numerical techniques, like new diffusive and convective schemes, to improve the accuracy of the existing methods, stabilization algorithms like TVD techniques, high order[2] schemes which are growing faster in the finite element framework with a high number of CFD applications, moving grid algorithms and immersed boundary techniques. This last two are used for fluid-structure problems. Also, the majority of the Large Eddy Simulation (LES) and other turbulence modeling methods are developed for a structured grid basis and are not easily applicable to a unstructured grid one.

1.2 Motivation

The commercial CFD packages are nowadays used on a wide range of cases, which can include complex geometries and turbulence modeling, requiring a high number of

[2]High order means forth, sixth and eighth order accurate schemes. Some authors use this terminology for schemes that include correction terms due to grid quality or TVD schemes, which is not a fully correct terminology.

grid points in order to obtained an accurate solution. On the other hand, the numerical techniques used by these packages have a high robustness due to the requirement to obtain a numerical solution in a wide range of cases.

To understand the possible characteristics of a CFD code, a triangle can be consulted, which is presented in figure 1.1. The basic idea is that there exist three important characteristics in a CFD code: robustness, accuracy and efficiency. Normally these three characteristics cannot be achieved easily in the same code, so a compromise must be established.

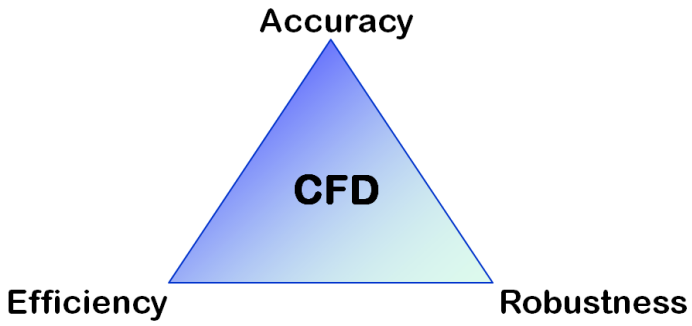


Figure 1.1: CFD code triangle.

For example, commercial codes will try to be efficient in order to obtain the results in a small time interval and have robustness, as explained previously. Another case are the codes that use spectral methods which have a high accuracy since they are a high order method, but they are very restrict since they can be only used in uniform and non-uniform grids and need periodic boundary condition in some directions of the domain: as consequence their efficiency and robustness are lower than the commercial codes.

One important goal of this Thesis is to understand ways to know and control the numerical error (specially in the context of unstructured grids), determinate if the automatic grid refinement is important for the CFD codes and, if it is possible, to improve the present FV discretization schemes.

In the case of fluid-structure interaction problems it may be required to couple different codes, one to solve the structure equations and another to solve the flow equations. However, there are some problems with this approach concerning the restrictions that can come from this coupling, not giving the user a satisfied control. This happened in a previous work of Albuquerque et al. (2011), where the coupling between equations could only be done explicitly, as presented by the example in figure 1.2. Other fluid-structure problems may have high degrees of grid deformation leading

to restrictions that require the use of immersed boundary in order to be simulated.

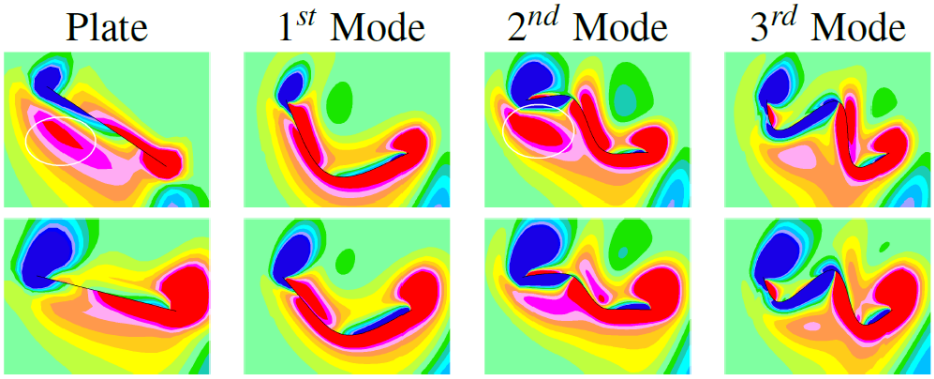


Figure 1.2: Example of the author previous work in the context of fluid-structure interaction problems - from Albuquerque et al. (2011).

Also, there is some demand from the industry to use multi-physics software packages, which couple different types of differential equations in an unique code. Once again the user may do not have the necessary control of the solution and require the programming of his own code to solve its specific problem.

1.3 Literature Survey

1.3.1 Error Estimators and Adaptive Refinement Algorithms

CFD applications on complex geometries require too much consuming efforts allocated to the mesh generation process and also the computing time. Adaptive techniques reduce the time of the unstructured mesh generation and potentially decrease the computing time because the adaptive mesh has a smaller number of cells than the equivalent uniform mesh. The adaptive refinement algorithm requires an error estimator to provide information of the local error. With this quantity, the adaptive algorithms can select which grid cells will be selected for refinement and which will not.

There is a significant number of adaptive grid methods in the CFD literature with the goal to reduce and control the numerical error. These algorithms can be separated into different categories:

- The local reference length can be decreased by splinting certain cells into smaller ones, in order to reduce the local error. This type of method falls into the cate-

gory of h-adaptivity and is the most popular adaptive method in the literature, therefore implemented in this Thesis.

- The number of computational points can be maintained but its location is changed in order to lower the numerical error of the solution and obtain an uniform distribution of the error field. These methods belong to the category of r-adaptivity.

- The local error can be reduced by increasing the order of the discretization method, which is applied in the context of Finite Element Methods (FEM). This type of algorithms falls into the category of p-adaptivity. A problem can take place if the solution is not smooth, due to the formations of wiggles or oscillations resultant of instabilities that occur at high order methods, so generally it is required some mechanism to lower the method order near these discontinuities. There is a lack of this type of methods in the context of FVM.

Also it is possible to create adaptive algorithms by combining two or more of these categories. For example, there exists applications that use hp-adaptive methods in the literature. Some authors refer in their works that the ideal adaptive algorithm will be the combination of an error estimator with a grid generator. There is a small number of applications of triangle grid generators in this way, to the author's knowledge.

From the point of view of Finite Element Methods (FEM), the error estimators are well established: gradient recovery methods, explicit and implicit residual error estimators. A summary of different error estimators in this framework has been reviewed, see e.g. Babuska and Rheinbolt (1978); Peraire et al. (1987); Berger and Colella (1989); Ainsworth and Oden (1997); Aimsworth and Oden (2000); Gratsch and Bathe (2005); Plewa et al. (2005); Segeth (2008). FEM are based in the classic Galerkin approach and this tends the flux estimation to not be consistent between elements, and consequently the numerical method lacks of local conservation properties, but not globally. Nevertheless, the FEM has been successfully applied to fluid flow problems, although with less success in turbulent flows.

For the computation of higher Reynolds flows in the FEM context it is required to add different stabilizing methods in the formulation, due to the numerical errors in this flow regime. On the other hand, it is known that the incorrect choice of interpolation functions for the velocity and pressure fields could originate instabilities in the numerical solution, see e.g. Franca and Nesliturk (2001). In the literature, different numerical techniques have been developed to solve these numerical instabilities from turbulent flows, like the unusual stabilized finite element method Franca and Farhat (1995), the variational multiscale method Gravemeier (2006) and Hachem et al. (2010)

and the residual based stabilized finite method Calhoun (2001). Also worth mentioning, adaptive grid refinement with the FEM has been already presented in several works, see for example Lacasse et al. (2004); Pelletier et al. (2004); Pelletier and Camareo (2005).

In the case of FVM there is a lower number of error estimators and automatic adaptive algorithms, when compared with the FEM framework. Also the FVM is more popular in flow problems due to their conservative proprieties for coarse grids and there is a better convergence when using turbulent models. This happens because the diffusive and convective terms are face integrals, (these methods are face-based). The FVM can be simply applied to cells with arbitrary topology, which cannot be easily done with the FEM since it requires the computation of the respective form functions and the integration with known Gauss points.

The arbitrary topology featured in the computational grid, permits the use of 3D polyhedral grids, which have the required versatility to be used on complex geometries and have the efficiency in terms of the computation time. This kind of characteristics are not possible to obtain with the tetrahedral based grid, see Juretic and Gosman (2008) for details. Also the majority of CFD applications use the FVM, this fact increases the need for the development of new error estimators and automatic adaptive refinement algorithms in this specific framework.

The Richardson Extrapolation Berger and Oliger (1984), Thompson and Ferziger (1989) is one the first developed error estimators in the FVM context and requires the solutions on two meshes with different spacings, which can be difficult to obtain in 3D industrial flow configurations. There are some attempts of single-mesh error estimators, based on energy conservation and angular moment conservation equations, see Haworth et al. (1993). An error estimator based on high order face interpolation was proposed by Muzaferija and Gosman (1997) and later, Jasak and Gosman (2000a,b,c) proposed an error estimator based in the Taylor series truncation error and another one based in the conservation of the second moment of the transported variable. Ait-Ali-Yahia et al. (2002) has applied the Taylor series truncation error to edges integral in the framework of r-adaptivity. An error estimator based on the residual error from the governing equations was investigated by Jasak and Gosman (2001) and afterwards Juretic (2004) extended it to a face based error estimator. Also some techniques were created to extend error estimators from FEM framework to a cell centered FVM, one with the concept of the Morley interpolation by Nicaise (2006) and another with the approach of self-equilibrium fluxes by Jasak and Gosman (2003).

Most of these methods still cannot be applied to all cases, since they are driven

by empirical values dependent of the numerical error distribution. As a consequence, previous knowledge of the solution by the user is required. This is not suitable, when building an automatic refinement procedure.

Very often a second order scheme for adaptive grids is achieved by applying specific projection methods in the grids interfaces[3] for Euler equations by Martin and Colella (2000) or incompressible Navier-Stokes equations by Martin et al. (2008). These schemes cannot be used on triangular, tetrahedral, prismatic adaptive and polyhedral grids, which are used in problems where the geometric complexity is relevant and important.

1.3.2 Cavity Flows

In this Thesis, the classic cavity flow will be used as a benchmark to the numerical solutions. The standard lid-driven cavity flow for Reynolds number $Re = 1000$ is well studied and documented in the following works: Burggraf (1966); Bozemann and Dalton (1973); Shankar and Deshpande (2000); Cheng and Hung (2006); Gustafson and Halasi (1986, 1987). Also, a number of well documented benchmark solutions for this cavity flow are available (*e.g.* Ghia et al. (1982); Botella and Peyret (1998); Erturk et al. (2005)).

Ham et al. (2002) has considered the same case for $Re = 400$ for Cartesian grid method with anisotropic adaptation driven by an error estimator based on the second derivative of the fluid velocities and mesh size, which could be considered one of the first cavity flow studies with an adaptive grid.

The lid-driven cavity flow on the vicinity of the lower corners displays multiple eddies in creeping flow regime. Moffatt (1963) has shown that under the assumption of 2D Stokes flow, the sharp corner flows are formed by the intersection of two boundaries, where either the velocity or the tangential stress vanishes on each boundary with a critical angle opening, which contains a sequence of vortices descending into the corner.

These corner eddies are very small compared to the cavity scale and tend to have little impact on the nature of the bulk flow. However, their description is relevant to understand the flow topology in increasingly slender cavities, see Heaton (2008). Other theoretical implications are related with 3D corner flows, see Collins and Dennis (1976); Shankar (2005).

The numerical resolution of corner flows has been pursued and constitutes a challenging task in terms of numerical resolution, efficiency, robustness, adaptive mesh

[3]Where interfaces are the zones between cells with different levels of refinement.

refinement and error estimation. One of the first attempts, and maybe the most impressive, was conducted by Gustafson and Leben (1988) that computed the Stokes flow solution in an uniform grid in the whole domain and then projected it on finer local grids near the corner. A sequence of smaller domains allowed to predict up to 21 corner vortices with the local maximum stream function intensity of 10^{-92}. The lack of global interaction prevents their solution to be a benchmark test case for Navier-Stokes solvers. Several other authors combined the solutions of the Navier-Stokes equations with exponential mesh refinement of the cavity corners flow regions and asymptotic of the flow near corners to predict the series of vortices, see Shapeev and Lin (2009).

1.3.3 TVD Schemes

The convective term discretization is of great importance in a CFD code but presents several challenges. The convective schemes described in this Thesis can achieve results with high accuracy even for unstructured grids with low quality in smooth regions. Near discontinuities, that occur for example near shocks of compressible fluid flows, unphysical and undesirable oscillations (overshoots and undershoots), are presented in the numerical solution. Also, this behaviour affects severally the convergence of the solution, which is critical in the case of the non-linear Navier-Stokes equations. This happens because the second order convective schemes cannot guarantee boundedness of the numerical solution in these regions.

One way to avoid this problems is to use a monotone convective scheme like the UDS, which guarantees a bounded solution even near these discontinuities and cleared of numerical oscillations. The UDS has a low accuracy due to its first order characteristic and suffers from numerical diffusion which disperses the solution in the computational domain. As a consequence, the converged field has a false diffusive distribution, even if the governing equation does not have a diffusive term.

A way to solve this problem is by creating a blended convective scheme that combines the boundedness and stability properties of the UDS scheme and the high accuracy of a second order method. Several methods based in this principle are available in the literature, such as: the total variation diminishing (TVD) based schemes Warming and Beam (1976), normalised variable diagram (NVD) based schemes Leonard (1991), flux spliting Steger and Warming (1981), Goudunov schemes Sweby (1984) and approximated Riemann solvers Godunov (1959).

Also a very important review work on the design of convective bounded schemes in the cell centered FV framework is done by Waterson and Deconinck (2007), which

includes a comparison of a high number of bounded convective schemes for two test cases.

From the literature, several authors proposed algorithms to implement TVD schemes on unstructured grids, as for example: Bruner and Walters (1995), Darwish and Moukalled (2003),Casulli and Zanolli (2005), Li et al. (2008) and Kong et al. (2013). Also Jasak et al. (1999) suggested an extension of the NVD to the unstructured grids framework guarantying boundedness of the numerical solution. In his work the gamma convective scheme is presented and it is used to solve compressible fluid flow problems on Cartesian and adaptive grids.

In this Thesis the concept of TVD schemes is explained, focusing in the issue of how this type of convective schemes can be applied on unstructured grids. Also, a new TVD method is proposed, which is showed to converge in an inferior number of iterations and to have more bounded results than other approaches. The values of the overshoots and undershoots are lower when compared to other alternative methods, specially if the grid quality is not high.

1.4 SOL Code

The SOL code was created by the LASEF group around 2003, which main goal was to create a common platform for CFD code developing. In order to the research group benefit from the different synergies by working in the same numerical code, instead of each researcher working in their individual code separately.

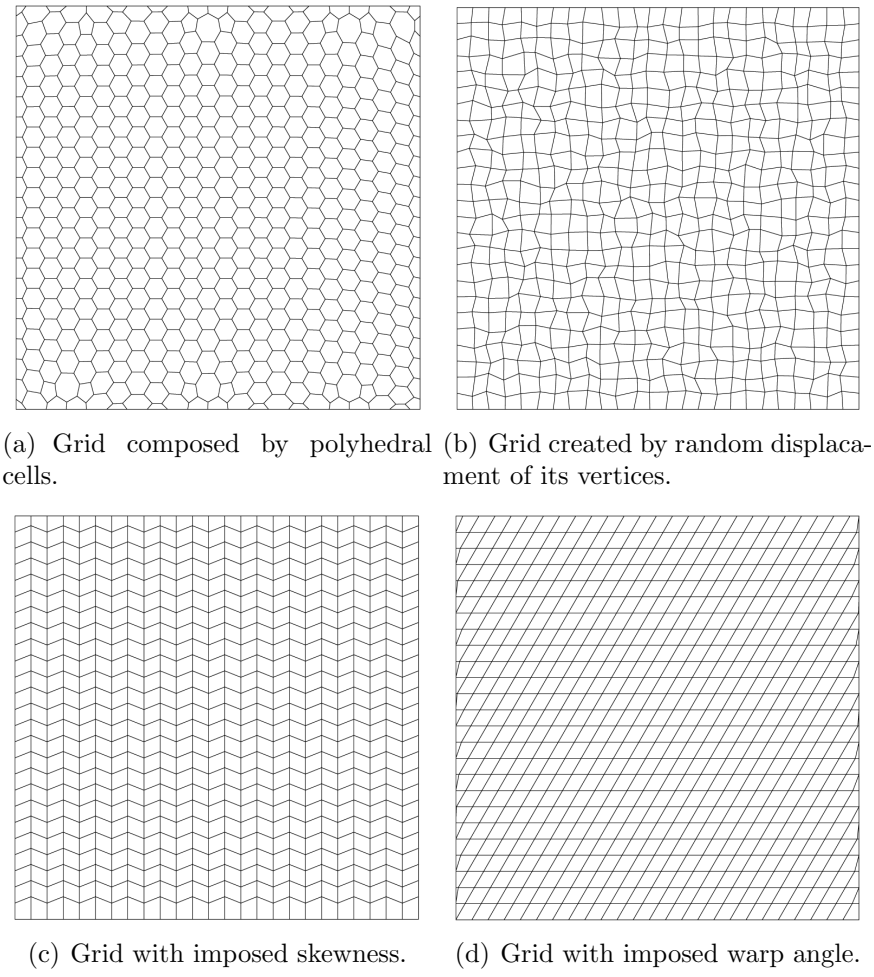
The code was developed for the computation of 3D flows on unstructured and adaptive grids. Since its creation, a Master Thesis was concluded by Reis (2005) which is focused in parallel computation of the Euler equations with a free-matrix method. Magalhães (2011), which was the coordinator of the SOL project, concluded his PhD thesis and his work is focused in the creation and development of this tool for the computation of numerical problems on adaptive grids.

The present coordinator, which is the author of this Thesis, created an importing grid system and reviewed almost all aspects of the previous works. From the computation of grid geometric properties, implicitly governing equations assembly, review of several numerical schemes, code verification with analytical solutions and other different aspects.

The code is written in C language, it has a git repository for code transmission between different developers and it uses the doxygen library to create a code documentation which facilitates the understanding of the several lines of code. Also all of

this tools are integrated in an editable wiki website which have tutorial examples for new developers. Only recently, the OpenFOAM added to the general public some of these tools, which makes the author believe that the SOL project is going well in its own path.

The code can import unstructured grids, including 3D polyhedral grids, from several CFD packages like STAR and OpenFOAM. Also the author created a database with different grids to study the accuracy of the FV schemes implemented in the code and some of these grids have imposed grid quality parameters[4]. Examples of these grids are showed in the figure 1.3.



(a) Grid composed by polyhedral cells.

(b) Grid created by random displacament of its vertices.

(c) Grid with imposed skewness.

(d) Grid with imposed warp angle.

Figure 1.3: Examples of the computational grids in the SOL repository.

The code has, at this time, 160 thousand code lines and 60 files. It also uses different libraries packages like: AZTEC, METIS, LAPACK, BLAS, ATLAS and TECIO; which are packages that are also included in other academic CFD codes. The code is also

[4]The different grid quality parameters are explained in Chapter 2.

parallelized with the MPI package and the several aspects like grid division, creation of hallo cells and communication during the velocity-coupling algorithms is working at the present time.

1.5 Present Contributions

The main contributions of the author, presented in this Thesis are:

- A full polyhedral and unstructured grids treatment is reviewed and verified for several FV schemes.
- Verification of a new second order diffusive and convective scheme based in the weighted least squares WLS method by creating a linear regression centered at the face centroid.
- A procedure to apply TVD schemes on unstructured grids which considers the grid quality and has better behaviour than other approaches presented in the literature.
- The results verification and exhaustive analysis of an adaptive procedure with a relative error estimator, which is suitable for multiscale problems. The adaptive method predicted analytical results from the corner vortices without forcing the local refinement.
- An adaptive procedure driven by an absolute error estimator which is suitable for unstructured grids. The results are verified by solving different cases with a known analytical solution. The automatic adaptive algorithm does not require an external input unlike other methods from the literature.

1.6 Thesis Outline

The remaining contents of the Thesis are organized as follows:

- Chapter 2 presents the mathematical formulation of the FVM and it covers almost every aspect of the discretization method. Explains the governing equations, the unstructured grids nomenclature and their characterization. An extensive list of the implemented FV schemes is presented, including the boundary conditions and pressure-velocity coupling algorithms. The chapter concludes with an exposition of the different TVD approaches on unstructured grids.

- Chapter 3 contains the results obtained in the unstructured grids framework and is divided into two parts: the first part is a verification process of the unstructured grid methodology by using two important benchmarks from the literature; the second part has a comparison study between two TVD approaches from the literature; and a third method proposed by the author. These methods are tested for four cases with a known analytical solution.

- Chapter 4 presents all the mathematical aspects from both relative and absolute error estimators used in this Thesis. Also the different adaptive algorithm for these two types of error estimators are explained and discussed.

- Chapter 5 contains the obtained results in the adaptive grids framework and it is divided into three parts: the first part has main results obtained with a relative error estimator and its associated adaptive algorithm; the second part contains the first results with an absolute error estimator and using an adaptive algorithm from the literature to study the different aspects of it; and the third part contains the obtained results using the same error estimator but with a proposed adaptive algorithm.

- Chapter 6 summarizes the Thesis with their major conclusions and suggestions for future research in the covered topics.

Chapter 2

Finite Volume Method on Unstructured Grids

2.1 Governing Equations

The objective of this section is to show the different governing equations solved in the SOL code. These equations will be introduced in a form without considering any discretization approximation.

The governing equations of the Fluid Mechanics or the so-called Navier-Stokes are a set of a partial differential equations that express the mass and momentum conservation in the domain. They are defined by the following continuity and momentum equations, respectively:

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

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = \rho \mathbf{g} - \nabla \left(p + \frac{2}{3} \mu \nabla \cdot \mathbf{u} \right) + \nabla \cdot \left[\mu \nabla \mathbf{u} + \mu \nabla^T \mathbf{u} \right] \tag{2.2}$$

where ρ is the fluid density, $\mathbf{u}$ is the velocity vector, t is the time, $\mathbf{g}$ is the gravity acceleration vector, p is the flow pressure and μ is the dynamic viscosity. In the case of compressible fluid flows, an extra partial differential equation is required to close the system, which represents the energy conservation:

$$\begin{aligned} \frac{\partial (\rho C_v T)}{\partial t} + \nabla \cdot (\rho C_v T \mathbf{u}) =& \rho \mathbf{g}.\mathbf{u} - \nabla \cdot (p \mathbf{u}) - \nabla \cdot \left(\frac{2}{3} \mu (\nabla \cdot \mathbf{u}) \mathbf{u} \right) + \\ &+ \nabla \cdot \left[\mu (\nabla \mathbf{u} + \nabla^T \mathbf{u}).\mathbf{u} \right] + \nabla \cdot (\lambda \nabla T) + \rho Q \end{aligned} \tag{2.3}$$

where C_v is the heat capacity at constant volume, T is the fluid temperature, λ is the heat conduction coefficient and Q is the volumetric heat source. To close the system of equations the equation of state for the ideal gas $p = \rho RT$ is required, where R is the universal gas constant.

In this Thesis, the flow will be considered to be isothermal and incompressible, which means that T and ρ are constant. Due to the incompressible constraint, the continuity equation (2.1) becomes, after simplification:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.4}$$

where the time derivative vanishes and the density is not affected by the divergence operator. By applying the continuity constraint and neglecting the gravity force term, the momentum equation (2.2) becomes:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \nabla \cdot (\nu \nabla \mathbf{u}) - \frac{1}{\rho} \nabla p \tag{2.5}$$

For convenience, the pressure term includes the density. Source terms can be added to the right-hand side of the momentum equation (2.5), this approach will be important in the verification process of the SOL code in order to cancel certain terms of the Navier-Stokes equations with the method of manufactured solutions (MMS), please see Eça et al. (2007) for an detailed explanation on this method.

With equations (2.4) and (2.5), the fluid variables $\mathbf{u}$ and p can be determined by using a pressure-velocity coupling algorithm. Three of these algorithms were programmed in this Thesis, the explicit fractional-step, the SIMPLE and the PISO algorithms. Each of these algorithms will be explained in the section 2.4.

The above equations can be written in a generalized transport equation which is defined by:

$$\frac{\partial (\rho \phi)}{\partial t} + \nabla \cdot (\rho \phi \mathbf{u}) = \nabla \cdot (\Gamma_\phi \nabla \phi) + S_\phi \tag{2.6}$$

where ϕ may stand for one of the $\mathbf{u}$ components or different scalars, Γ_ϕ is the scalar diffusivity and S_ϕ is the transport equation source term. The equation (2.6) can be written in an integral form, which is required for the application of the Finite-Volume method (FVM). This consists in the volume integration of the equation in the three dimensional (3D) control volume or computational cell Ω:

$$\underbrace{\int_\Omega \frac{\partial (\rho \phi)}{\partial t} dV}_{\text{temporal term}} + \underbrace{\int_\Omega \nabla \cdot (\rho \phi \mathbf{u}) dV}_{\text{convective term}} = \underbrace{\int_\Omega \nabla \cdot (\Gamma_\phi \nabla \phi) dV}_{\text{diffusive term}} + \underbrace{\int_\Omega S_\phi dV}_{\text{source term}} \tag{2.7}$$

which is followed by the application of the Gauss or Divergence theorem (that is part of the fundamental Stokes theorem for the vectorial calculus) to the convective and diffusive terms:

$$\underbrace{\int_{\Omega} \frac{\partial(\rho\phi)}{\partial t} dV}_{\text{temporal term}} + \underbrace{\int_{\partial\Omega} \rho\phi \mathbf{u}.d\mathbf{S}}_{\text{convective term}} = \underbrace{\int_{\partial\Omega} \rho\Gamma_{\phi} \nabla\phi.d\mathbf{S}}_{\text{diffusive term}} + \underbrace{\int_{\Omega} S_{\phi} dV}_{\text{source term}} \tag{2.8}$$

where $\partial\Omega$ is a closed boundary of the control volume Ω and $d\mathbf{S}$ represents an infinitesimal surface element vector aligned with the outward normal of $\partial\Omega$. Please note that from the Gauss or Divergence theorem the following identities are known:

$$\int_{\Omega} \nabla \cdot \mathbf{u} dV = \int_{\partial\Omega} \mathbf{u}.d\mathbf{S} \tag{2.9}$$

$$\int_{\Omega} \nabla\phi dV = \int_{\partial\Omega} \phi d\mathbf{S} \tag{2.10}$$

This treatment is still an exact mathematical operation and it is required for the FVM discretization. Typically the surface fluxes from the convective and diffusive terms are computed for specified FV schemes, which will be explained in the subsection 2.3.

In the case of the momentum equation (2.5), the application of the volume integration and the Stokes Theorem results in the following expression:

$$\underbrace{\int_{\Omega} \frac{\partial \mathbf{u}}{\partial t} dV}_{\text{temporal term}} + \underbrace{\int_{\partial\Omega} \mathbf{u}\mathbf{u}.d\mathbf{S}}_{\text{convective term}} = \underbrace{\int_{\partial\Omega} \nu\nabla\mathbf{u}.d\mathbf{S}}_{\text{diffusive term}} + \underbrace{\int_{\Omega} -\frac{1}{\rho}\nabla p \, dV}_{\text{pressure term}} \tag{2.11}$$

where the pressure term requires the computation of the cell centered pressure gradient, which can be considered as a third FV scheme in a CFD code. Since several approaches are available in the literature to compute this quantity and they will be covered in subsection 2.3.5.

The Poisson and advection equations may be often considered for verification of the programmed FV schemes. In the case of diffusive schemes, a Poisson equation can be used with the following form:

$$\int_{\partial\Omega} \nabla\phi.d\mathbf{S} = \int_{\Omega} S_{\phi} dV \tag{2.12}$$

where S_{ϕ} is the source term and it is equal to the Laplacian of the analytic solution ϕ. This is required for some analytical solutions where $\nabla^2\phi$ is not null.

In the case of the convective scheme, this type of verification is done with a scalar transport equation with a fixed constant velocity **U**:

$$\int_{\partial\Omega} \phi \mathbf{U}.d\mathbf{S} - \int_{\partial\Omega} \Gamma_\phi \nabla\phi.d\mathbf{S} = \int_{\Omega} S_\phi dV \tag{2.13}$$

where the source term S_ϕ, once again, is required when the convective and diffusive terms do not cancel each other.

2.2 Unstructured Grids

The spatial discretization of the computational domain requires a grid or mesh formed by 3D control volumes or cells, that do not intersect with each other and fill completely the domain. In this Thesis, the unstructured grid approach was chosen for the two main reasons:

Nowadays, CFD codes deal with 3D curved geometries, since it is a demand of the industrial applications where regular geometries are not a valid approximation. As a consequence of this, the unstructured grids approach is adopted. The majority of the codes uses tetrahedral cells when constructing the grid, which is simplest cell type that can be used in complex geometries. The boundary surfaces can be well represented by a set of connected triangles and a tetrahedral grid can be created from these surfaces. Some of these codes can have polyhedral cells with arbitrary topology and with a variable number of faces and vertices. The advantage of these type of cells is that the higher number of neighbor cells, increases the convergence speed of numerical solution when compared to the case of tetrahedral grids, see Juretic and Gosman (2008) for details.

The second reason is related with the successive adaptive grid refinements which will create interfaces between the cells with different refinement levels will create inaccuracy issues due to the lower grid quality. As a consequence, these interfaces require a special treatment when computing the face's fluxes with the FVM in order to maintain a second order accuracy. To avoid projection methods, the adaptive grid will be treated as an unstructured grid with arbitrary topology by considering multiple planar faces in the adaptive grid interfaces. By using suitable FV schemes for unstructured grid, high accuracy in the numerical solution can be achieved, as it will be seen in Chapter 4.

In the case of unstructured grids with arbitrary topology, each cell P is a quasi-convex[1] polyhedral, delimited by a closed boundary ∂P which consists in a set of faces

[1]Polyhedral grid generators cannot guarantee that all the cells are complete convex.

$\mathcal{F}(P)$ or $\{S_1, S_2, ..., S_F\}$. The number of faces of each cell P is arbitrary and variable within the grid. A face f is a quasi-planar[2] polygon with arbitrary orientation and delimited by the consecutive line segments or edges connecting a set of vertices in a well defined order. Finally, a vertex is a point in the 3D physical space and it is defined by its coordinates.

An example of a polyhedral cell P and a set of vertices from the cell's face f are represented in the figure 2.1.

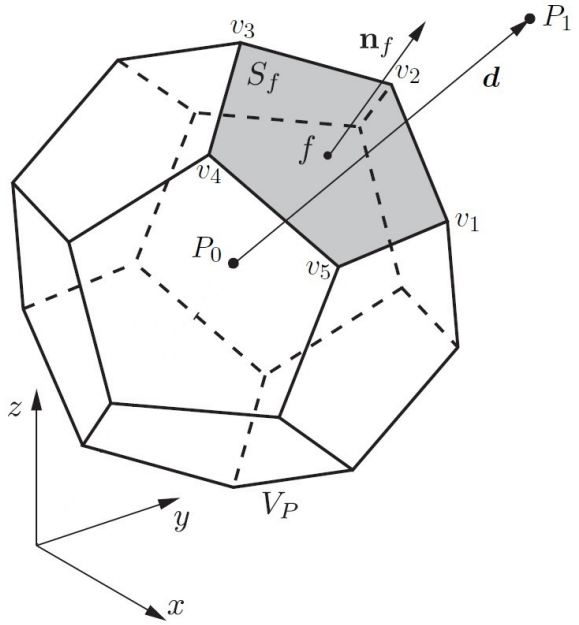


Figure 2.1: Example of a 3D polyhedral cell and a set of vertices from the cell's face f - edited from Tukovik and Jasak (2012).

The cell faces in a grid can be classified as internal faces, which are between two cells or as boundary faces that coincide with the boundaries of the computational domain. The internal faces establish the connectives between cells and, as a consequence, the surface intersection of any two neighbor cells is a single face and each internal face belongs to only two cells. The internal face normal $\mathbf{n}$ points outwards from the cell with the lower label or number. In the case of boundary faces, they belong to only one cell. The boundary face normal points outwards from its cell and consequently, from the computational domain.

The computational points are located at the geometric center of each cell. This corresponds to the so-called "collocated" or "non-staggered" arrangement. Figure 2.2

[2]Polyhedral cells are obtained by agglomeration of different tetrahedral cells and sometimes is not possible to obtain complete planar faces.

provides a sketch of the storage scheme and a example of a 2D unstructured grid. To increase the stability of the pressure-velocity coupling, the convective fluxes U_f are also stored at each face's centroid; this issue is of great import to the overall stability of the numerical method and it will be further developed in section 2.4.

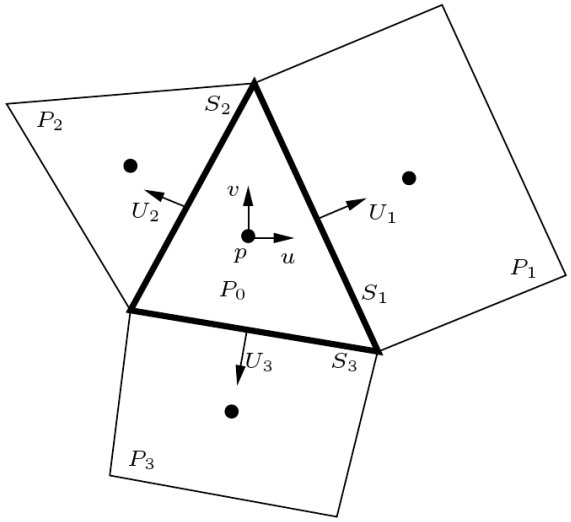


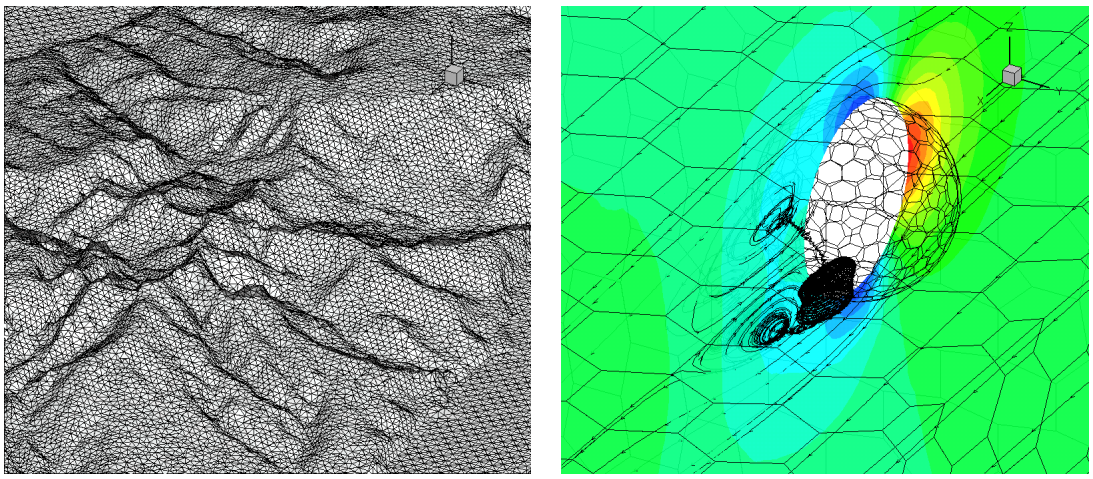
Figure 2.2: Example of a 2D polyhedral cell and computational variables location - Magalhães (2011).

The SOL code has the capability of importing polyhedral grids from both OpenFOAM and STAR-CCM+®. This is done by reading two ASCII files: a vertex file which has all the vertices coordinates in the 3D space with the respective number or label, and a cell file that contains the number of faces and a list of vertices for each face. This list of vertices must be in such order that makes possible to create a regular polygon and to define the edges of the face.

Different routines were programmed to detect the cells neighbors, to enumerate the grid faces and classify them as internal or boundary faces, since this information is not available from the two ASCII files. These routines are based in the following principles: two neighbor cells have at least three common vertices that will form a internal face, the other remaining faces, that belong to only one cell, will be classified as boundary faces. Figure 2.3 shows examples of grids generated by the author, which were imported with these routines: subfigure 2.3(a) shows a tetrahedral grid used to solve the environment flow in the Madeira island and subfigure 2.3(b) shows the polyhedral grid used to solve the flow around a sphere for $Re = 200$.

After reading the ASCII files of the unstructured grid or creating it with a pro-

grammed routine[3], it is required for the FV discretization the computation of different geometrical properties of the mesh elements: the face area S, normal $\mathbf{n}$ and centroid $\mathbf{f}$; the cell volume V_P and centroid $\mathbf{P}$.



(a) Imported tethaedral grid example. (b) Imported polyhedral grid example.

Figure 2.3: Two examples of 3D imported grids.

Certain quantities in the unstructured grids nomenclature share some parallelism with the other quantities from the uniform grids, for example the reference length or cell dimension. One simple way to do this is to compute the cubic root of the cell volume:

$$\Delta = (V_P)^{1/3} \tag{2.14}$$

where Δ is normally used as the spatial filter size for LES models on unstructured grids, which makes a parallel with the one used for non-uniform grids $\Delta = max(\Delta_x, \Delta_y, \Delta_z)$. Although, a more suitable indicator is used to compute the reference length of a 3D cell, the hydraulic diameter h:

$$h_P = \frac{6V_P}{\sum_{f \in \mathcal{F}(P)} ||\mathbf{S}_f||} \tag{2.15}$$

where it is required the summation of all the faces areas. It can be deduced that the hydraulic diameter h (equation (2.15)) reverts to the length size if a uniform hexahedron is used in the formula. For the case of a 2D cell, the hydraulic diameter h

[3]Cartesian and non-uniform grids are created this way in the code.

formula must be modified to:

$$h_P = \frac{4V_P}{\sum_{f \in \mathcal{F}(P)} ||\mathbf{S}_f||} \tag{2.16}$$

It is quite often used in the unstructured grid framework as the cell reference length, specially when studying the numerical error decay with the grid refinement. For this reason, it will be used in this Thesis.

2.3 Finite Volume Discretization

2.3.1 Main Assumptions

The majority of the numerical schemes used in this Thesis have second order accuracy. This means that the truncation error decays with the squared value of the grid size or hydraulic diameter h. In these cases, the profile of the computational variable ϕ inside a cell P will be linear and each cell can have a different linear profile. The value ϕ of a point with coordinates $\mathbf{x}$, which is inside of a cell P, can be computed with the following formula:

$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{P}).(\nabla\phi)_P \tag{2.17}$$

where ϕ is the value of the computational variable in the cell geometric center, $\mathbf{P}$ is the coordinates vector of the cell geometric center and $\nabla\phi$ is the cell centered gradient. The only term in the equation (2.17) that is not easily available is the vectorial quantity $(\nabla\phi)_P$, that is computed with a gradient scheme, which will be developed in subsection 2.3.5. This vector is an important quantity in the unstructured grids framework and it will be required by the convective and diffusive schemes. It is assumed that $(\nabla\phi)_P$ is already computed and available in other subsections.

The second order assumption in the FVM is a consequence from the application of the Gauss-Legendre quadrature to the different terms of the governing equations in the integral form. For example, the source term of the transport equation (2.7) can be approximated by using only one Gauss point, which is a second-order quadrature:

$$\int_P S_\phi dV = S_\phi(\mathbf{P})V_P \tag{2.18}$$

With this formula it is only required to compute the source value at the cell centroid and the correspondent cell volume. This will be the way that the source terms are

treated by the SOL code.[4]

In the case of the terms with surface integrals, since the cell surface ∂P is a set of its own faces, the surface integral can become a sum of faces integrals. Afterwards, the Gauss-Legendre quadrature can be applied by using one Gauss point at the face centroid $\mathbf{f}$. In the case of the convective term, these considerations results in:

$$\int_{\partial P} \rho\phi\mathbf{u}.d\mathbf{S} = \sum_{f\in\mathcal{F}(P)} \int_f \rho\phi\mathbf{u}.d\mathbf{S} = \sum_{f\in\mathcal{F}(P)} \rho\phi_f\mathbf{u}_f.\mathbf{S}_f = \sum_{f\in\mathcal{F}(P)} \rho\phi_f U_f \tag{2.19}$$

where $\mathcal{F}(P)$ is a set that contains all the cell faces, $\mathbf{S}_f$ is a vector that has magnitude equal to the face area $A(f)$, is normal to the face and points outside of the region defined by the cell. The convective flux or conservative velocity U_f is defined as the dot product between the velocity vector $\mathbf{u}_f$ and the surface face vector $\mathbf{S}_f$. The convective scheme consists in a formula to compute the computational variable ϕ_f value at the face centroid from the adjacent cells, the different types of convective schemes will be explained in the subsection 2.3.2.

For the case of the diffusive term, the decomposition of the surface integral and the application of the second-order quadrature results in:

$$\int_{\partial P} \Gamma_\phi\nabla\phi.d\mathbf{S} = \sum_{f\in\mathcal{F}(P)} \int_f \Gamma_\phi\nabla\phi.d\mathbf{S} = \sum_{f\in\mathcal{F}(P)} (\Gamma_\phi)_f(\nabla\phi)_f.\mathbf{S}_f \tag{2.20}$$

where the diffusive scheme is an algorithm to compute the gradient of the dependent variable from the available information of the surrounding cells (the different diffusive schemes will be explained in the subsection 2.3.3). The quantity $(\Gamma_\phi)_f$ is computed with one of the convective schemes.

2.3.2 Convective Schemes

As it was seen previously, the convective schemes are used to interpolate the dependent variable ϕ at the face centroid from the surrounding cells. The first family of convective schemes can be defined by the following equation:

$$\phi_f = (1-\eta)\phi_{P_0} + \eta\phi_{P_1} \tag{2.21}$$

where η is a blending factor that will be used to estimate ϕ_f; the pair ϕ_{P_0} and ϕ_{P_1} are the values of the dependent variable at the cells P_0 and P_1, respectively.

[4]Except when high-order numerical schemes are being used.

The simplest method to compute the ϕ_f value is the upwind differencing scheme (UDS) (the name comes from its origin in the finite differences method). A interpolation can also be made rather than a differentiation, due to the fact that the FVM uses the integral form of the governing equations. The concept of the UDS is based on the principle that the dependent value ϕ is being transported by the conservative velocity U_f and this is done by choosing between the ϕ_{P_0} or ϕ_{P_1} values according to the U_f signal. The UDS is defined by the following expression:

$$\phi_f = \begin{cases} \phi_{P_0} & \Leftarrow \quad U_f \geq 0 \\ \phi_{P_1} & \Leftarrow \quad U_f < 0 \end{cases} \tag{2.22}$$

and this corresponds to use the following formula to the blending factor:

$$\eta_{UDS} = \begin{cases} 0 & \Leftarrow \quad U_f \geq 0 \\ 1 & \Leftarrow \quad U_f < 0 \end{cases} \tag{2.23}$$

Because the UDS forces ϕ to be constant in the upstream cell, the convective scheme is first-order accurate and guarantees boundedness of the solution. However, this scheme suffers from a false diffusion problem, which results in a general numerical inaccuracy.

Another significant factor is that the UDS is the only convective scheme that can be used implicitly in the transport matrix, it is the only convective scheme that can guarantee that the main diagonal of the transport matrix is always dominant. In the case of other convective schemes, a deferred correction algorithm is required which will be explained in the subsection 2.4.

The next convective schemes do not suffer from this accuracy problem, since they use the values of ϕ from both cells. One simple approach is to use an arithmetic average (AVG):

$$\phi_f = \frac{\phi_{P_0} + \phi_{P_1}}{2} \tag{2.24}$$

which results in a blending factor of $\eta_{AVG} = 0.5$. The following scheme does not use any geometrical information, so the interpolation point location is different from the face centroid $\mathbf{f}$, which is the intended location. As a consequence, the AVG convective scheme will not show second-order accuracy on non-uniform grids. The advantage of this scheme is that it conserves the kinetic energy, making it a robust method in the computation of LES models on unstructured grids as it has been shown by Mahesh et al. (2004). To solve the accuracy problem on non-uniform meshes, it is required to make a linear interpolation (the LIN convective scheme). This is done by computing

the intersection point $\mathbf{z}_f$ between the surface plane that contains the cell face f with the following equation:

$$\mathbf{z}_f = \mathbf{P}_0 + \eta_{LIN}(\mathbf{P_1} - \mathbf{P_0}) = \mathbf{P}_0 + \eta_{LIN}\mathbf{d} \tag{2.25}$$

where $\mathbf{d}$ is the distance vector between the cells P_0 and P_1 centroids and the blending factor η_{LIN} is computed with the projection of the $(\mathbf{f} - \mathbf{P_0})$ and $\mathbf{d}$ vectors to the face normal space and these projections results in the following formula:

$$\eta_{LIN} = \frac{\frac{(\mathbf{f} - \mathbf{P_0}).\mathbf{S}_f}{\mathbf{S}_f.\mathbf{S}_f}}{\frac{(\mathbf{P_1} - \mathbf{P_0}).\mathbf{S}_f}{\mathbf{S}_f.\mathbf{S}_f}} = \frac{(\mathbf{f} - \mathbf{P_0}).\mathbf{S}_f}{(\mathbf{P_1} - \mathbf{P_0}).\mathbf{S}_f} \tag{2.26}$$

The location of the different interpolation points from the convective schemes can be seen in the figure 2.4, for the case of an adaptive grid. From the observation of this figure, it can been concluded that on adaptive grids, the interpolation points can be located far from the face centroid $\mathbf{f}$, where it is required the computation of the ϕ_f value.

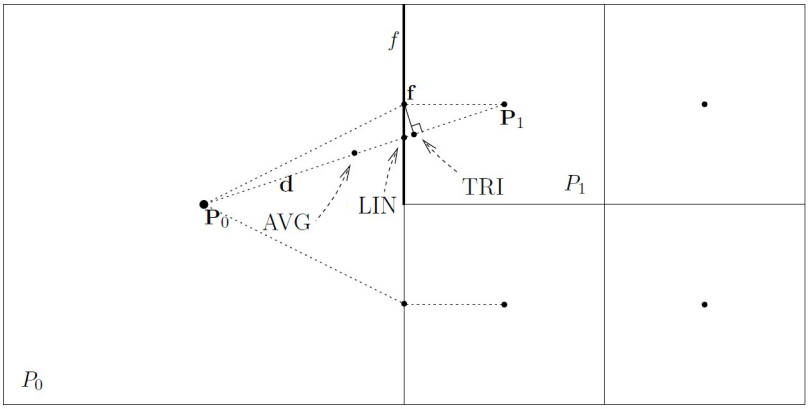


Figure 2.4: Location of the interpolation points for different convective schemes - edited from Magalhães (2011).

It is possible to use a triangular like interpolation (TRI) by using the closest point to the face centroid $\mathbf{f}$ from the line segment defined by the $\mathbf{d}$ vector. This is achieved by projecting the $(\mathbf{f} - \mathbf{P_0})$ to the 1D space defined by the $\mathbf{d}$ vector, resulting in the following formula to the blending factor:

$$\eta_{TRI} = \frac{(\mathbf{f} - \mathbf{P_0}).\mathbf{d}}{\mathbf{d}.\mathbf{d}} \tag{2.27}$$

This concludes the presentation of the first family of FV schemes that do not use any information of the grid quality.

One of the first grid quality parameters that can be derived is the skewness or eccentricity factor[5]. It basically, measures the deviation between the distance **d** vector and the face centroid, which was already seen that will provoke an accuracy error in the convective scheme. An example of how is the skewness factor computed can be seen in figure 2.5, where $\sigma(f)$ is non-dimensional factor proposed by Magalhães (2011) to compute the skewness or eccentricity factor and is defined by:

$$\sigma(f) = \frac{1}{2}\frac{\|\mathbf{f} - \mathbf{z}_f\|}{\sqrt{\|\mathbf{S}_f\|}} \tag{2.28}$$

where the $\frac{1}{2}$ value is used to limit the $\sigma(f)$ in a closed interval $[0, 1]$. To increase the accuracy of the convective scheme when $\sigma(f)$ is different from zero on unstructured grids, several approaches can be used which will be explained in the following paragraphs.

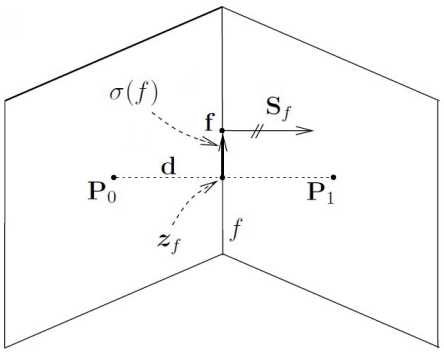


Figure 2.5: Skewness or eccentricity factor representative example - Magalhães (2011).

A different group of convective schemes can be created by adding a correction factor that uses the information provided by the cell centered gradient $\nabla\phi$. This group is constructed by applying the following equation:

$$\phi_f = (1-\eta)\phi_{P_0} + \eta\phi_{P_1} + \underbrace{(1-\eta)(\mathbf{f}-\mathbf{P}_0).(\nabla\phi)_{P_0} + \eta(\mathbf{f}-\mathbf{P}_1).(\nabla\phi)_{P_1}}_{\text{correction factor}} \tag{2.29}$$

by applying the blending factor η_{UDS} and η_{AVG} in the previous equation (2.29), the convective schemes UDS2 and AVG2 can be obtained, respectively.

[5]Authors use one of these terminologies.

The UDS2 convective scheme corresponds to the second-order upwind scheme for unstructured grids which is available in commercial CFD codes and has been studied previously by Kobayashi et al. (1998). This scheme is basically a second-order interpolation from the upstream cell and the second-order accuracy can be achieved for the majority of the unstructured grids. Also it has better stability properties than other convective schemes when using the deffered correction approach.

The AVG2 convective scheme is used to solve problems where the conservation of the kinetic energy is important since it is an upgrade from the AVG scheme.

Magalhães (2011) also applied the equation (2.29) to create a LIN2 and TRI2. The problem with this extension is that it does not solve the skewness grid quality issue since the interpolation point still lies in line segment defined by distance vector **d**.

Some authors (for example Juretic (2004) and de Villers (2006)) applied a correction term to the LIN convective scheme which is directly proportional to the skewness or eccentricity factor. By adding this correction term to equation (2.21) it will result in the following equation:

$$\phi_f = (1-\eta)\phi_{P_0} + \eta\phi_{P_1} + \underbrace{(\mathbf{f} - \mathbf{P}_0 - \eta\mathbf{d}).(\overline{\nabla\phi})_f}_{\text{skewness correction}} \tag{2.30}$$

where the skewness vector is written in the form $(\mathbf{f} - \mathbf{P}_0 - \eta\mathbf{d})$. By making use of both η_{LIN} and η_{TRI} blending factors will result in the LIN-SK and TRI-SK convection schemes, respectively. The $(\overline{\nabla\phi})_f$ vector is the interpolated cell centered gradient from the cells P_0 and P_1 available values.

The overline operator is used in quantities that are considered to be constant inside of each cell like the gradient variables or the cell integrated variables, meaning that the interpolation scheme cannot use correction factors. Typically, the AVG is used in these cases since it gives more robustness to the overall numerical algorithm. Some authors use the LIN or a weighted volume interpolation for these types of quantities, but since in this Thesis adaptive grids are used, the AVG was selected for numerical stability.[6]

There are other two types of convective schemes groups the projection or ghost points ones and the reconstruction ones. The explanation of these type of schemes will be done in the subsections 2.3.4 and 2.3.7.

[6]In the SOL code, it is possible to select other types of interpolation schemes for the overlined quantities.

2.3.3 Diffusive Schemes

As it was seen previously, the diffusive scheme consists in the computation of the dependent variable gradient $\nabla\phi$ at the face centroid $\mathbf{f}$, from the available information of the adjacent cells.

To construct a two points diffusive scheme a finite central differences like formula is applied:

$$(\nabla\phi)_f = (\phi_{P_1} - \phi_{P_0})\frac{\mathbf{d}}{\mathbf{d}.\mathbf{d}} \tag{2.31}$$

where information about the face normal is not used in this equation. Normally, in the unstructured grids framework this diffusive scheme is not used in the literature[7]. It is more common to use a two point diffusive scheme that contains information of the face normal represented by the following equation:

$$(\nabla\phi)_f = (\phi_{P_1} - \phi_{P_0})\frac{\mathbf{S}_f}{\mathbf{S}_f.\mathbf{d}} \tag{2.32}$$

which has second-order accuracy for unstructured grids with small deviations. This diffusive scheme is called central differences scheme (CDS) and it is part of the first family of FVM already presented in the convective scheme subsection 2.3.2, which do not use any type of correction related with the grid quality.

The angle formed between the distance vector $\mathbf{d}$ and the face normal $\mathbf{S}_f$ is used as a grid quality parameter. The warp angle measures the orthogonal departure of the diffusive scheme, which it is directly proportional to an inaccuracy problem and it can be computed by the following formula:

$$\alpha(f) = \arccos\left(\frac{\mathbf{d}.\mathbf{S}_f}{\|\mathbf{d}\|\|\mathbf{S}_f\|}\right) \tag{2.33}$$

also a representative example of the warp angle can be seen in the figure 2.6.

Jasak (1996) in his work proposed a family of diffusive schemes that have an extra non-orthogonal correction term that depends of the warp angle and of the cell centered gradient from both cells. This family of schemes is defined by the following equation:

$$(\nabla\phi)_f = (\phi_{P_1} - \phi_{P_0})\mathbf{p} - \underbrace{\left[(\overline{\nabla\phi})_f - (\mathbf{d}.(\overline{\nabla\phi})_f)\mathbf{p}\right]}_{\text{warp angle correction}} \tag{2.34}$$

where the vector $\mathbf{p}$ is obtained by decomposition of the face surface vector $\mathbf{S}_f$ and the

[7]The diffusive scheme suffers severally from accuracy problems when small deviations exist in the unstructured grid, as concluded by Magalhães (2011).

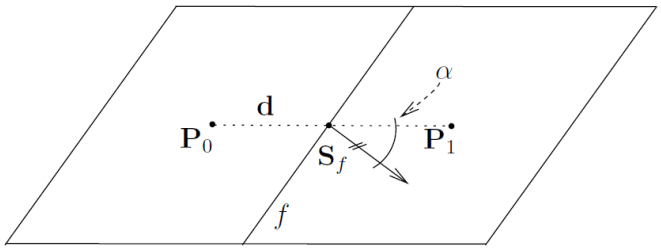


Figure 2.6: Warp angle representative example - Magalhães (2011).

$(\overline{\nabla\phi})_f$ is interpolated from the cell centered gradient from the cells P_0 and P_1. As it was discussed in the previous subsection 2.3.2, the overlined quantities in this Thesis are interpolated with the AVG scheme.

With the equation (2.34), Jasak (1996) studies three diffusive schemes with non-orthogonal or warp angle correction by using three different values for $\mathbf{p}$: the minimum correction, the orthogonal correction and the over relaxed or tangential correction. The expressions for vector $\mathbf{p}$ for each of the presented correction schemes are, respectively, in the equations (2.35), (2.36) and (2.37).

$$\mathbf{p}_{MC} = \frac{\mathbf{d}}{\mathbf{d}.\mathbf{d}} \tag{2.35}$$

$$\mathbf{p}_{OC} = \frac{\mathbf{S}_f}{\|\mathbf{S}_f\|\|\mathbf{d}\|} \tag{2.36}$$

$$\mathbf{p}_{TC} = \frac{\mathbf{S}_f}{\mathbf{S}_f.\mathbf{d}} \tag{2.37}$$

Jasak (1996) compares the convergence history of the three diffusive schemes with the deferred correction method where he concluded that the over-relaxed or tangential correction had the better convergence properties. Magalhães (2011) shows in his work that the tangential correction (TC) scheme is more accurate than the other two non-orthogonal correction schemes. The FV commercial codes and the majority of works in the unstructured grids framework also use this scheme to deal with the accuracy problem caused by the warp angle.

The TC diffusive scheme in association with the UDS2, AVG2, LIN-SK and TRI-SK convective schemes form a family of FV schemes that have a correction terms directly proportional to grid quality parameters.

An example of the face surface vector $\mathbf{S}_f$ decomposition for the TC scheme is shown in figure 2.7. Where the vector $\mathbf{\Delta}$ is parallel to the distance vector $\mathbf{d}$, the vector $\mathbf{t}$ is

parallel to the plane defined by the face f and the following equations are valid:

$$\mathbf{t} = \mathbf{S}_f - \boldsymbol{\Delta} \tag{2.38}$$

$$\boldsymbol{\Delta} = \frac{\mathbf{S}_f.\mathbf{S}_f}{\mathbf{S}_f.\mathbf{d}}\mathbf{d} = (\mathbf{p}_{TC}.\mathbf{S}_f)\mathbf{d} \tag{2.39}$$

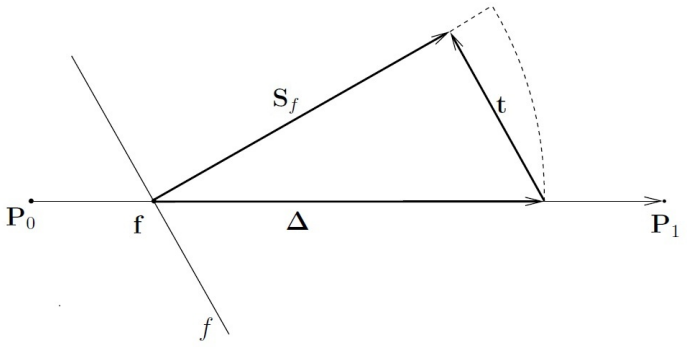


Figure 2.7: Example of the face surface vector $\mathbf{S}_f$ decomposition for the TC scheme - edited from Magalhães (2011).

2.3.4 Ghost Points - Convective and Diffusive Schemes

A problem with the previous diffusive schemes is that they only considered the warp angle of the face when computing the face gradient disregarding the skewness deviation. As a consequence, the face gradient is not computed at the face center with these schemes.

A third family of diffusive and convective schemes can be created by taking into account both warp angle and skewness grid quality parameters, proposed initially by Ferziger and Peric (1999) to to be used as a diffusive scheme and extended for the convective one on adaptive grids by Ham et al. (2002).

First to apply these FV schemes, it is necessary to project the cell centroids $\mathbf{P}_0$ and $\mathbf{P}_1$ to a line which is parallel to the face normal $\mathbf{S}_f$ and that passes through the face centroid $\mathbf{f}$. These two projected points P_0' and P_1' are called ghost points. The figure 2.8 shows a geometrical example of this projection scheme and the points location.

The computation of the ghost points can be done by projecting the distance vector between the cell centroid and the face centroid to a one dimensional space defined by the face normal. As a result of these projections, the distance betyween the cell centroid and the respective ghost point is as small as possible. The location of the

ghost points can be computed with the following equations:

$$\mathbf{P}'_0 = \mathbf{f} - \frac{[(\mathbf{f} - \mathbf{P}_0).\mathbf{S}_f]\,\mathbf{S}_f}{\mathbf{S}_f.\mathbf{S}_f} \tag{2.40}$$

$$\mathbf{P}'_1 = \mathbf{f} - \frac{[(\mathbf{f} - \mathbf{P}_1).\mathbf{S}_f]\,\mathbf{S}_f}{\mathbf{S}_f.\mathbf{S}_f} \tag{2.41}$$

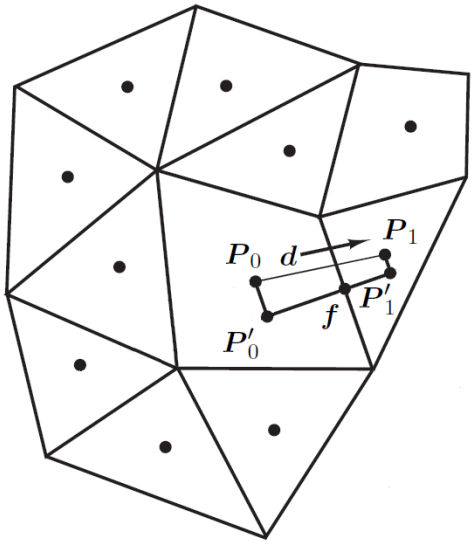


Figure 2.8: Example of the ghost points schemes and respective points location.

To compute the computational values in the ghost points, the cell centered gradient is required, which assumes a linear variation of the computational variable inside the cell. These values can be computed by:

$$\phi_{P'_0} = \phi_{P_0} + (\mathbf{P}'_0 - \mathbf{P}_0).(\nabla\phi)_{P_0} \tag{2.42}$$

$$\phi_{P'_1} = \phi_{P_1} + (\mathbf{P}'_1 - \mathbf{P}_1).(\nabla\phi)_{P_1} \tag{2.43}$$

From the interpolated values is possible to apply the convective and diffusive schemes of this family or category. In the case of the convective scheme, the approach is to apply a formula similar to the LIN scheme:

$$\phi_f = \eta\phi_{P'_0} + (1-\eta)\phi_{P'_1}, \qquad \eta = \frac{(\mathbf{f} - \mathbf{P}'_0).\mathbf{S}_f}{(\mathbf{P}'_1 - \mathbf{P}'_0).\mathbf{S}_f} \tag{2.44}$$

and in the case of the diffusive scheme, a formula similar to the CDS scheme is used:

$$(\nabla\phi)_f = (\phi_{P_1'} - \phi_{P_0'})\frac{\mathbf{S}_f}{\mathbf{S}_f.\mathbf{d}'} \tag{2.45}$$

where $\mathbf{d}'$ is the distance vector between the two ghost points P_0' and P_1', which is defined by the following equation:

$$\mathbf{d}' = \mathbf{P}_1' - \mathbf{P}_0' \tag{2.46}$$

The ghost points family of schemes seems to be more accurate than the previous family or set of schemes that depend on the grid quality, for small non-orthogonal grids. In the case of severe grids with high warp angles, it is possible that the ghost points will lie outside of its own cell. When this happens, it is necessary to use of a search algorithm to check in each cell the ghost points are inside. On another hand, using linear interpolation to compute the ghost points values may not be accurate enough for this degree of grid quality.

For these reasons, the second family of FV schemes is more robust than this one and can be applied to a larger range of different unstructured grids. As a consequence, schemes like the LIN-SK and TC are more popular in different CFD codes and in the literature of the unstructured grids FV schemes.

2.3.5 Cell Centered Gradient Schemes

The cell centered gradient of the computational variable is a required quantity to improve the accuracy of some FV schemes or to compute the gradient terms of the governing equations, like the pressure gradient. It can be computed with either the Gauss or the Weighted Least Squares (WLS) method, which have different characteristics and will be explained in this subsection.

In the literature, it is also possible to apply a combination of both methods in order to obtain bounded gradients. To achieve this, different approaches have been proposed by Barth and Frederickson (1990); Venkatakrishnan (1994, 1995) and Park et al. (2010).

Gauss Method

The Gauss method consists in applying the Gauss or Divergence theorem to the variable gradient $\nabla\phi$ and transform it into a summation of each cell faces fluxes $\phi_f\,\mathbf{S}_f$.

The derivation of the Gauss method can be obtained by the following sequence:

$$\int_P \nabla\phi dV = \int_{\partial P} \phi d\mathbf{S} \Rightarrow (\nabla\phi)_P = \frac{1}{V_P} \sum_{f\in\mathcal{F}(P)} \phi_f \, \mathbf{S}_f \tag{2.47}$$

where the second order assumption inside the cell and the Gauss-Legendre quadrature were considered.

For the computation of the quantity ϕ_f, the already presented convective schemes can be used, except for the Upwind based schemes. If the convective schemes have terms that depends on the cell centered gradient, a deferred correction approach can be used by considering an initial guess. This process converges in the maximum of four or five iterations for skewed meshes as was reported in the work of Ni et al. (2007).

One advantage of the Gauss method in the computation of the cell centered gradient is that it depends on information about the computational grid, like the cell volume, the faces area and the faces centroids distribution. As disadvantage, since the variable gradient consist in two or three values[8] from the available of the cell neighbors, it can be considered an unweighted average and cause an underestimation of the gradient. As a consequence, it can lead to an inaccuracy problems when compared with WLS method.

Weighted Least Squares Method

The Weighted Least Squares (WLS) method is an alternative approach to compute the cell centered gradient. It consists in a polynomial fit with the discrete values of the surroundings cells, considering the arbitrary centroids location. Other geometric data from the computational grid like the cell volume, face centroids and normals are ignored with this method.

The polynomial to be fit in this case will be a linear profile with the constant equal to the value of the cell centroid ϕ_P and centered at the cell centroid $\mathbf{P}$, which can be written with the following formulation for a 3D case:

$$\phi(x,y,z) = \phi_P + (x-x_P)\left(\frac{\partial\phi}{\partial x}\right)_P + (y-y_P)\left(\frac{\partial\phi}{\partial y}\right)_P + (z-z_P)\left(\frac{\partial\phi}{\partial z}\right)_P \tag{2.48}$$

where for a 2D case the z component is not considered, each derivative is one component of cell gradient $(\nabla\phi)_P$ vector to be computed.

The application of the WLS on unstructured grids is well documented in the works

[8]It depends if you are considering a two or three dimensional case.

of Muzaferija and Gosman (1997) and Kobayashi et al. (1999). The generalized WLS method for any type of polynomial will be explained in the next subsection 2.3.6.

The application of the polynomial (equation (2.48)) fit in the cell neighbors ϕ values with the WLS method, results in the solving of the linear system $\mathbf{Ag} = \mathbf{h}$, where $\mathbf{g}$ is the gradient vector to be computed. The matrix $\mathbf{A}$ and the vector $\mathbf{h}$ components are defined by:

$$a_{mn} = \sum_{f \in \mathcal{F}(P)} w_f d_{f,m} d_{f,n} \tag{2.49}$$

$$h_m = \sum_{f \in \mathcal{F}(P)} \left(\phi_{P_k} - \phi_P\right) w_f d_{f,m} \tag{2.50}$$

where m and n denote the Cartesian components, $d_{f,m}$ the component m of the distance vector between the cell P and the neighbor cell P_f and w_f is the weight of cell P_f, obtained as the inverse of the square of the distance vector module between the cells P and P_f second order approximation is being considered. The weighted function is important to give a higher contribution to the closest cells in the polynomial fit procedure and avoid inaccuracy issues caused by using a too much cells in the regression.

With this method, the computation of the cell centered gradient requires the inversion of the matrix square $\mathbf{A}$, which can be done analytically with the Cramer's rule. The matrix $\mathbf{A}$ has size 2×2 for 2D cases and 3×3 for 3D cases.

2.3.6 Generalized Weighted Least Squares Method

In this subsection, the generalized WLS method is presented, which is used to compute the polynomial fit of an arbitrary set of points with different values ϕ. This algorithm has the versatility required to deal with different sets of point data which are assembled with either the first or second cell neighbors by face or by vertex. Also, polynomial with different orders can be used with this algorithm depending on the final goal. The only requirement to build the minimization algorithm is that the number of polynomial coefficients to be **computed** is lower or equal to the number of available cell values from the established computational stencil[9].

This method is very important in the context of this Thesis since it is applied in different cases. It is applied in the computation of the cell centered gradient (already presented in subsection 2.3.5), in the computation of the face WLS second order

[9] A computational stencil is a set of cells that contributes in the computation of a certain scheme or quantity.

schemes which will be presented in the next subsection 2.3.7. Finally, these regressions technique are also applied in the computation of the relative and absolute error estimators that are explained in chapter 4.

By considering a base polynomial for the computational variable ϕ centered in the cell P:

$$\phi(x, y, z\,; P) = \phi_P + \frac{\partial \phi}{\partial x}(x - x_P) + \frac{\partial \phi}{\partial y}(y - y_P) + \frac{\partial \phi}{\partial z}(z - z_P) + ... \tag{2.51}$$

where the previous equation has a similar form to a Taylor series expansion. It can be generalized in the following form:

$$Y = b_0 + \sum_{k=1}^{n} b_k X_k \tag{2.52}$$

where y_i is a linear combination of n variables X_k plus a constant and the parameters b are polynomial coefficients to be computed. Using the polynomial approach of the Taylor series, the b parameters would be related to the gradient, Hessian, third and higher derivatives of the ϕ variable. A component y_i of the observation vector $\mathbf{y}$ is defined by:

$$y_i = (b_0 + b_1 X_{1,i} + b_2 X_{2,i} + \ldots + b_n X_{n,i}) + e_i \tag{2.53}$$

Considering all cell values from the stencil, an overdetermined system of equations is formed, given by $\mathbf{yW} = \mathbf{XbW} + \mathbf{eW}$, where $\mathbf{y}$ is a vector with dimension $m \times 1$ containing the observations of y, $\mathbf{W}$ is a diagonal matrix with dimension $m \times m$ containing the square root of the weight function for each observation, $\mathbf{X}$ is a matrix $m \times (n+1)$ containing the values of the several explaining variables for each observation, $\mathbf{b}$ is a vector with dimension $n + 1$ with the polynomial coefficients to be computed and $\mathbf{e}$ is the vector with the correlation error for each of the observations:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & X_{1,1} & X_{2,1} & ... & X_{n,1} \\ 1 & X_{1,2} & X_{2,2} & ... & X_{n,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{1,m} & X_{2,m} & ... & X_{n,m} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{2.54}$$

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} \sqrt{w_{1,1}} & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \sqrt{w_{m,m}} \end{bmatrix} \tag{2.55}$$

The correlation error is defined as the difference between the observed $\mathbf{y}$ and the estimated value $\hat{\mathbf{y}} = \mathbf{Xb}$:

$$e_i = y_i - \hat{y}_i \tag{2.56}$$

where the solution vector $\mathbf{b}$ is obtained by the WLS method, which corresponds to the minimum of the following sum:

$$\min_{\mathbf{b}} \sum_{i=0}^{m} (w_{i,i}) e_i^2 \quad \Leftrightarrow \quad \min_{\mathbf{b}} (\mathbf{W}^T \mathbf{e}^T)(\mathbf{We}) \tag{2.57}$$

The least squares solution of this system is obtained by forming the normal system:

$$\mathbf{X'WWy} = \mathbf{X'WWXb} + \mathbf{X'WWe} \tag{2.58}$$

or (by neglecting the residuals):

$$\mathbf{b} = \underbrace{(\mathbf{X'W}^2\mathbf{X})^{-1}\mathbf{X'W}}_{(\mathbf{XW})^{-1}_{left}} \mathbf{Wy} \tag{2.59}$$

where $(\mathbf{XW})^{-1}_{left}$ is the left inverse of the rectangular matrix $\mathbf{XW}$. If no weight function is being used, the previous equation (2.59) becomes:

$$\mathbf{b} = \underbrace{(\mathbf{X'X})^{-1}\mathbf{X'}}_{\mathbf{X}^{-1}_{left}} \mathbf{y} \tag{2.60}$$

where $\mathbf{X}^{-1}_{left}$ is the left inverse of the rectangular matrix $\mathbf{X}$.

Essentially, the application of the WLS method consists in the computation of the matrix $(\mathbf{XW})^{-1}_{left}$. This is done by applying the Cramer's rule when the polynomial is linear. In the case of higher order polynomials or a higher number of cells are being used, this computation is done with the single value decomposition (SVD) method, due to bad conditioning problems of the matrix.

Since the left inverse matrix only dependents of geometric data from the computational grid, this result can be saved for the future iterations. It is only necessary to perform the matrix vector product $(\mathbf{XW})^{-1}_{left}\mathbf{Wy}$ to obtain the polynomial coefficients

in each iteration.

2.3.7 Face WLS - Convective and Diffusive Schemes

A fourth family or category of convective and diffusive schemes for arbitrary unstructured grids can be created with the WLS method. This consists in building the following linear polynomial centered in the face centroid coordinates, with the discrete information of the surrounding cells:

$$\phi(\mathbf{x}) = \phi_f + (\nabla\phi)_f.(\mathbf{x} - \mathbf{f}) \tag{2.61}$$

where both convective and diffusive values (ϕ_f and $(\nabla\phi)_f$) are available in the same regression. This way, it allows to deal with the several orthogonality and skewness deviations that can exist on the unstructured grids. For this type of regressions, the cells that have one of the face vertices are included. Figure 2.9 shows examples of different computational stencils used in these regressions. In each example, the regression k is centered in the face S_k and each cell of the stencil is marked with the respective number k.

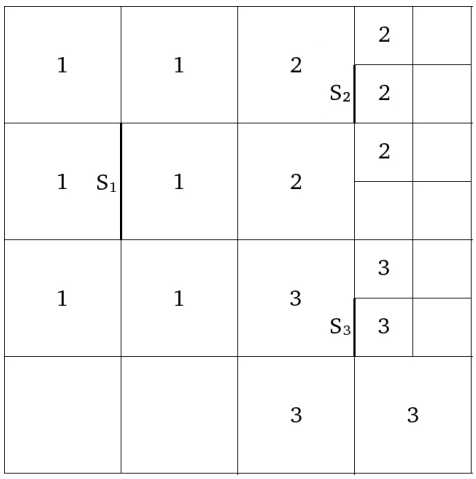


Figure 2.9: Possible Stencils used in the WLS Schemes.

The computational cost of solving the WLS problem for each face is significant but it is only required to be solved once. The matrix values can be saved and used in future iterations. All the least squares regressions use a weight function w_P, given by the inverse square of the distance:

$$w_{P_k} = \frac{1}{||\mathbf{P}_k - \mathbf{f}||^2} \tag{2.62}$$

where $\mathbf{P}_k$ is the cell P centroid and $\mathbf{f}$ is the face centroid, which are the coordinates of the regression reference.

To the author knowledge, there are not present in the literature many applications with this type of reconstruction schemes centered at the face centroid. In subsection 5.3.1, it will be shown that this type of schemes can have better a accuracy than the other ones when applied to adaptive grids.

2.3.8 Time Scheme - Temporal Term

The computation of the temporal term in the governing equations requires the selection of a suitable of time scheme. For the application of the time scheme, it is convenient to write the governing equations in the following formula:

$$\frac{\partial \phi}{\partial t} = F(\phi, t) \tag{2.63}$$

where the term $F(\phi, t)$ is the summation of the all terms of the governing equation except for the temporal term, for the designated ϕ values and time value t:

$$F(\phi, t) = C(\phi, t) + D(\phi, t) + PG(\phi, t) + S(\phi, t) \tag{2.64}$$

and it is also necessary to define an initial condition for the first time level t_0, when using the temporal schemes:

$$\phi(t_0) = \phi_0 \tag{2.65}$$

From the combination of the equations (2.63) and (2.64), different time schemes can be obtained that were implemented in the SOL code. The simplest one is the explicit Euler scheme, which is a first order time accurate scheme:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = F(\phi^n, t^n) \tag{2.66}$$

that uses the fractional-step algorithm to solve the pressure-velocity coupling, which is explained in subsection 2.4.1. The first order Euler implicit version is written with the following formula:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = F(\phi^{n+1}, t^{n+1}) \tag{2.67}$$

where the right term is now computed in the future time level $n + 1$, and due to this it is necessary to solve the pressure-velocity coupling with the SIMPLE or PISO algorithms, which are explained in subsection 2.4.2 and 2.4.3.

Due to the non linearity of the momentum equations it is necessary to use space iterations in each time step. This requirement comes from the convective term dependency of the face velocity U_f that initially is only available at the time level n. In each space iteration, the matrix of the momentum equations will change and the face velocity U_f will converge to an approximated value at time level $n+1$. Also the pressure gradient term will change from the time level n to $n+1$ during these space iterations.

A second order time accurate implicit method can be obtained by considering the correspondent regressive finite difference formula:

$$\frac{3\phi^{n+1} - 4\phi^n + \phi^{n-1}}{2\Delta t} = F(\phi^{n+1}, t^{n+1}) \tag{2.68}$$

This scheme requires the additional storage for the variable ϕ at the time level $n-1$.

Another implicit second order time scheme can be obtained by separating the right side term in a explicit and implicit contribution:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \frac{1}{2}[F(\phi^n, t^n) + F(\phi^{n+1}, t^{n+1})] \tag{2.69}$$

where this time scheme is known as the Crank-Nicolson scheme. There is also other explicit time schemes implemented in the SOL code like the second order Runge-Kutta scheme:

$$\phi_*^{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{2} F(\phi^n, t^n) \tag{2.70}$$

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{2} F(\phi_*^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}) \tag{2.71}$$

where this temporal term is computed by using an auxiliary stage level. Also the fourth order Runge-Kutta explicit scheme was implemented:

$$\phi_*^{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{2} F(\phi^n, t^n) \tag{2.72}$$

$$\phi_{**}^{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{2} F(\phi_*^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}) \tag{2.73}$$

$$\phi_*^{n+1} = \phi^n + \Delta t F(\phi_{**}^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}) \tag{2.74}$$

$$\begin{aligned} \phi^{n+1} = \phi^n + \frac{\Delta t}{6}[F(\phi^n, t^n) + 2F(\phi_*^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}) + \\ + 2F(\phi_{**}^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}) + F(\phi_*^{n+1}, t^{n+1})] \end{aligned} \tag{2.75}$$

where for this scheme three auxiliary stage levels are required. As explained previously, the governing equations are solved with the fractional-step method when using explicit time schemes. When using explicit methods to solve the momentum equations, the time step value should be limited due to two numerical stability constrains: the Courant-Friedrichs-Lewy condition and the diffusive stability criterion.

The Courant-Friedrichs-Lewy condition or Courant number is a limit imposed when the convective term is discretized explicitly, which is defined by:

$$\frac{||\mathbf{u}||\Delta t}{h} < \lambda_c \tag{2.76}$$

where Δt is the time step, h is the hydraulic diameter and λ_c is the Courant number, that should be lower than 1 for the case of the explicit Euler time scheme. Equation (2.76) results in the following time step constraint:

$$\Delta t < \frac{h\lambda_c}{||\mathbf{u}||} \tag{2.77}$$

that must be satisfied to avoid numerical instability. The diffusive stability criterion is required when using a explicit method to compute the diffusive term, which is defined by:

$$\frac{\nu\Delta t}{h^2} < \lambda_d \tag{2.78}$$

where λ_d is the diffusive stability criterion, that must be lower than $1/2$ for the explicit Euler time scheme. This results in the following time step constraint:

$$\Delta t < \frac{h^2\lambda_d}{\nu} \tag{2.79}$$

The limit value of the Courant number and of the diffusive stability criterion depends of the time scheme, the spatial discretization used and other characteristics from the governing equation. These limits are necessary but not sufficient to guarantee the numerical solution convergence. In Pereira and Pereira (2001) several time schemes are studied for a range of different values of λ_c and λ_d and limits for theses values are established for one dimensional and linear equations. In the case of this Thesis, these values should be lower since the Navier-Stokes equations are non linear and the grid is unstructured.

For the case of implicit temporal schemes they are unconditionally stable[10], which means that they are not constrained by stability . This means that implicit methods

[10]This is proven for linear problems.

do not have these stability limits. For transient problems there is a restriction for the time step size that comes from the truncation error of the finite difference formula of the time scheme. If the time step is too large there will be an accuracy problem in the solution, so there is a restriction in the time step for implicit methods.

Due to the formulation of the fractional-step method (for details read subsection 2.4.1), it is possible to use a specific time scheme for the convective term and a different one for the diffusive term. So, the diffusive term can be treated implicitly, eliminating the time step restriction from the diffusive stability criterion. Since this term is linear, the system of equations needs to be solved only one time, at each time step.

The assembly of this hybrid temporal schemes has the advantage of eliminating the time step restriction from the λ_d, which depends of the quadratic degree of h (the grid reference length), unlike the Courant number λ_c that depends linearly of h. For these reasons the diffusive stability criterion can be very restrict for high refined grids and adaptive grids that have high intervals of length scales (multi-scale problems), like the ones showed in this Thesis.

In the SOL code, the hybrid time scheme is implemented by using the second or the fourth Runge-Kutta scheme to compute explicitly the convective term and the Crank-Nicolson scheme to compute implicitly the diffusive term.

2.4 Pressure-Velocity Coupling

The pressure-velocity coupling algorithm and the compatibility between the different numerical schemes affect the overall accuracy of the results, stability and energy conservation proprieties, see Verstappen and Veldman (2003) for details. It is well known that the collocated grid arrangement may induce unphysical oscillatory pressure profiles, when computing incompressible fluid flows.

In this section the different algorithms used for the pressure-velocity coupling are explained. In explicit cases the fractional-step algorithm is used to solve the Navier-Stokes equations and for steady or implicit cases the SIMPLE or the PISO algorithms are used.

Other segregated algorithms are available in the literature like the SIMPLEC and SIMPLER. A full coupled approach is also well documented in the Thesis of Darwish et al. (2009), the idea being to put all the momentum and continuity equations in the same matrix. Some of the terms have to be put in the right hand size of the linear system with deferred correction due to numerical instability issues. The main advantage of this approach is that the number of iterations to solve the system is

independent of the grid size, as it is showed by Darwish et al. (2009).

2.4.1 Fractional-Step

A fractional-step algorithm can be used to solve numerically the continuity (2.4) and the momentum (2.5) equations. The method consists in dividing each of the momentum terms in different stages in each time step. For example, if the explicit Euler temporal scheme is being used, equation (2.5) can be written in the following form:

$$u_P^{n+1} = u_P^n + (C_P + D_P + PG_P)\Delta t \tag{2.80}$$

where C_P, D_P and PG_P[11] are the convective, diffusive and pressure gradient terms, respectively, at cell P. It is possible to split the previous equation in two stages in the following form:

$$u_P^* = u_P^n + (C_P + D_P)\Delta t \tag{2.81}$$

$$u_P^{n+1} = u_P^n + (PG_P)\Delta t \tag{2.82}$$

It will be seen later why is important to have a last stage where is only added the pressure term. Some authors split the first stage (equation (2.81)) into two stages, where each one has a different time scheme but with the same accuracy order. The diffusion term is solved with a Crank-Nicholson implicit scheme and the convection term is solve with a explicit second-order Runge-Kutta. This type of hybrid time schemes was implemented in the SOL code and brings numerical stability to the overall coupling algorithm by freeing the need for a diffusive stability criterion. That is a constraint when explicit schemes are being used to treat the diffusive terms, specially for high refined meshes since it depends on the inverse square of the hydraulic diameter h.

In each time step, the method starts by computing from the previous velocity field $\mathbf{u}^n$ a velocity field that satisfies the momentum equations without considering the pressure gradient contribution:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\nabla \cdot (\mathbf{u}^n \otimes \mathbf{u}^n) + \nabla \cdot (\nu \nabla \mathbf{u}^n) \tag{2.83}$$

where in this example the explicit Euler time scheme is being considered. Afterwards, the predicted $\mathbf{u}^*$ is projected to a space where the continuity is satisfied by adding

[11] In the FD framework the pressure term is represented by P_i but since the letter P is already used for definition of cell, it was choose PG for definition of the pressure term.

implicitly the gradient pressure term:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\nabla p^{n+1} \tag{2.84}$$

By applying the divergence operator in equation (2.84) and recalling that from the continuity equation (2.4), the term $\nabla \cdot \mathbf{u}^{n+1}$ should be forced to zero. The previous equation results in the Poisson equation for the pressure field:

$$\frac{\Delta t}{\rho}(\nabla^2 p^{n+1}) = \nabla \cdot \mathbf{u}^* \tag{2.85}$$

Since the computational variables are located at the cells centroids, the convective fluxes U_f at each face must be also stored to provide the pressure-velocity coupling. By applying to equations (2.83) and (2.85) the volume integral, the Gauss or Divergence theorem and the second-order Gauss-Legendre quadrature will result in the following expressions:

$$\mathbf{u}_P^* = \mathbf{u}_P^n - \frac{\Delta t}{V_P} \sum_{f \in \mathcal{F}(P)} \mathbf{u}_f^n U_f^n + \frac{\Delta t}{V_P} \sum_{f \in \mathcal{F}(P)} \nu_f (\nabla \mathbf{u}^n)_f . \mathbf{S}_f \tag{2.86}$$

$$\frac{\Delta t}{\rho} \sum_{f \in \mathcal{F}(P)} (\nabla p^{n+1})_f . \mathbf{S}_f = \sum_{f \in \mathcal{F}(P)} \mathbf{u}_f^* . \mathbf{S}_f = \sum_{f \in \mathcal{F}(P)} U_f^* \tag{2.87}$$

where in the Poisson equation (2.87) the left term is treated implicitly with a diffusive scheme and the right term is computed explicitly from the computed values of $\mathbf{u}^*$ by equation (2.86).

From the pressure field at the time level $n+1$, the velocities at the face center U_f and at the cell center $\mathbf{u}$ can be updated in order to satisfy the continuity constraint:

$$U_f^{n+1} = U_f^* - \frac{\Delta t}{\rho}(\nabla p^{n+1})_f . \mathbf{S}_f \tag{2.88}$$

$$\mathbf{u}_P^{n+1} = \mathbf{u}_P^* - \frac{\Delta t}{\rho}(\nabla p^{n+1})_P \tag{2.89}$$

and from this point the time can advance an extra level and the fractional-step process starts again from the momentum equation (2.86).

The fractional-step algorithm is mainly used with time explicit schemes, small time steps and in cases where the coupling between these pressure and velocity fields is not required to be strong. The SIMPLE and PISO methods are more robust and exhibit a more stable coupling between these two fields. This method has the advantages to be less demanding in terms of the computational cost required and the Poisson equation

to correct the velocities fields can be assembled only one time, since it only depends on geometric factors from the computational grid.

When using diffusive schemes different from the CDS, a deferred correction may be used here and the values from the pressure field of the previous iteration are used to compute the explicit part. The deferred correction approach will be better explained in subsection 2.4.2.

To increase the robustness of the overall coupling algorithm a stabilization procedure proposed by Mahesh et al. (2004) was implemented in the SOL code and will be explained in the next paragraphs.

Pressure-velocity stabilization procedure

When solving the Poisson equation (2.87) to satisfy the continuity constraint at the time level $n+1$, the face velocity U_f is solved to the machine precision but the same does not happen to the cell velocities $\mathbf{u}$. This is driven by the fact that only the face pressure gradient $(\nabla p)_f$ is considered in the Poisson equation. So, it is possible that cell velocities $\mathbf{u}$ do not satisfy completely the continuity equation, since the cell centered pressure gradient $(\nabla p)_P$ is computed from the resultant pressure field.

In their work, Mahesh et al. (2004), computed the contribution of the pressure gradient to the kinetic energy equation for staggered and collocated grid arrangement and concluded that, in the case of collocated grids, the relation between pressure gradient could be as energy conserving as possible by minimizing the following expression:

$$\sum_{f \in \mathcal{F}(P)} [(\nabla p)_P . \mathbf{S}_f - (\nabla p)_f . \mathbf{S}_f] \tag{2.90}$$

where the least squares method should be used to compute the vector $(\nabla p)_P$ from the already computed values $(\nabla p)_f$ at each cell faces. This scheme is more robust but it requires additional computational work. This procedure was also presented earlier by Benhamadouche et al. (2002). The stabilization contribution to the overall algorithm increases when skewed cells are being used, which can happen in the unstructured grids framework. Mahesh et al. (2004) used this procedure to compute LES flows in complex geometries like a gas-turbine combustor and showed that this procedure was necessary to solve the Taylor problem for $Re = 10^9$ without any kinetic energy build up.

From this approach, Magalhães (2011) proposed a simplified equation to reduce the computational work from the least squares method when the grid quality is sufficient.

The idea is to compute a weighted average of the cell centered pressure gradient $(\nabla p)_P$ from the values of the cell faces:

$$(\nabla p)_P = \frac{\sum_{f \in \mathcal{F}(P)} [(\nabla p)_f . \mathbf{S}_f (\mathbf{S}_f / \|\mathbf{S}_f\|)]}{\sum_{f \in \mathcal{F}(P)} \|\mathbf{S}_f\|} \tag{2.91}$$

which is a good approximation when the computational grid has high quality and does not require the solving of a local WLS problem for each cell.

Fractional-step algorithm summary

The fractional-step algorithm can be summarized by these following steps:

1. Define the pressure p, velocity field $\mathbf{u}$ and the face velocity U_f at the time level n. The face velocity U_f is important for the definition of the convective term.
2. Compute a prediction of the velocity field $\mathbf{u}^*$ from the momentum equation (2.86).
3. Interpolate the predicted velocity field $\mathbf{u}^*$ to the faces center and compute the continuity of each cell (last term of the equation (2.87)).
4. Solve implicitly the Poisson equation (2.87) to obtain the pressure p at the time level $n + 1$. This matrix only needs to be assembled one time since the matrix values just depend of geometric factors from the computational grid.
5. Correct the face velocity U_f at the time level $n + 1$ with equation (2.88).
6. If the stabilization option is not being used, compute the cell centered pressure gradient $(\nabla p)_P$ from the pressure field p. If the stabilization option is being used compute $(\nabla p)_P$ from the LS minimization of equation (2.90) or directly from the equation (2.91).
7. The velocity field $\mathbf{u}$ is corrected with equation (2.89). Increase time $t = t + \Delta t$ and $n = n + 1$, go back to point 1.

2.4.2 SIMPLE

The semi-implicit method for pressure-linked equations (SIMPLE) algorithm, presented by Patankar and Spalding (1972), is used for the pressure velocity coupling of steady cases and of unsteady cases that require a implicit discretization of the temporal term. For simplification proposes, the SIMPLE algorithm will be presented for the steady state equations.

The algorithm starts by computing an approximate velocity field $\mathbf{u}^*$, which satisfies the momentum equations using the values from the previous iteration n. The steady equation is solved implicitly and linearization of the convection contribution is required:

$$\sum_{f=1}^{F} U_f^n \mathbf{u}_f^* - \nu \sum_{f=1}^{F} (\nabla \mathbf{u}^*)_f . \mathbf{S}_f = -\frac{V_P}{\rho} (\nabla p^n)_P \tag{2.92}$$

where U_f^n is the face velocity defined by $\mathbf{u}_f^n . \mathbf{S}_f$, $\mathbf{S}_f$ is the face surface vector defined by $S_f \mathbf{n}_f$ and $\mathbf{n}_f$ is the normal unit vector of the face f. A system of linear equations is assembled in this form:

$$\frac{1}{\alpha_u} a_p \mathbf{u}_P^* + \sum_{l=1}^{F} a_l \mathbf{u}_l^* = -\frac{V_P}{\rho} (\nabla p^n)_P + \frac{1 - \alpha_u}{\alpha_u} a_p \mathbf{u}_P^n \tag{2.93}$$

being α_u the under relaxation factor for the momentum equations, a_p are the main diagonal matrix values from the momentum system and a_l are the other non-zero values of the momentum matrix, which represent the contribution of the cell P neighbors to the momentum system. The under relaxation factor α_u is required when solving a non-linear system to bring stability to the overall algorithm by improving the main diagonal dominance. It consists in dividing the first term with a_p by α_u and by adding an explicit term at the right hand side of the equation (2.93), which is the last term of this equation.

With the predicted velocity values $\mathbf{u}^*$, interpolation to the face is required to compute the face velocity U_f^*. Because a collocated grid system is being used, the pressure field can have a checkboard like distribution that is caused by the incompatibility between the pressure gradients at the face centers and at the cell centers. To achieve the required compatibility, a Rhie-Chow interpolation (Rhie and Chow (1983)) is used when computing the face velocity U_f^*. This interpolation requires the values of cell centered pressure gradient used in the momentum equations and is defined by the following equation:

$$U_f^* = \mathbf{u}_f^* . \mathbf{S}_f - \overline{\left[\frac{\alpha_u V_P}{\rho \, a_p} \right]}_f \left[(\nabla p^n)_f - (\overline{\nabla p^n})_f \right] . \mathbf{S}_f \tag{2.94}$$

where $\mathbf{u}_f^*$ is a prediction of the face centered velocity[12] which is computed with a convective scheme and using the velocities from the adjcent cells. This convective

[12]The velocity field at the face centroid $\mathbf{u}_f^*$ is only conservative, after the continuity equation being satisfied, which occurs in the end of the SIMPLE iteration.

scheme cannot be of the family of the Upwind schemes since it will cause divergence of the algorithm. The over-lined values are obtained by a special interpolation from the cells P_0 and P_1 values and this type of interpolations was already discussed previously in subsection 2.3.3.

The $\frac{\alpha_u V_P}{\rho\, a_p}$ is a factor that relates the pressure gradient with the velocity field, which results from the momentum equation. The pressure gradient at the face $(\nabla p)_f$ is obtained with the diffusive scheme from the pressure values of the adjacent cells and the $(\overline{\nabla p^n})_f$ quantity is obtained by interpolation of the cell centered pressure gradients $(\nabla p)_{P_0}$ and $(\nabla p)_{P_1}$. The computation of these terms is required in the Rhie-Chow interpolation to bring numerical stability to the SIMPLE algorithm when a collocated grid arrangement is used.

Afterwards, the velocity field must be changed in order to the continuity equation be satisfied. To do this, the objective is to compute a pressure correction p' which projects the velocity field to a conservative one. The assembly of the pressure correction equation, which relates the pressure field with the velocity field, results in the following equation:

$$\sum_{f=1}^{F} \overline{\left[\frac{\alpha_u V_P}{\rho\, a_p}\right]}_f (\nabla p')_f . \mathbf{S}_f = \sum_{f=1}^{F} U_f^* \tag{2.95}$$

after solving the Poisson equation (2.95). The velocity values are corrected with the new p' values, therefore satisfying the continuity equation (2.4). The conservative face velocities are computed by:

$$U_f^{n+1} = U_f^* - \overline{\left[\frac{\alpha_u V_P}{\rho\, a_p}\right]}_f (\nabla p')_f . \mathbf{S}_f \tag{2.96}$$

and the cell centered velocities are corrected by:

$$\mathbf{u}_P^{n+1} = \mathbf{u}_P^* - \frac{\alpha_u V_P}{\rho\, a_p} (\nabla p')_P \tag{2.97}$$

where the compatibility between the $(\nabla p')_f$ and $(\nabla p')_P$ quantities is ensured by the Rhie-Chow interpolation. And the pressure is updated by:

$$p^{n+1} = p^n + \alpha_p p' \tag{2.98}$$

where in the case of the SIMPLE algorithm an under relaxation factor α_p for the pressure update is required since the coupling is semi-implicit. Typically, the values for the under relaxation are $\alpha_u = 0.8$ and $\alpha_p = 0.2$ but some authors use the values

$\alpha_u = 0.7$ and $\alpha_p = 0.3$, for example Darwish et al. (2009).

From this point, the momentum residuals $\mathbf{R}_M$ are computed by the following equation:

$$\mathbf{R}_M = \frac{a_p \mathbf{u}_P^{n+1} + \sum_{l=1}^{F} a_l \mathbf{u}_l^{n+1} + \frac{V_P}{\rho} \nabla p}{a_p \; max(\mathbf{u})} \tag{2.99}$$

where it is important to use a residual adimensionalization, due to the different cell volumes that occur during the adaptive algorithm, giving them the same residual contribution for each cell. The continuity residual R_C for each cell is defined by:

$$R_C = \sum_{f=1}^{F} U_f^* \tag{2.100}$$

If all residuals 2^{nd} norm are lower than a prescribed value the cycle ends, if not the computation advances to the next iteration, back to equation (2.92). For the majority of the computed cases, this tolerance is equal to 10^{-10}.

Deferred correction

The convective and diffusive schemes used in this work, that correct the grid quality issue, may originate a non positive definite matrices if assembled in a fully implicit manner. Thus, the deferred correction approach Ferziger and Peric (1999) was used to avoid divergence when solving the linear system of equations with these type of schemes.

This consists in computing the contribution of a stable lower order scheme both implicitly and explicitly and the contribution of the required high order scheme explicitly. The idea is to assemble these contributions in the following form when assembling the linear system of equations:

$$C_{low}(\phi^{n+1}) = C_{low}(\phi^n) - C_{high}(\phi^n) \tag{2.101}$$

where C is the contribution of the low or high order scheme, respectively. When the method is converging, the difference of the values between the level $n+1$ and n are negligible, and the contributions $C_{low}(\phi^{n+1})$ and $C_{low}(\phi^n)$ will cancel with each other. At this stage, only the high order scheme is contributing to the FV discretization.

The application of the deferred correction algorithm in the discretized momentum

equation (2.93) results in:

$$\frac{1}{\alpha_u}a_p\mathbf{u}_P^* + \sum_{l=1}^{F} a_l\mathbf{u}_l^* = -\frac{V_P}{\rho}\nabla p^n + \frac{1-\alpha_u}{\alpha_u}a_p\mathbf{u}_P^n + [C_{low}(\mathbf{u}^n) - C_{high}(\mathbf{u}^n)] \qquad (2.102)$$

where the matrix values a_p and a_l are computed from the contribution of the stable low order schemes. As stable schemes, the central differences scheme CDS (equation (2.32)) is used for the diffusion terms, while the convective fluxes are approximated by the first order upwind scheme UDS (equation (2.22)).

Non-orthogonal correction steps

In the SIMPLE and PISO algorithms, when solving the pressure correction equation (2.95) for a diffusive scheme different of the CDS, the application of the deferred correction[13] is required for this Poisson equation. Since the solving the pressure correction equation is the most demanding part of the SIMPLE algorithm, it is required to solve the pressure equation in several correction steps.

For the first correction step, the explicit terms are computed with the pressure correction p' from the previous outer iteration. And for the next correction steps, the values of p' from the previous step are used in the explicit part. Normally, each correction step needs less $25\% - 50\%$ iterations to solve the linear system than from the previous one.

The use of non-orthogonal correction steps in the SIMPLE algorithm is a common practice in commercial codes and in the literature of unstructured grids discretization.

SIMPLE algorithm summary

Here it is summarized the SIMPLE algorithm step-by-step:

1. Define the pressure p, the velocity field $\mathbf{u}$ and the face velocity U_f for the outer iteration n.
2. Assemble the matrix system[14] of the momentum equations (2.93). Since the face velocity U_f can change in each outer iteration, the matrix values can also change. Store the main diagonal values of the matrix a_p and the cell centered gradient pressure $(\nabla p)_P$ in an auxiliary vector.

[13] In the case of the diffusive schemes, the deferred correction can be simplified by decomposing the high order scheme into an implicit and an explicit part. The explicit part is the one that depends in the cell centered gradient.

[14] One, two or three equations if a 1D, 2D or 3D problem is being considered.

3. Solve the momentum linear system of equations and obtain the predicted velocity field $\mathbf{u}^*$.
4. Compute the predicted face velocity U_f^* by applying the Rhie-Chow interpolation equation (2.94). Which depends of the values from the velocity field at the cell center $\mathbf{u}^*$, the pressure p, the cell centered pressure gradient $(\nabla p)_P$ and the matrix main diagonal values a_p.
5. Compute the continuity quantity for each cell and assemble the matrix for the pressure correction (equation (2.95)). The matrix must be assembled in each outer iteration.
6. Solve the pressure correction system and obtain p'. Apply the number of prescribed non-orthogonal correction steps if a diffusive scheme different from the CDS is being used.
7. Correct the face velocity U_f and the cell velocities $\mathbf{u}$, for the outer iteration $n+1$, by applying equations (2.96) and (2.97), respectively.
8. Update the pressure, for the outer iteration $n+1$, with equation (2.98).
9. Compute the momentum residuals $\mathbf{R}_M$ and the continuity residual R_C with equations (2.99) and (2.100), respectively. If they are higher than a prescribed value, advance to the next outer iteration $n = n+1$ and back to step 1. If not, the computation is finished in the case of a steady problem; advance to the next time step $t = t + \Delta t$ and go back to step 1 in the case of a unsteady problem.

2.4.3 PISO

The pressure implicit with split operator (PISO), proposed by Issa (1986), is an algorithm that improves the SIMPLE algorithm by increasing the coupling between the pressure and the velocity fields, in each outer iteration. This is achieved by adding correction steps after solving the pressure correction equation. Due to simplification, only the second correction step will be shown in this subsection.

The face velocity correction $U_f^{'}$ and the cell velocity correction $\mathbf{u}^{'}$ can be computed from the last term of the equations (2.96) and (2.97) of the SIMPLE algorithm.

$$U_f^{'} = -\overline{\left[\frac{\alpha_u V_P}{\rho\, a_p}\right]}_f (\nabla p')_f . \mathbf{S}_f \tag{2.103}$$

$$\mathbf{u}_P^{'} = -\frac{\alpha_u V_P}{\rho\, a_p}(\nabla p')_P \tag{2.104}$$

The problem with the velocity correction field is that it does not satisfy the momentum equations. This can be forced by computing a second prediction of the velocity field $\mathbf{u}^{**}$:

$$\mathbf{u}_P^{**} = \frac{1}{a_p}\left[-\sum_{l=1}^{F} a_l \mathbf{u}_l^{'} + C_{low}(\mathbf{u}^{'}) - C_{high}(\mathbf{u}^{'})\right] \tag{2.105}$$

where the values of the matrix come from the momentum matrix used in the first velocity prediction. Afterwards, a face interpolation is required:

$$U_f^{**} = (\mathbf{u}^{**})_f.\mathbf{S}_f \tag{2.106}$$

and the second pressure correction p'' equation is assembled:

$$\sum_{f=1}^{F} \overline{\left[\frac{\alpha_u V_P}{\rho\, a_p}\right]}_f (\nabla p'')_f.\mathbf{S}_f = \sum_{f=1}^{F} U_f^{**} \tag{2.107}$$

where the matrix of this pressure correction is the same as the one from the previous correction step, so it is not necessary to recompute the matrix values. The right hand side of the system is the continuity of the second predicted velocity field.

Finally, with the second pressure correction p''. The velocity fields at the faces and the cells centers can be updated by:

$$U_f^{n+1} = U_f^{*} + U_f^{**} - \overline{\left[\frac{\alpha_u V_P}{\rho\, a_p}\right]}_f (\nabla p')_f.\mathbf{S}_f - \overline{\left[\frac{\alpha_u V_P}{\rho\, a_p}\right]}_f (\nabla p'')_f.\mathbf{S}_f \tag{2.108}$$

$$\mathbf{u}_P^{n+1} = \mathbf{u}_P^{*} + \mathbf{u}_P^{**} - \frac{\alpha_u V_P}{\rho\, a_p}(\nabla p')_P - \frac{\alpha_u V_P}{\rho\, a_p}(\nabla p'')_P \tag{2.109}$$

and the pressure can be computed by:

$$p^{n+1} = p^n + p' + p'' \tag{2.110}$$

where, in the case of the PISO algorithm, it is not necessary to use a pressure under relaxation factor α_p due to the strong pressure-velocity coupling, even if only two corrections steps are being used. For the case of the velocity under relaxation factor α_u is only required for steady cases to create a matrix with a main diagonal dominant. For unsteady cases, the additional temporal term is sufficient to create this effect.

PISO algorithm summary

In the case of the PISO algorithm, the first six steps are the same of the SIMPLE algorithm (subsection 2.4.2). Due to simplification reasons, these steps will be not listed here and the summary will start from step 7.

7. With the pressure correction p' compute the corresponding cell and face velocities corrections ($U_f^{'}$ and $\mathbf{u}^{'}$) by using the equations (2.103) and (2.104), respectively.
8. Compute explicitly a second prediction for the velocity field $\mathbf{u}^{**}$ with equation (2.105). Interpolate this quantity to the face centroid to obtain U_f^{**}.
9. Compute the cell continuity with U_f^{**} and solve the second pressure correction equation (2.107) to obtain p''. The matrix is the same as the one used in the first pressure correction stage. Once again, apply the number of prescribed non-orthogonal correction steps if a diffusive scheme different from the CDS is being used.
10. Correct the face and cell velocities fields for the outer iteration $n+1$ by applying equations (2.108) and (2.109), respectively.
11. Update the pressure, for the outer iteration $n+1$, with equation (2.110).
12. Compute the momentum residuals $\mathbf{R}_M$ and the continuity residual R_C with equations (2.99) and (2.100), respectively. If they are higher than a prescribed value, advance to the next outer iteration $n = n+1$ and back to step 1. If not, the computation is finished in the case of a steady problem; advance to the next time step $t = t + \Delta t$ and go back to step 1 in the case of a unsteady problem.

2.5 Boundary Conditions

When solving a computational problem in a closed domain, it is necessary to prescribe boundary conditions in all the boundary faces, which are the faces that belong to only one cell. The boundary conditions can have two meanings: one mathematical and another physical. This section is divided in four subsection:

- First, an explanation of the different mathematical boundary conditions (BCs) is done.
- Afterwards, the mathematical BCs treatment for both convective and diffusive terms are shown, where each term is explained in its own subsection.

- Finally, the meaning of physical boundary condition is presented and each of the implemented physical BC are explained in detail.

2.5.1 Mathematical Boundary Condition

The mathematical BC must always be imposed and it is defined by a mathematical formula. The most basic one is the Dirichlet BC, which corresponds to impose a fixed value ϕ_b of the computational variable at the boundary face[15]. Another one is the Neumann BC, that corresponds to impose the gradient along the boundary normal of the computational variable $\left(\frac{\partial \phi}{\partial n}\right)_b = g_b$. A third type of mathematical BC can also be imposed, that is called the Robin BC. It consists in imposing a linear combination between the computational variable value and its gradient, which is translated by the following equation:

$$\phi_b + c\left(\frac{\partial \phi}{\partial n}\right)_b = d \tag{2.111}$$

where c and d are known constants. The Robin BC was not implemented in the SOL since it is not presented in any of the physical boundary condition for incompressible fluid flow. It was mentioned because it takes part of the mathematical BC for second order differential equations.

2.5.2 Convective Term - Boundary Treatment

As it was explained in the subsection 2.3.2, the convective term consists in computing a face value ϕ_f. In the case of the Dirichlet boundary condition, this corresponds to match the face value ϕ_f to the boundary value ϕ_b:

$$\phi_f = \phi_b \tag{2.112}$$

In the case of the Neumann BC, it is necessary to linearly interpolate the face value by using the following equation:

$$\phi_f = \phi_P + \left(\frac{\partial \phi}{\partial n}\right)_b (\mathbf{f}_b - \mathbf{P}).\frac{\mathbf{S}_f}{||\mathbf{S}_f||} \tag{2.113}$$

where $\mathbf{f}_b$ is the centroid coordinates of the boundary face f_b and the index P corresponds to the cell that contains the respective boundary face. In the case of a implicit method,

[15] A function can be a Dirichlet boundary condition. This results that each face centroid has a fixed value that comes from the function value at the face centroid coordinates.

the contribution of ϕ_P term must be added to the matrix and the last term contribution must be added to the right hand side of the system of equations.

The face velocity U_f must be computed from the dot product between the boundary velocity vector and the face area vector $\mathbf{S}_f$.

To compute the cell centered gradient with the Gauss method, the treatment of the boundary condition is the same as for the convective term. In the case of WLS method, the face centroid value ϕ_b is included in the regression and the distance vector $\mathbf{d}$ is computed as the difference between the face center $\mathbf{f}$ and the cell centroid $\mathbf{P}$.

2.5.3 Diffusive Term - Boundary Treatment

For the diffusive term, the boundary treatment requires the knowledge of the following equality:

$$(\nabla\phi)_f.\mathbf{S}_f = \left(\frac{\partial\phi}{\partial n}\right)||\mathbf{S}_f|| \tag{2.114}$$

where for the case of a Neumann BC it is only required to replace the $\left(\frac{\partial\phi}{\partial n}\right)_b$ by the imposed value g_b. For the case of Dirichlet BC it is required to use a finite difference formula for the normal gradient:

$$\left(\frac{\partial\phi}{\partial n}\right)_b = \frac{\phi_b - \phi_P}{d_n} \tag{2.115}$$

where d_n is the distance value between the cell centroid $\mathbf{P}$ and the face centroid $\mathbf{P}$ projected to the space defined by the face normal. This quantity can be computed by the following formula:

$$d_n = (\mathbf{f} - \mathbf{P}).\frac{\mathbf{S}_f}{||\mathbf{S}_f||} \tag{2.116}$$

2.5.4 Physical Boundary Conditions

The physical boundary conditions are related with physical aspects of the fluid flow. For each physical BC, a set of mathematical BC are prescribed for the velocity field and the pressure, in order to a certain physical meaning be satisfied by the flow solution. In the case of the SIMPLE and PISO algorithms, the same pressure BC is applied for both the pressure and the pressure correction, except for the pressure outlet BC. In this subsection, the physical boundary conditions implemented in the SOL code are explained.

Wall

The velocity field near the wall corresponds to the wall's velocity due to the no-slip condition, as a consequence a Dirichlet BC is imposed for the velocity field. The wall does not have velocity in its normal direction[16] so this BC does not contribute to the face velocity U_f. In the case of the pressure, a Neumann BC is imposed with a null gradient.

Inlet

Essentially, the inlet BC consists in imposing a fixed velocity value (Dirichlet BC). In the case of the pressure, a Neumann BC is imposed with a null gradient at domain's boundaries.

Outlet - Outflow

In the case of a outlet or outflow, a Neumann BC with a null gradient is imposed in the velocity field. At each time step or outer iteration, after solving the momentum equations it is required to extrapolated the velocity at the outlet boundary $\mathbf{u}_b$. This can be done by considering the velocity at cell that contains the boundary face:

$$\mathbf{u}_b = \mathbf{u}_P \tag{2.117}$$

which is a first order approximation. It is also possible to extrapolate linearly by considering the cell centered gradient of each velocity component. The face velocity U_f at the outlet boundary face is simply updated by:

$$U_f = \mathbf{u}_b.\mathbf{S}_f \tag{2.118}$$

Afterwards, it is necessary to compute the total mass flux at the inlet and outlet boundary faces, since mass conservation in the computational domain cannot be guarantee by any extrapolation:

$$Q_{in} = \sum_{f \in \partial\Omega_{inlet}} U_f \tag{2.119}$$

$$Q_{out} = \sum_{f \in \partial\Omega_{outlet}} U_f \tag{2.120}$$

[16] Except for moving grid problems, which are not covered in this Thesis.

where, due to the face normal convection, Q_{in} is negative and Q_{out} is positive. These values should cancel each other when the numerical solution converges, this can be forced by either computing a mass flux correction Q_{corr} or a ratio between the inlet and outlet fluxes R_{flux}:

$$Q_{corr} = -(Q_{in} + Q_{out}) \tag{2.121}$$

$$R_{flux} = \frac{-Q_{in}}{Q_{out}} \tag{2.122}$$

and the face velocity at the outlet boundary is updated to ensure mass conservation at the domain. This can be done by considering the mass flux correction Q_{corr}:

$$U_f^* = U_f + Q_{corr}\frac{||\mathbf{S}_f||}{S_{outlet}} \tag{2.123}$$

where S_{outlet} is the total area of the outlet boundary and it is computed by:

$$S_{outlet} = \sum_{f \in \partial\Omega_{outlet}} ||\mathbf{S}_f|| \tag{2.124}$$

Alternately, the face velocity can be updated with the ratio between the inlet and outlet fluxes R_{flux}, by scaling the outlet face velocity:

$$U_f^* = U_f R_{flux} \tag{2.125}$$

both methods work and have similar convergence rates. Since the face velocity at the outlet faces is corrected with the explained procedure, a Neumann BC with null flux for the pressure is imposed at the outlet.

Pressure Outlet

An alternative way to create an open BC is done by prescribing a pressure outlet. This means that the pressure will now have a prescribed value as a Dirichlet BC, which is different from the other physical boundaries. A Neumann BC with null gradient is prescribed for the velocity field.

For the case of the SIMPLE and PISO algorithms, the prescribed pressure correction value is zero since the pressure has a fixed value. In the other physical BC, it is not required to compute a velocity correction since the pressure correction gradient is null. In the case of the pressure outlet BC it is required to compute a pressure gradient[17] with the equation (2.115), in order to correct the face velocity U_f at these

[17] In the case of SIMPLE and PISO algorithms, it is a pressure correction gradient.

boundary faces. No additional treatment is required and the mass conservation in the domain is guaranteed by the pressure equation.

A difference between the pressure outlet and the outlet BC is that at the pressure outlet the pressure is forced to have a constant distribution along the boundary. In the case of the outlet or outflow BC the pressure can have a free distribution along the boundary.

Symmetry

The symmetry BC can be done by imposing Neumann BC with a null gradient to all the quantities. In the case of vectorial quantities it is necessary to imposed a null value to its normal component, since a vector can only have tangential components at the symmetry plane. This treatment is required because the velocity field is a vectorial quantity.

In the SOL code, only boundary planes align with one of the reference axis can be selected for symmetry BC. The process is to check which component of the reference axis is normal to the boundary and impose a Dirichlet BC with a null value for that velocity component. The other ones have a Neumann BC with a null gradient value.

2.6 Solution of the Linear Equations System

When solving the momentum equations and the pressure equation, a linear system of the type $\mathbf{Ax} = \mathbf{b}$ is assembled, where the matrix $\mathbf{A}$ is sparse and only the non-zero values are stored due to the big size of matrix system. Due to its size, iterative methods must be used. In this Thesis, these systems are solved with the biconjugated gradient stabilized method (BIGSTAB) and the Incomplete LU decomposition ILU(0) is used as preconditioner. The SOL code uses the AZTEC library which has these methods already programmed for both single and multi core computations.

There is an option in the code to incorporate all the terms of the TC diffusive scheme in the matrix when building the pressure equation system. As a consequence, it is not required to use the non-ortogonal correction steps for in this case. When the unstructured grid has some degree of deviations, the system can only be solved with the generalized minimal residual method (GMRES), also the number of iterations and computational time is increased with this method. The method proved to be useful in the computation of laminar flows on adaptive grids and the obtained results are presented in the section 5.1 with the fractional-step algorithm. Although, it is not robust enough to be used in all cases of unstructured grids. For example with the SIMPLE

algorithm on a tetrahedral grid, the solver GMRES does not converge the pressure equation of some outer iterations, increasing significantly the computation time. For this reason, the non-orthogonal correction steps were implemented (a common practice in other CFD codes).

For the computation of the WLS regressions, dense matrices must be used. For the cases that require the inversion of a square matrix with size between 2×2 and 4×4, the Cramer's rule is used due to its simplicity and being a direct algorithm. The number of components of this square matrix is equal to the number of coefficients to be determined of the polynomial used in the regression. For other dimensions, the singular value decomposition (SVD) method was chosen because of two reasons: firstly, the Cramer's rule can have inaccuracy issues due to the truncation error when computing each component of the inverse matrix; secondly, these square matrices can have conditioning problems due to the higher order polynomials and the weighted function can create a matrix. Pina (2001) presents different iterative methods to solve WLS problems and concludes that the SVD method is the most robust for a range of different problems.

2.7 TVD on Unstructured Grids

The following section is divided into two parts: firstly, an exposition of the total variation diminishing (TVD) concept and application on structured grids is presented; and secondly, its extension to the unstructured grids framework is explained, including a new method based on the ghost points schemes, that is proposed by the author.

2.7.1 TVD principle on structured grids

The goal of a TVD scheme is to create a second order accurate convective scheme that is free or slightly free of the boundness issue, so a desirable property is monotonicity preserving. In order to achieve this, a numerical discretization must not create new local extremes and the existent ones cannot be accentuated. This latter means that a local maximum cannot increase during each outer iteration or time step and a local minimum cannot decrease.

These properties have an implication in the total variation $TV(\phi)$ of the numerical solution, which can be measured by the following equation:

$$TV(\phi) = \sum_{f} |\phi_{P_0} - \phi_{P_1}| \tag{2.126}$$

where in the summation all the interior faces of the computational grid are being considered and the cells P_0 and P_1 are the two cells that contain the common face f. The monotonicity principle is respected if the global quantity total variation decreases in each outer iteration or time steps $TV(\phi^{n+1}) \leq TV(\phi^n)$, therefore the classification of total variation diminishing or TVD.

The assembly of a TVD scheme is similar to the UDS, it requires the identification of three points in the computational grid where the variable ϕ are located. Two of these points are from the cells P_0 and P_1 that contain the face f. One of this points is identified as downwind point D which is located in the side where the flow direction is pointing and the other point is identified as point C. Finally, the third point is identified with a U and is the other neighbor of the cell C that is located at the opposite side of the cell D. Figure 2.10 shows two graphical examples of the points identification in a TVD scheme is done.

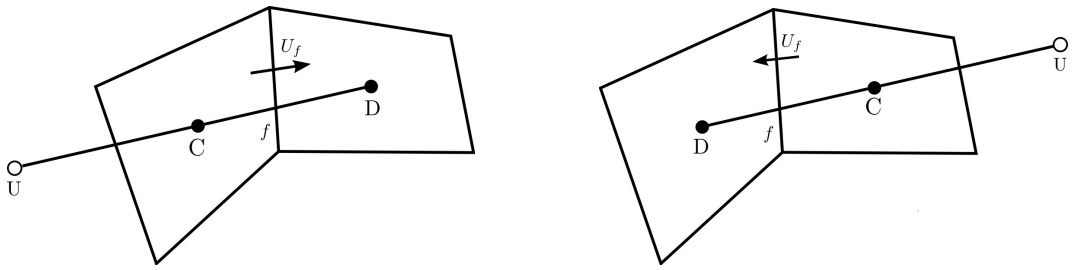


Figure 2.10: Examples of the point identification for the TVD schemes.

A TVD scheme is a combination between a diffusive UDS and an anti-diffusive one, which results in the following form:

$$\phi_f = \underbrace{\phi_C}_{\text{UDS}} + \underbrace{\frac{1}{2}\psi(r)(\phi_D - \phi_C)}_{\text{anti-diffusive}} \tag{2.127}$$

where $\psi(r)$ is a selected flux limiter and r is a factor that measures the ratio between successive gradients and it is defined by:

$$r = \frac{\phi_C - \phi_U}{\phi_D - \phi_C} \tag{2.128}$$

By observing the equation (2.127), it can be concluded that:

- If $\psi = 0$, it reverts to the UD convective scheme.
- If $\psi = 1$, it reverts to the LIN convective scheme.

- If $\psi = r$, it reverts to the second order UD convective scheme (UDS2).

In his work, Sweby (1984) proves that in order to a second order scheme be TVD, it must be bounded by the UDS2 and the LIN schemes for certain values of r. To help the definition of these limits, a Sweby diagram can be used, which is presented in figure 2.11, where the gray regions represents the valid ψ values in function of r in order to a second-order convective scheme have TVD properties. From the observation of this diagram, is possible to conclude that for each values of r, a scheme is TVD.

For example, the UDS2 is TVD if $|r| < 2$ and the LIN scheme is TVD if $|r| > 0.5$. The UDS is always TVD for any value of r but this fact cannot be observable from the Sweby diagram since it is a first order scheme.

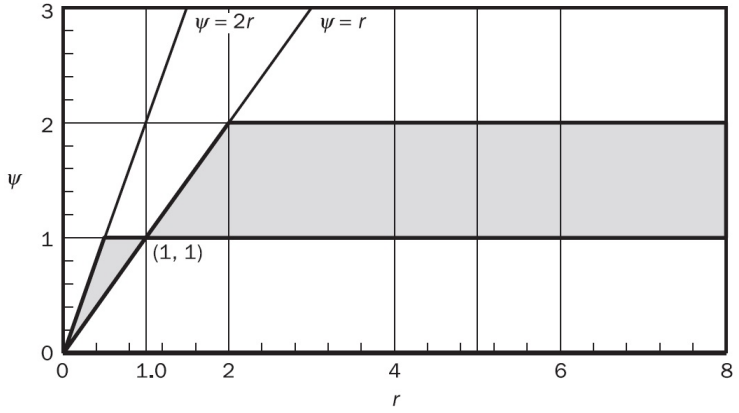


Figure 2.11: Sweby diagram example with the respective TVD regions - from Versteeg and Malalasekera (2007).

The principle of the flux limiter is to construct a function for $\psi(r)$ that appears in the anti-diffusive part of the equation (2.127) and it is bounded by the grey region of the Sweby diagram.

In table 2.1, a list of the programmed and tested flux limiters is shown with the respective references. To compare the different properties of the flux limiters they are plotted in the same Sweby or $r - \psi$ diagram presented in figure 2.12.

It can be observed that all the limiter functions are inside the grey region, passing through point $(1, 1)$ of the Sweby diagram, so they all can be classified as second-order TVD schemes. The SUPERBEE and MINMOD functions trace the upper and lower boundaries of the TVD regions, respectively. The OSHER is a piecewise combination between the MINMOD and the SUPERBEE functions. The MUSCL and Van Albada are continuous functions and more old flux limiters. The UMIST was design to be a symmetrical version of the QUICK limiter, which is not studied in the context of this

Name	Flux Limiter function $\psi(r)$	Reference
SUPERBEE	$max[0, min(2r, 1), min(r, 2)]$	Roe (1985)
MINMOD	$max(min(1, r), 0)$	Harten (1983)
Osher	$max[0, min(2, r)]$	Osher (1983)
MUSCL	$\frac{r + \|r\|}{1 + \|r\|}$	Leer (1979)
Van Albada	$\frac{r + r^2}{1 + r^2}$	Albada et al. (1982)
UMIST	$max[0, min(2r, (1 + 3r)/4, (3 + r)/4, 2)]$	Lien (1993)

Table 2.1: Flux limiter functions list.

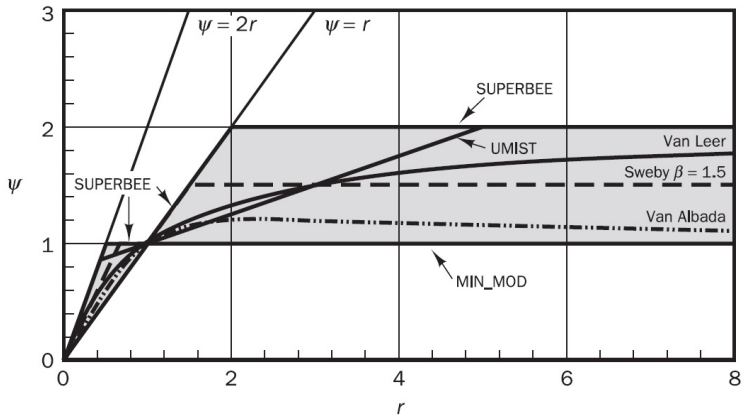


Figure 2.12: Flux limiter functions at the Sweby diagram - from Versteeg and Malalasekera (2007).

Thesis. A flux limiter is a symmetric one if the following condition is verified:

$$\frac{\psi(r)}{r} = \psi(1/r) \tag{2.129}$$

The symmetric property is important to ensure that both forward and backward gradients are treated in the same way, without the need of special programming.

2.7.2 TVD Extension to Unstructured Grids

The extension of the TVD and flux limiters concept on unstructured grids is not evident because it is possible that the regressive point U is not coincident with or even close to a centroid of another neighbor cell. Also, the line formed by the points **C** and **D** can be too far from any computational point. Figure 2.13 shows an example of this problem on unstructured grids.

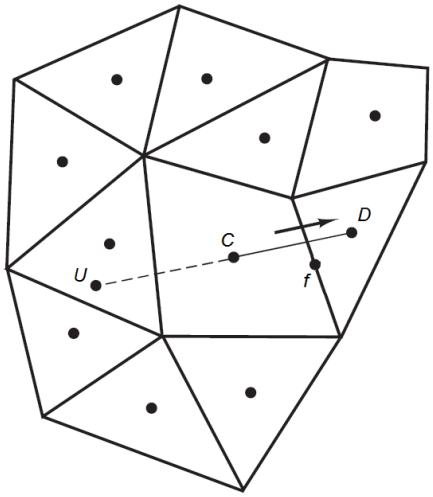


Figure 2.13: TVD point identification on unstructured grids problem example.

Implemented Methods

In the SOL code, the algorithms of Darwish and Moukalled (2003) and Li et al. (2008) were implemented in order to compare them with the proposed one in this Thesis. These methods were chosen because they showed the most accurate results from the literature survey and, for text simplification, they will be called Darwish and Li method, respectively.

The Darwish method consists at writing the r factor equation (2.128) in a form

that does not require the computational value of the point U:

$$r = \frac{\phi_C - \phi_U}{\phi_D - \phi_C} = \frac{(\phi_D - \phi_U) - (\phi_D - \phi_C)}{\phi_D - \phi_C} = \underbrace{\frac{2(\mathbf{D} - \mathbf{C}).(\nabla\phi)_C}{\phi_D - \phi_C} - 1}_{\text{Darwish } r \text{ factor}} \tag{2.130}$$

where $\mathbf{D}$ and $\mathbf{C}$ are the centroid of the cells D and C, respectively. In the previous equation (2.130), it was assumed that:

$$(\phi_D - \phi_U) = (\mathbf{D} - \mathbf{U}).(\nabla\phi)_C = 2(\mathbf{D} - \mathbf{C}).(\nabla\phi)_C \tag{2.131}$$

One advantage of the Darwish method is that the cell centered gradient is an already available quantity on unstructured grid based code. So, its implementation is quite straightforward for the application of the different TVD flux limiters.

Li et al. (2008) discusses in their work the limitations of the Darwish method, showing that, if the field has a parabolic distribution, it will result in an accurate computation of the r factor. However, if the distribution is exponential, there will exist an inaccuracy issue, even on a Cartesian grid, due to the extrapolation effect of the cell centered gradient.

Another issue comes from the fact that the quantity $(\nabla\phi)_C$ can filter information for a specific direction. For example, in a 2D polyhedral grid each cell can have six neighbors and the gradient vector has only two values. This means that the use of the cell centered gradient should be restricted for small corrections.

Li method consists in computing an approximation of the ϕ_U value at the point U, when computing the r factor. To achieve this, a search algorithm of the closest cell to the point U is required. In this search, the neighbors by vertex $\mathcal{N}_v^1(C)$ of the cell C are included (the cells that contain at least one vertex of the cell C). Figure 2.14 shows examples of the Li method, where the closest cell in the search is marked with an U_r. In the Li method, the r factor equation (2.128) is modified to the following expression:

$$r = \frac{\phi_C - \phi_{U_r} - (\mathbf{U} - \mathbf{U_r}).(\nabla\phi)_{U_r}}{\phi_D - \phi_C} \tag{2.132}$$

Li et al. (2008) verified that in his method there was a reduction of the wiggles effect in the numerical solution when comparing with the Darwish method, although both methods converged for a similar number of iterations.

After analyzing these two methods, some issues already discussed in the Thesis can be raised, which will cause an inaccuracy problem when computing the r factor:

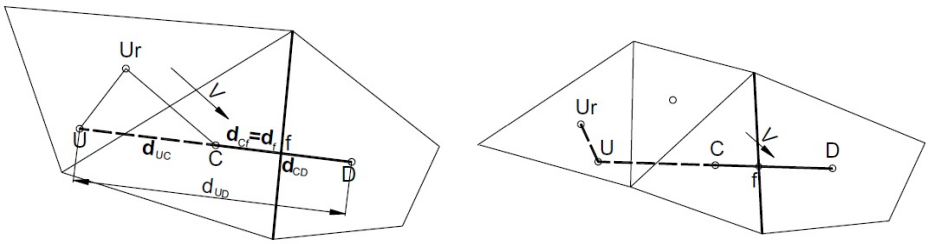


Figure 2.14: Examples of the Li method - from Li et al. (2008).

- These methods do not depend of the face centroid $\mathbf{f}$, so a numerical error is committed by not considering the skewness or eccentricity factor $\sigma(f)$ from the grid, which was explained previously in subsection 2.3.2. This is important since the convective schemes that consider the skewness factor recover the expected second-order accuracy when the grid quality of the unstructured grid is not the ideal one.

- It is showed in previous subsections that both convective and diffusive schemes can be improved by considering additional correction terms, which depend on the cell centered gradient $\nabla\phi$. Since this quantity results in an averaging process from the surrounding values, its use should be as small as possible.

Proposed Projection Method

With the ghost points schemes (equation (2.3.4)) it is possible to project the computational points to a line parallel to the face normal that passes through the face centroid f. Bear in mind, that the minimum distance between the computational point and the projection line is the same distance between the computational point and the ghost point. The proposed method to compute the r factor of the TVD scheme is based in this concept and will be named projection method, which is represented in the figure 2.15.

The ghost point C' is computed by projecting the vector $(\mathbf{f}-\mathbf{C})$ into the 1D space defined by the face normal:

$$\mathbf{C}' = \mathbf{f} - \frac{[(\mathbf{f}-\mathbf{C}).\mathbf{S}_f]\mathbf{S}_f}{\mathbf{S}_f.\mathbf{S}_f} \tag{2.133}$$

and the distance vector between the cell centroid $\mathbf{C}$ and the ghost point $\mathbf{C}'$ is given by:

$$\mathbf{d}_{CC'} = \mathbf{C}' - \mathbf{C} = \mathbf{f} - \mathbf{C} - \frac{[(\mathbf{f}-\mathbf{C}).\mathbf{S}_f]\mathbf{S}_f}{\mathbf{S}_f.\mathbf{S}_f} \tag{2.134}$$

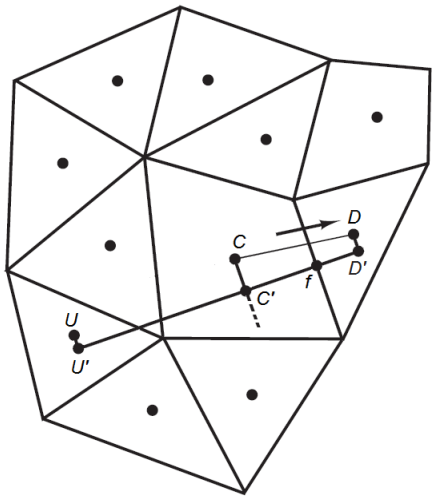


Figure 2.15: Projection method for TVD schemes.

Afterwards, the equations (2.133) and (2.134) are also applied to the points D and U. To determine cell U, a search algorithm is required to find the cell with the lowest distance $\mathbf{d}_{UU'}$ and that makes the point C' being located between the points U' and D'. In the search algorithm, all cells that have vertices of the cell C are included $\mathcal{N}_v^1(C)$. Additionally, if the cell C has boundary faces, their centroids are also included in the search. After computing the location of the three projection points, the respective ghost point values are computed by using linear interpolation:

$$\phi_{C'} = \phi_C + \mathbf{d}_{CC'}.(\nabla\phi)_C \tag{2.135}$$

The r factor from equation (2.128) must be modified to account for the values of the ghost points. Also a correction is added in order to consider the different distances between the ghost points, resulting in the following equation:

$$r = \frac{(\phi_{C'} - \phi_{U'})||\mathbf{D}' - \mathbf{C}'||}{(\phi_{D'} - \phi_{C'})||\mathbf{C}' - \mathbf{U}'||} \tag{2.136}$$

and finally, the equation (2.127) that is used to compute the face centroid value ϕ_f, must be modified to use the ghost points, resulting in the following equation:

$$\phi_f = \phi_{C'} + \frac{1}{2}\psi(r)(\phi_{D'} - \phi_{C'}) \tag{2.137}$$

this concludes the exposition of the projection method for application TVD schemes on unstructured grids.

Chapter 3

Results on Unstructured Grids

This chapter comprises the results obtained on unstructured grids, without considering any kind of adaptive grid refinement. It is divided in two sections:

- In the first one, the FVM verification is done by solving two different cases. One with an analytical solution to study the numerical error decay with the grid refinement and a second one with benchmark data to compare this data with the numerical results obtained by the code.

- The second section has a comparison of three methods to apply TVD schemes on unstructured grids. Two of them come from the literature and the other one is proposed by the author to correct the r factor computation by considering the skewness of the grid.

3.1 Finite Volume Verification

3.1.1 Analytic Cavity - Second Order Verification

To study the numerical error decay during the grid refinement, the analytic 2D cavity problem was selected. This benchmark appeared, previously, in Shih et al. (1989), Kobayashi et al. (1999) and Pereira and Pereira (2001), where details of this analytical solution can be found. In this example $\nu = 1.0$ was used, which corresponds to a Reynolds number of $Re = UL/\nu = 1.0$. These values are selected to give the convective and diffusive terms the same weight in the Navier-Stokes equations.

The 2D momentum equations of the flow are considered with an additional source term given by:

$$B(x,y) = \frac{8}{Re}\left[24\int \zeta_1(x)dx + 2\zeta_1'(x)\zeta_2''(y) + \zeta_1'''(x)\zeta_2(y)\right] \\ -64\left[\Phi_2(x)\Psi(y) - \zeta_2(y)\zeta_2'(y)\Phi_1(x)\right] \tag{3.1}$$

where $'$ is the differential operator and the functions $\zeta_1(x)$, $\zeta_2(y)$, $\Phi_1(x)$, $\Phi_2(x)$ and $\Psi(y)$ are defined by:

$$\zeta_1(x) = x^4 - 2x^3 + x^2 \tag{3.2}$$

$$\zeta_2(y) = y^4 - y^2 \tag{3.3}$$

$$\Phi_1(x) = \zeta_1(x)\zeta_1''(x) - \zeta_1'(x)\zeta_1'(x) \tag{3.4}$$

$$\Phi_2(x) = \int \zeta_1(x)\zeta_1'(x)dx \tag{3.5}$$

$$\Psi(y) = \zeta_2(y)\zeta_2'''(y) - \zeta_2'(y)\zeta_2''(y) \tag{3.6}$$

The problem is solved in a square domain $[0;1]^2$ and a Dirichlet boundary condition is imposed in all boundaries with zero velocity, except for the upper boundary ($y = 1.0$) where the function (3.7) is imposed:

$$u(x,1) = 16\zeta_1(x) \tag{3.7}$$

Finally, the analytical velocity field, which is independent of the Reynolds number, is given by:

$$u(x,y) = 8\zeta_1(x)\zeta_2'(y) \tag{3.8}$$

$$v(x,y) = -8\zeta_1'(x)\zeta_2(y) \tag{3.9}$$

and the pressure field is given by:

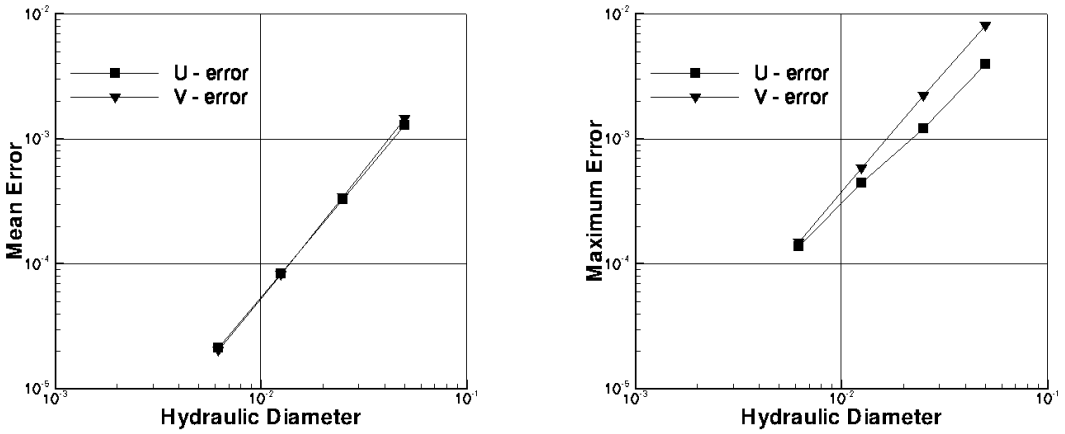
$$p(x,y) = \frac{8}{Re}\left[24\int \zeta_1(x)dx\zeta_2'''(y) + 2\zeta_1'(x)\zeta_2'(y)\right] \\ +64\Phi_2(x)\left[\zeta_2(y)\zeta_2''(y) - \zeta_2(y)\zeta_2(y)\right] \tag{3.10}$$

The numerical solution is computed for a sequence of four Cartesian grids, the first grid has $20 \times 20 = 400$ cells and the other ones are obtained by successive refinement. Since it is solved in a Cartesian grid the CDS, the LIN and the Gauss method are used respectively as the diffusive, convective and gradient schemes of the FV discretization.

The local error E_P of the cell P can be obtained by computing the absolute deviation between the cell value and the analytical solution at the cell centroid. To evaluate

globally the error field of the grid, the mean and maximum error can be computed[1]. The mean error is computed as the volume average of the error and the maximum error is the highest error value in the all grid. The volume average is important due to the different cell volumes that occur in both unstructured and adaptive grids.

Figure 3.1 shows both the mean and maximum velocity error evolution with the hydraulic diameter h, the velocity components are considered separately. It is common in the literature to show these curves using a logarithmic scale. Both u and v error have similar tendencies, the only differences is a slightly lower slope in the curve of u maximum error.



(a) Mean error.

(b) Maximum error.

Figure 3.1: Mean and maximum error evolution with the hydraulic diameter.

To compute the slope p of the error curve a formula can be deduced by assuming that the error varies with the hydraulic diameter $E \sim h^p$:

$$p = \frac{ln(E_{i+1}/E_i)}{ln(h_{i+1}/h_i)} \tag{3.11}$$

The slope error decay for each of the global quantities are listed in table (3.1). Since the values are close to 2 for the mean errors, this means that the FV discretization has second order accuracy.

The second order verification for unstructured grids is done in the adaptive results chapter 5 to different cases with an analytic solution. Please note that the discretization on an adaptive grid is the same as on an unstructured grid.

[1]Some authors in the literature use the terminologies $Norm_1$ and $Norm_\infty$.

Quantity	Order value p
Mean error u	1.97
Maximum error u	1.625
Mean error v	2.037
Maximum error v	1.945

Table 3.1: Order values p for the analytical cavity case.

3.1.2 Cavity Flow - 2D Cartesian Grid Verification

In this case, the flow under consideration is the driven square cavity flow. The boundary conditions consist of the no-slip condition in all boundaries, but the top boundary moves at a given constant velocity U. The Reynolds number is given by $Re = UL/\nu$ where L is the cavity length. For this case $U = 1$, $L = 1$ and $\nu = 0.001$, which results in a Reynolds number of 1000.

The first results were obtained with a Cartesian grid of $160 \times 160 = 25600$ hexahedrons using the same FV schemes from the previous case. Figure 3.2 shows that the results agree very well with the benchmark data of the finite-difference predictions of Ghia et al. (1982) or the spectral solutions by Botella and Peyret (1998). This means that the code implementation is verified for the particular case of Cartesian grid, from the FV discretization until the assembly of the Navier-Stokes equations and respective coupling.

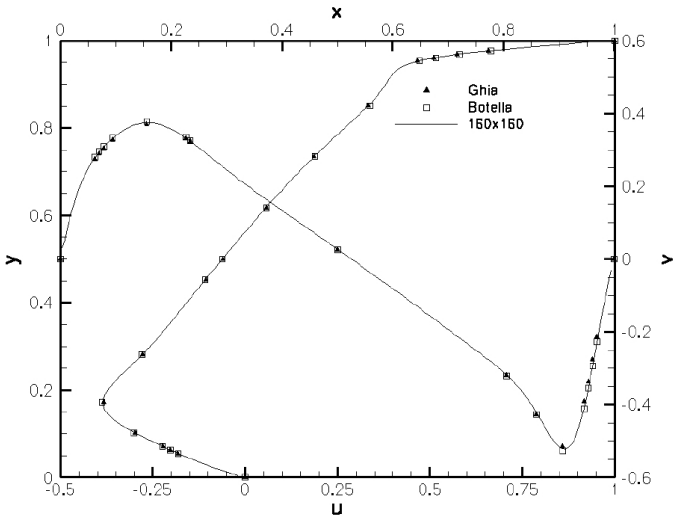


Figure 3.2: Cartesian grid: comparison of the velocity profiles with benchmark results from Ghia et al. (1982); Botella and Peyret (1998).

3.1.3 Cavity Flow - 2D Unstructured Grids Verification

In this subsection, the cavity flow case is solved using 3 different types of 2D unstructured grids: a triangular one, a hybrid one and a polyhedral one.

The triangular grid is a mesh[2] composed by triangles cells or 3D prisms[3], the hybrid grid is a mesh composed by both quadrilaterals and triangles, this grid is obtained by agglomeration of a random set two of neighbor triangles, so a initial triangle grid is required to generate this type of mesh. Finally, the polyhedral grid is obtained by using the Voronoi diagram in a triangle grid.

The three grids at study have approximately the same reference length $h = 1/160 = 0.00625$ from the Cartesian grid case and each grid has respectively 58030, 47525 and 29336 cells. Figure 3.3 shows a comparison between the velocity profiles of the three unstructured grids with benchmark data. Once again a very good agreement between the results can be observed and the differences between the unstructured grids are almost non-existent. This means that the code is verified for this kind of unstructured grids for the computation of laminar flows with this type of boundary conditions.

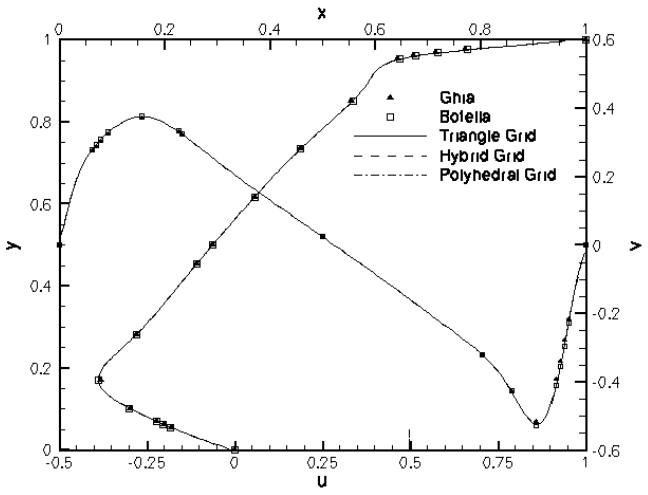


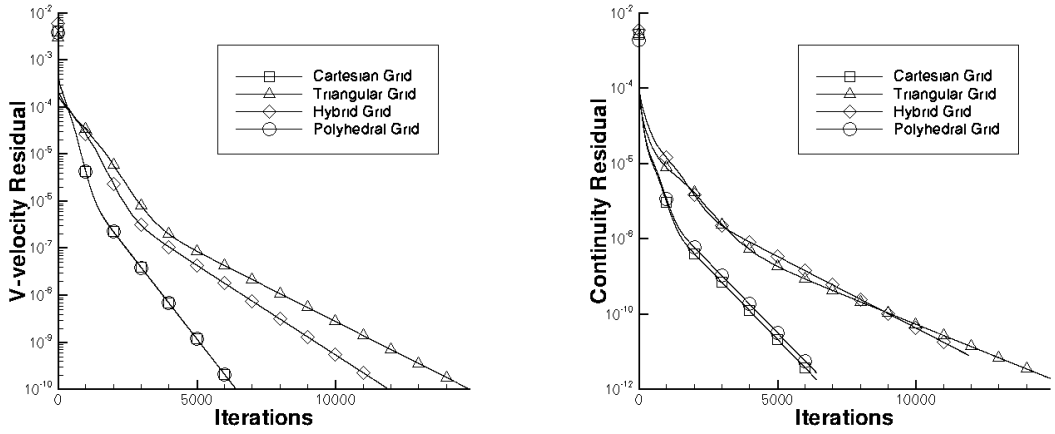
Figure 3.3: Unstructured grids: comparison of the velocity profiles with benchmark results from Ghia et al. (1982); Botella and Peyret (1998).

Figure 3.4 shows the v-velocity and continuity residual for the Cartesian and unstructured grids, where a comparison of these curves can be done. The bottleneck in the convergence process is the v-velocity. which is the last residual to reach the defined

[2]Grid and mesh are synonyms in the context of this thesis.

[3]Some CFD codes use always a 3D geometry data of the grid even for 1D and 2D cases and this is the case of the SOL code.

tolerance of 10^{-10} for the four cases.



(a) V-velocity residual.

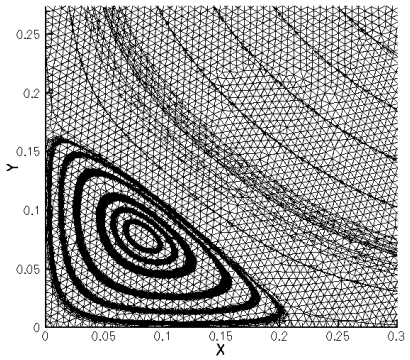
(b) Continuity residual.

Figure 3.4: V-velocity and continuity residual evolution of the four grids at study.

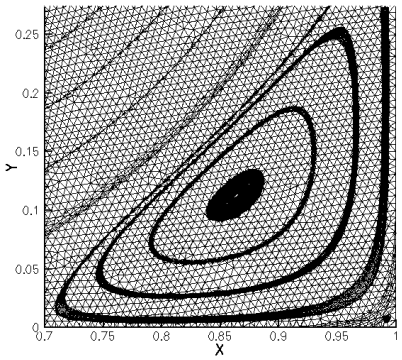
From the comparison of the residuals it can be observed that the residual between the Cartesian and polyhedral grids are very similar in both cases. The hybrid grid converges with almost the double number of iterations and the triangle grids is the case that takes more iterations to converge.

From the study performed by Darwish et al. (2009) it is concluded that the increment of the grid number of cells increases the number of iterations required to converge the solution. This conclusion is consistent with the v-velocity curve, see figure 3.4(a). In the case of the continuity residual, it suffers more the influence from the grid geometric topology, since the curves of the hybrid and triangle grids are more close to each other.

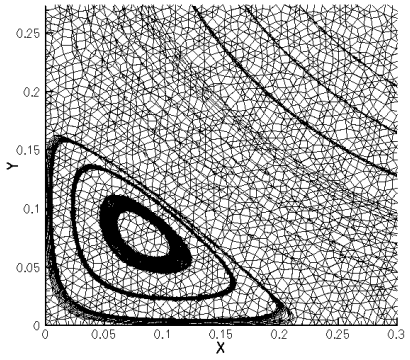
Finally, the two corner vortices located at the cavity bottom are showed for a streamline plot in the figure 3.5. This proves that the code has the ability to obtain the same geometric form of these structures on grids with different topologies.



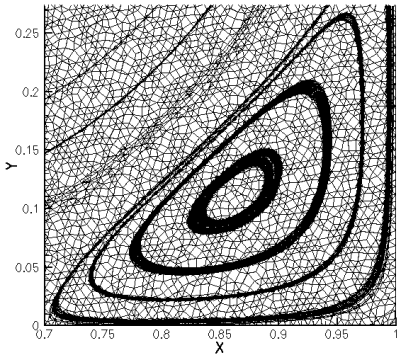
(a) Triangle grid: Left vortex.



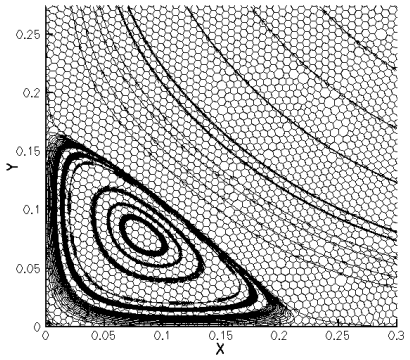
(b) Triangle grid: Right vortex.



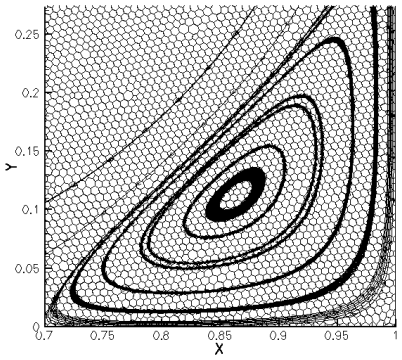
(c) Hybrid grid: Left vortex.



(d) Hybrid grid: Right vortex.



(e) Polyhedral grid: Left vortex.



(f) Polyhedral grid: Right vortex.

Figure 3.5: Vortices structures at the cavity bottom for different unstructured grids.

3.2 TVD schemes on Unstructured Grids

3.2.1 Introduction

In this section, the results obtained with the implemented TVD convective schemes are showed and discussed. The numerical simulations will be done on different types of unstructured grids for cases with a known analytical solution, in order to make a comparison study. Thus, a discussion between the advantages and disadvantages of the two methods from the literature (Darwish and Moukalled (2003); Li et al. (2008)) and the proposed projection method can be performed.

To compute these numerical solutions, the generalized transport equation without diffusive and source terms is considered:

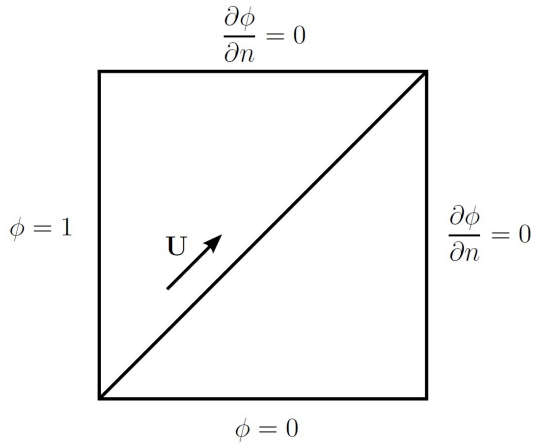
$$\nabla.(\mathbf{U}\phi) = 0 \tag{3.12}$$

where the convective velocity $\mathbf{U}$ is fixed. The equation is solved implicitly with the deferred correction method, for several iterations until the second norm of the iterative residual is lower than 10^{-6} or the maximum number of 10000 iterations is reached. The iterative residual is computed as the absolute difference of the solution between iteration $n+1$ and n. The cell centered gradient used for these results was the WLS.

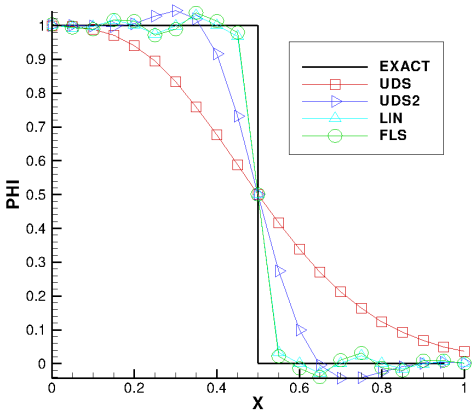
3.2.2 Cartesian Grid

For the first case of this study, the problem of the diagonal convection is considered with a function that has a singularity in the gradient to cause numerical instabilities in the convective schemes. This problem is solved using a 20×20 cartesian grid with a reference squared domain $L = 1$. Dirichlet BC are imposed with the indicated values of the figure 3.6(a), and the remaining boundaries have a Neumann BC prescribed with a null gradient. A convective velocity is imposed with the value 1 in both components of the reference axis.

Figure 3.6(b) shows the results obtained for 4 convective schemes (UDS, UDS2, LIN and FLS), which do not use any flux limiter, in line section located for $y = 0.5$. From the four schemes, the UDS is the only that presents bounded results between the imposed values at the boundaries, although it suffers from false diffusion as it can be seen in the UDS curve. The second order Upwind scheme has only one overshoot and undershoot in the results. Considering the LIN and FLS schemes, they are severely affected by the instabilities: which is observable due to the several oscillations that



(a) Analytical case geometric description.



(b) Results with convective schemes $y = 0.5$.

Figure 3.6: Cartesian grid: analytical case geometric description and results with convective schemes.

occur before and after the transition zone $x = 0.5$[4].

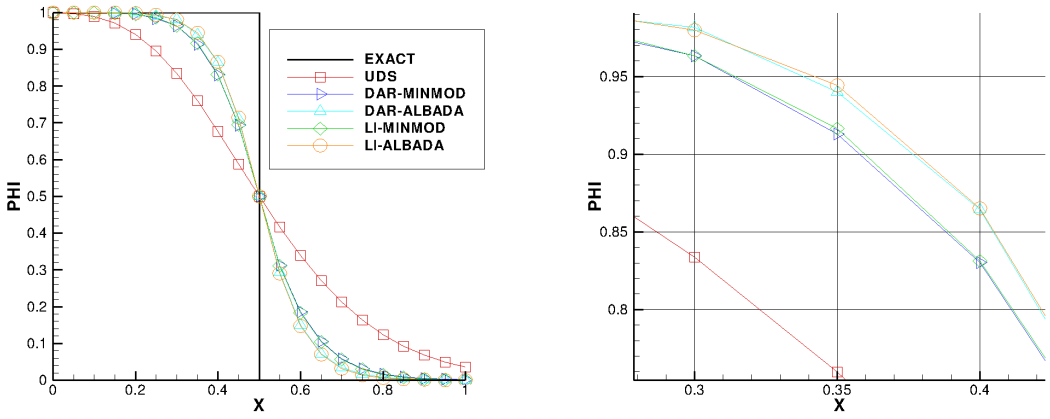
These results show that the second order schemes developed for unstructured grids are not suitable to solve problems with discontinuous fields, even for cases with cartesian grids which do not have skewness or warp angles quality problems.

For the same case, the TVD schemes for the Darwish and Li methods are studied. Due to simplifications reasons not all results from the simulated cases will be showed. Since a cartesian grid has null skewness factor $\sigma(f)$, the Li and the project methods produce the same results. Therefore, the curves obtained with projection method will not be showed for this case.

Figure 3.7(a) shows the obtained results for the Darwish and Li methods using the MINMOD and Van Albada flux limiters for the same line section $y = 0.5$ and figure 3.7(b) shows a zoom near the point $x = 0.35$.

For the same flux limiters, both methods show similar results and this fact is observable even in the zoom (figure 3.7(b)). Thus, the differences between the Darwish and Li method could be observed for this case: the TVD results show are more accurate than the UDS ones and are bounded between the imposed boundary values, showing that the implemented TVD approach is being applied.

[4]Although these schemes also have the most accentuated gradient in this zone.



(a) Results for the line section $y = 0.5$.

(b) Results zoom near $x = 0.35$.

Figure 3.7: Cartesian grid: results for two TVD methods with the flux limiters MINMOD and Van Albada.

3.2.3 Triangular Grid

In this subsection, the results obtained for a triangular grid, with a hydraulic diameter $h = 0.05$ and 892 cells (see figure 3.8(a)), are presented. For this case, the analytic solution consists on the diagonal convection of a step function (see figure 3.8(b)), which is created by imposing the following Dirichlet BC at the left boundary $x = 0$:

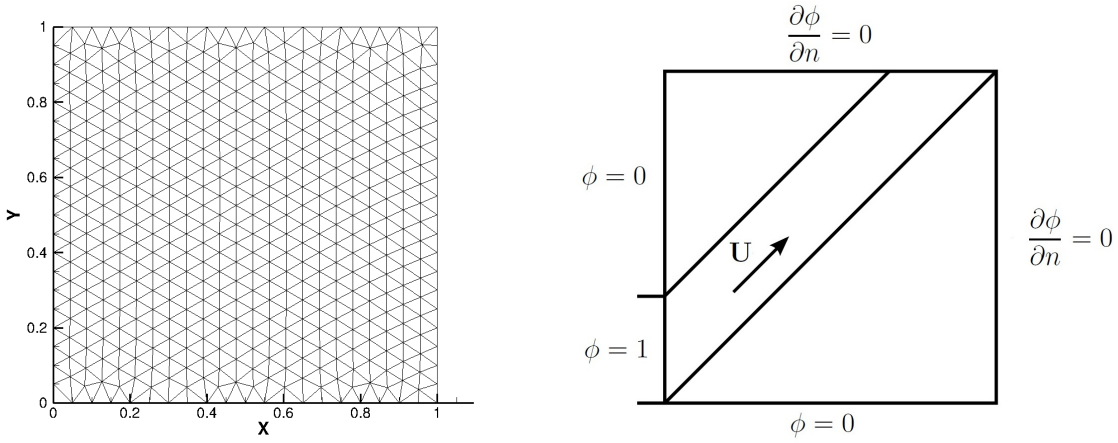
$$\phi = \begin{cases} 0 & \Leftarrow \quad y \geq 0.3 \\ 1 & \Leftarrow \quad y < 0.3 \end{cases} \tag{3.13}$$

and for the other boundaries the imposed conditions are the same as in the previous case.

Figure 3.9 shows the results obtained for the three TVD methods at study, for two flux limiters the MINMOD and the Van Albada. The results are extracted from a line probe located at $y = 0.8$.

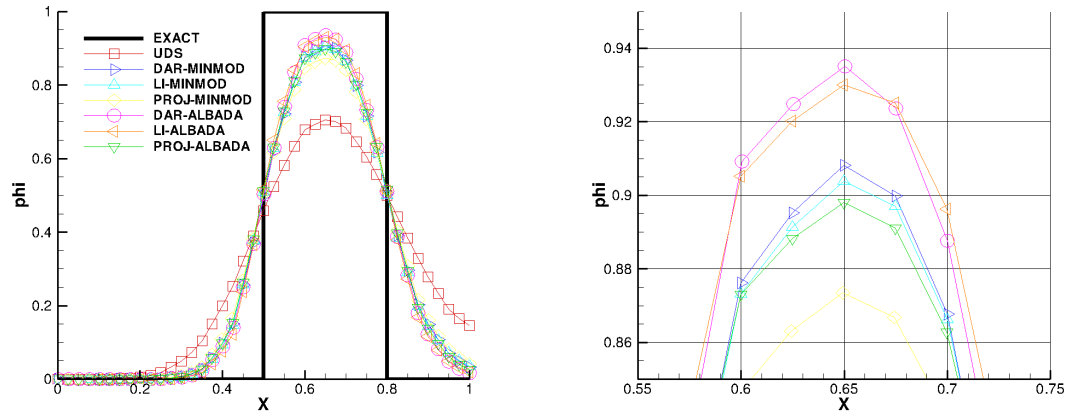
From the analysis of figure 3.9, it can be observed that the Darwish and Li methods have results closer to the analytical solution than the projection method since it presents more numerical dissipation. Although the Darwish method has superior results than the Li method, it was found that it did not have good convergence properties, since this method never converged to the iterative residual tolerance of 10^{-6} for all flux limiters.

Afterwards, a convergence study is carried out for these results with the Li and projection methods. Figure 3.10(a) shows the iterative residual evolution for the two



(a) Triangular mesh. (b) Analytical case geometric description.

Figure 3.8: Triangular grid: triangular mesh and analytical case geometric description.



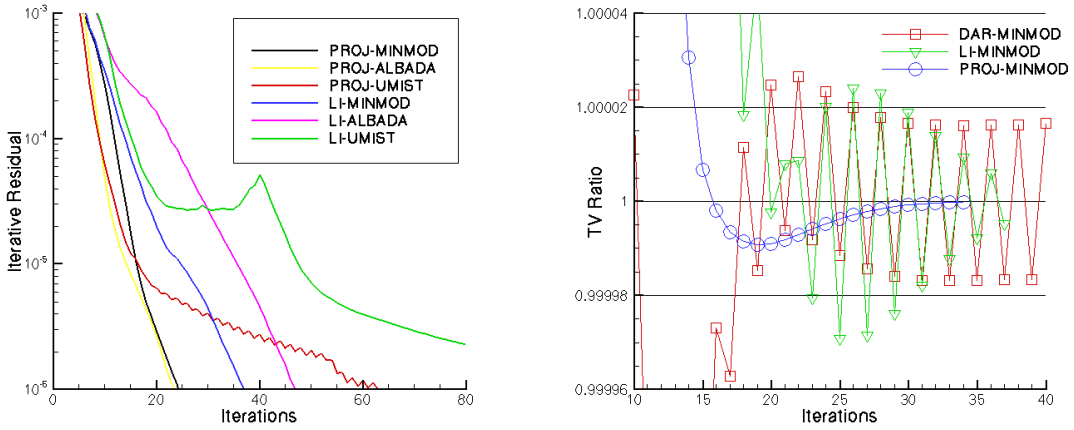
(a) Results for the line section $y = 0.8$. (b) Results zoom near $x = 0.65$.

Figure 3.9: Triangle grid: results for three TVD methods with the flux limiters MINMOD and Van Albada.

method with three different flux limiters, where it is observable that the project method converges for a lower number of iterations (approximately half) than the Li method. That happens because the computation of the r factor is taken into account in the projection method, so its computation is more accurate and the flux limiters effect is more evident in the computation. It was also noted that the UMIST flux limiter takes more iterations to converge than the other ones. To understand better this behavior a quantity based in the total variation (TV) definitional (equation (2.126)) is introduced, which is the TV ratio (TV_{ratio}) and is defined by the following equation:

$$TV_{ratio} = \frac{TV(\phi^{n+1})}{TV(\phi^{n})} \tag{3.14}$$

and if the TVD condition is satisfied the TV ratio is lower or equal to 1.



(a) Convergence evolution for the TVD methods. (b) TV ratio evolution for the TVD methods.

Figure 3.10: Triangle grid: convergence and TV ratio evolution for the TVD methods.

Figure 3.10(b) shows the TV ratio evolution for the three TVD methods with the MINMOD flux limiter. The Darwish method has a constant oscillation between the unitary value $TV_{ratio} = 1.0$ value in each iteration and with a fixed amplitude. Due to this, the numerical results do not converge to the designated residual tolerance. In the case of the Li method, the oscillations between the unitary value in each iteration are also presented but their amplitude decays, resulting in the convergence of the solution. The TVD principle is not globally respected, despite this method having a residual decay free of oscillations (figure 3.10(a)).

In the case of the projection method, when the TV ratio is lower than 1, it stays this way until the solution converges. For this particular case, the iterative residual

tolerance was decreased to show the full TV ratio curve tendency, as it can be seen in the figure 3.10(b). These results show that the projection method is the only one that respects the total variation reduction, which is expected for these types of schemes. The TV ratio can be computed locally for each cell, by just taking into account the cell faces contribution. This quantity could be used in the future to improve the desirable effect of flux limiters if the TVD principle is not being fulfilled.

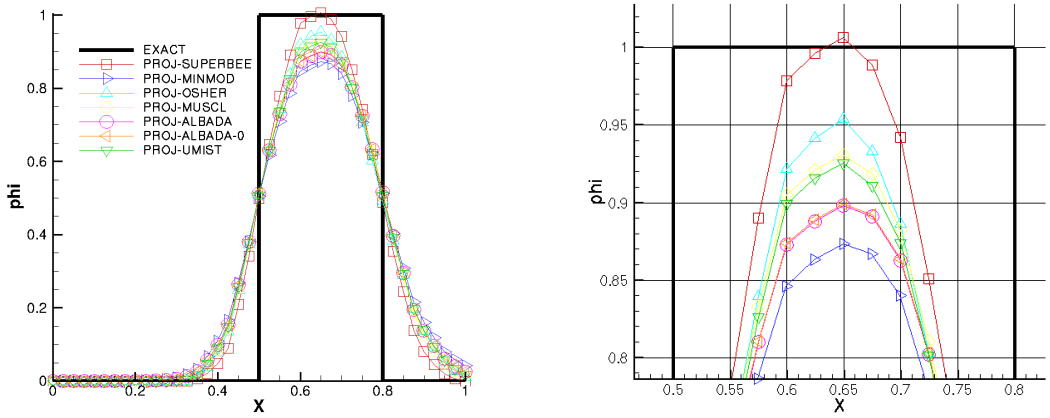
The better convergence properties of the projection method result in a significant reduction of the overshoot and undershoots values. These values are listed in the table 3.2 for all the simulated cases and are computed considering all the points of the computational domain.

Flux Limiter	TVD Method	Triangle Grid	
		Overshoot	Undershoot
SUPERBEE	Darwish	$7.470714E-02$	$2.630533E-02$
	Li	$1.041570E-01$	$9.088535E-02$
	Projection	$3.043870E-02$	$1.598411E-04$
MINMOD	Darwish	$4.506625E-02$	$2.484922E-03$
	Li	$0.000000E+00$	$6.113121E-03$
	Projection	$0.000000E+00$	$0.000000E+00$
OSHER	Darwish	$4.368892E-02$	$4.368892E-02$
	Li	$0.000000E+00$	$8.410396E-03$
	Projection	$0.000000E+00$	$1.030365E-02$
MUSCL	Darwish	$6.636838E-02$	$8.069478E-03$
	Li	$2.595976E-03$	$5.669911E-02$
	Projection	$7.163560E-05$	$1.763501E-06$
Van Albada	Darwish	$5.179832E-02$	$4.083330E-03$
	Li	$3.960993E-04$	$1.495524E-02$
	Projection	$0.000000E+00$	$0.000000E+00$
Van Albada - 0	Darwish	$5.229272E-02$	$4.075890E-03$
	Li	$3.997572E-04$	$1.503622E-02$
	Projection	$1.978424E-04$	$4.217010E-10$
UMIST	Darwish	$7.350687E-02$	$1.076967E-02$
	Li	$1.033364E-02$	$3.136989E-02$
	Projection	$2.346119E-03$	$5.436182E-05$

Table 3.2: Triangle grid: obtained overshoot and undershoot values.

Figure 3.11 shows the obtained results for the projection method with all of the flux limiters at study. Like it is concluded by Li et al. (2008), the flux limiters that have higher maximum values are the SUPERBEE and OSHER. In figure 3.11 the SUPERBEE flux limiter clearly passes the $\phi = 1.0$ upper limit. On the other hand, the MINMOD and Van Albada flux limiters show lower results since they have higher

false diffusion and additionally these limiters are free of overshoots and undershoots in the numerical solution (as it can be seen at the table 3.2).



(a) Results for the all section $y = 0.8$.

(b) Results zoom near $x = 0.65$.

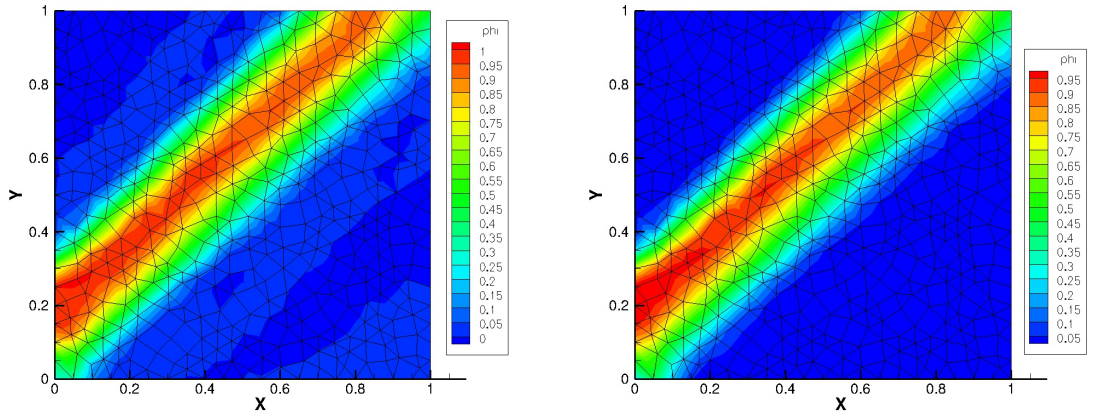
Figure 3.11: Triangle grid: results for projection method with all flux limiters at study.

3.2.4 Hybrid Grid

In this subsection, the same analytical solution from the previous case is used to solve the convection equation on a hybrid grid, composed by 731 cells. This grid is obtained by aleatory agglomeration of some cells from a triangle grid and afterwards it passes to an orthogonalization process with an elliptic generator. This kind of grids are used as a challenge to the TVD schemes, due to the significant volume variations between the cells.

Figures 3.12(a) and 3.12(b) show the contour plot of the numerical solution obtained with the Li and projection method, respectively, for the flux limiter OSHER. Although the contour plots are similar, the different scales and the negative contours (dark blue) in figure 3.12(a) show that the numerical solution of Li method is not bounded or not completely inside of the imposed limits by the analytical solution, unlike the projection method.

For other flux limiters, the projection method has lower values of undershoot and overshoot when compared with the other methods. For example, the flux limiters MINMOD, Van Albada and UMIST have a null overshoot value with the projection method. In the other cases, the overshoot and undershoot values are one or more orders of magnitude lower than with the Darwish and the Li methods (see table 3.3).



(a) Contour plot for the Li method.

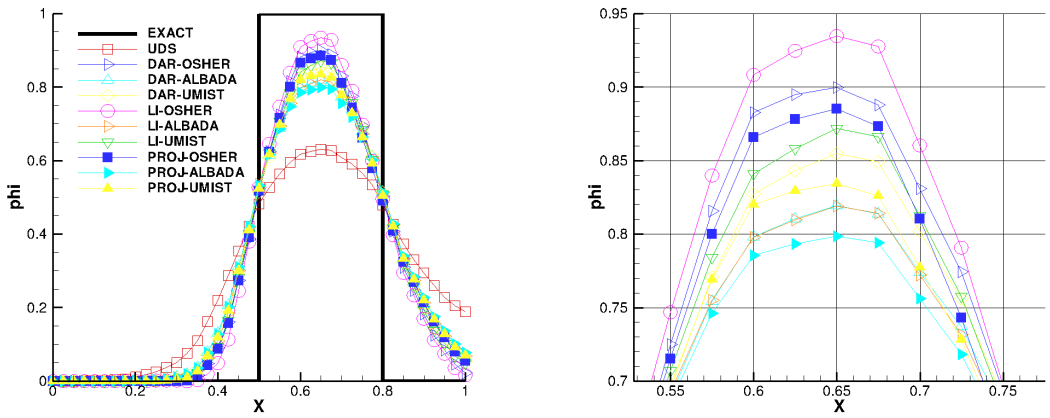
(b) Contour plot for the projection method.

Figure 3.12: Hybrid grid: contour plot for the Li and projection method with the OSHER flux limiter.

Flux Limiter	TVD Method	Hybrid Grid	
		Overshoot	Undershoot
SUPERBEE	Darwish	$1.882454E-02$	$1.915379E-02$
	Li	$8.058661E-02$	$6.949808E-02$
	Projection	$3.468411E-03$	$1.235839E-02$
MINMOD	Darwish	$4.210421E-03$	$9.404591E-04$
	Li	$3.213569E-03$	$7.222680E-03$
	Projection	$0.000000E+00$	$4.901668E-03$
OSHER	Darwish	$1.303373E-02$	$1.860756E-02$
	Li	$3.715890E-02$	$2.184242E-02$
	Projection	$7.594598E-04$	$7.897562E-04$
MUSCL	Darwish	$1.289964E-02$	$7.652252E-03$
	Li	$9.788584E-03$	$6.831873E-03$
	Projection	$5.486960E-04$	$5.586190E-03$
Van Albada	Darwish	$8.950148E-04$	$1.415682E-03$
	Li	$7.383857E-03$	$8.049898E-03$
	Projection	$0.000000E+00$	$2.491915E-04$
Van Albada - 0	Darwish	$5.783595E-03$	$1.030591E-03$
	Li	$6.551860E-03$	$8.343659E-03$
	Projection	$0.000000E+00$	$7.117527E-04$
UMIST	Darwish	$1.467938E-02$	$1.028316E-02$
	Li	$9.479815E-03$	$6.198651E-03$
	Projection	$0.000000E+00$	$5.750208E-03$

Table 3.3: Hybrid grid: obtained overshoot and undershoot values.

Figure 3.13 shows the obtained results for a line section at $y = 0.8$ for the three methods at study with the three different flux limiters. The goal is to make a comparison study between the numerical results.



(a) Results for the all section $y = 0.8$.

(b) Results zoom near $x = 0.65$.

Figure 3.13: Hybrid grid: results for the three methods at study with the OSHER, Van Albada and UMIST flux limiters.

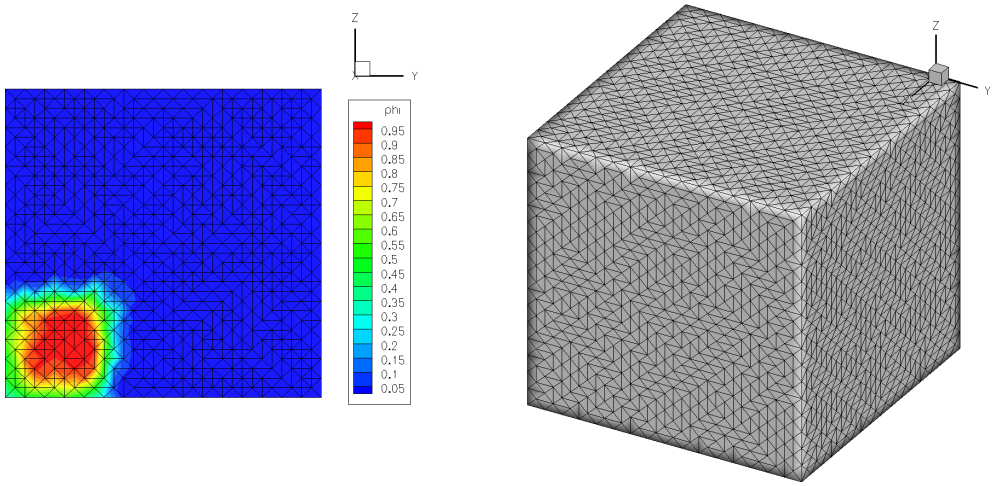
As it was observed previously, the projection method shows lower results than the other methods since it is more dissipative. The differences between the methods vary between 0.01 and 0.02. Also the numerical results are lower than compared to the case of the triangle grid due to the higher numerical dissipation.

3.2.5 Tetrahedral Grid

A three dimensional (3D) case is built by considering the diagonal convection of a square with $l = 0.3$ of side and located at the $x = 0$ plane. The problem is computed on a cubic computational domain with side of $L = 1$ and each component of the fixed convective velocity $\mathbf{U}$ and equal to 1. This problem is an 3D extension of the analytical solution used in the previous subsections 3.2.3 and 3.2.4.

For this test case, a tetrahedral grid with 49636 cells is used, for the three methods at study. Figure 3.14(a) shows the imposed boundary condition at the plane $x = 0$ and figure 3.14(b) shows a perspective of the tetrahedral grid used.

The 3D tetrahedral grid is an important test for the TVD schemes, since the interior faces can have different geometric displays along the computational domain, affecting the convection of the imposed square and increasing the numerical diffusion.



(a) Imposed boundary condition at $x = 0$.

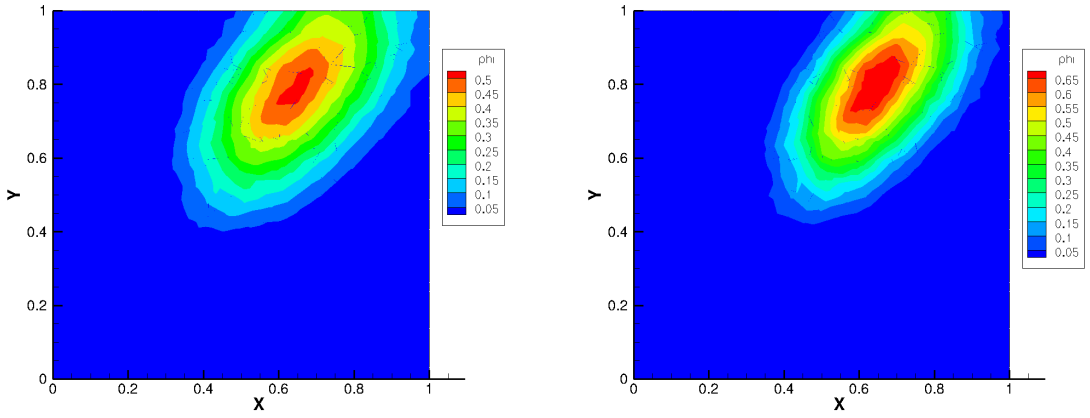
(b) Perspective of the tetrahedral mesh.

Figure 3.14: Tetrahedral grid: imposed boundary condition at the plane $x = 0$ and perspective of the tetrahedral mesh.

For a first interpretation of the results, the contour plots for a section plane at $z = 0.8$ is obtained. The analytical solution at this plane is a parallelogram with an inclination of 45 degrees, centered at the point with coordinates $(0.65, 0.8, 0.8)$ and with 0.3 of height. Figure 3.15 shows two examples of these contour plots, one of them obtained with the UDS and the other one with the MUSCL flux limiter for the projection method. It is noticeable a higher numerical diffusion with the UDS case, because of the curvilinear distribution solution and a lower scale of values. In the case of the figure 3.15(b), the numerical diffusion is lower and the solution distribution has less curvature (although it has still considerable differences from the analytical solution).

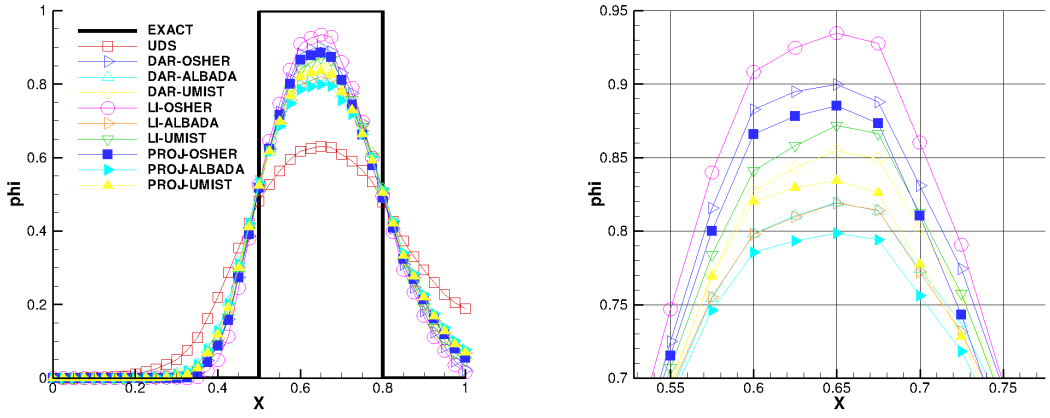
After obtaining these results, a line section defined by $y = z = 0.8$ is used to compare the different numerical results. In figure 3.16, the results obtained for the three TVD methods with selected flux limiters are showed. From a general point of view, the results have some degree of numerical diffusion, since the curves from the TVD schemes are closer to the UDS result than the analytic solution.

By analyzing table 3.4 with the obtained overshoot and undershoot values, it is concluded once again that the projection method has values one or two order magnitude lower than the other methods. The overshoot and undershoot values obtained with this grid are much higher than with the previous grids, specially with the Darwish method. This demonstrates that the TVD schemes are not completely applicable to tetrahedral grids and in the resolution of 3D problems with complex geometries. Bear



(a) UDS. (b) Projection method with MUSCL flux limiter.

Figure 3.15: Tetrahedral grid: contour plot at a plane section $z = 0.8$ for the UDS and the projection method with MUSCL flux limiter.



(a) Results for the all section $y = 0.8$, $z = 0.8$. (b) Results zoom near $x = 0.65$.

Figure 3.16: Tetrahedral grid: results for projection method with all flux limiters at study.

in mind that the number of studies focusing on TVD schemes for this kind of grids is very low.

Flux Limiter	TVD Method	Tetrahedral Grid	
		Overshoot	Undershoot
SUPERBEE	Darwish	–	–
	Li	$1.227445E-01$	$1.179273E-01$
	Projection	$6.936989E-02$	$4.608841E-03$
MINMOD	Darwish	$1.889054E-01$	$5.882758E-02$
	Li	$4.077224E-02$	$5.137406E-02$
	Projection	$2.928273E-03$	$4.459196E-04$
OSHER	Darwish	–	–
	Li	$7.966723E-02$	$9.622075E-02$
	Projection	$6.720500E-02$	$3.904223E-03$
MUSCL	Darwish	$1.790426E-01$	$1.172419E-01$
	Li	$6.467824E-02$	$5.904153E-02$
	Projection	$1.306035E-02$	$8.104952E-04$
Van Albada	Darwish	$1.907406E-01$	$1.066235E-01$
	Li	$5.607558E-02$	$6.005168E-02$
	Projection	$9.822187E-03$	$5.266756E-03$
Van Albada - 0	Darwish	$1.936765E-01$	$1.058539E-01$
	Li	$5.696992E-02$	$5.731571E-02$
	Projection	$3.594216E-03$	$8.858051E-04$
UMIST	Darwish	$1.814518E-01$	$1.023068E-01$
	Li	$6.321429E-02$	$5.654292E-02$
	Projection	$1.568378E-02$	$1.183650E-03$

Table 3.4: Tetrahedral grid: obtained overshoot and undershoot values.

3.2.6 Final Remarks

In this section, a comparative study of different TVD schemes response on unstructured grids is done. For this study, the methods from Darwish and Moukalled (2003), Li et al. (2008) and a proposed one by the author were considered. The last one is based on the projection of the computational points to a line define by the face normal. This way, the skewness effect from the grid is considered in the computation of the solution.

For all the tested cases, the projection method had better convergence properties and a reduction of the overshoot and undershoot values. For the case of the triangle grid, it was possible to obtain solutions free of overshoots with the projection method for certain flux limiter (this was not possible with the other methods). This perfor-

mance is justified by the better estimation of the r factor improving the usage of the flux limiter effect in the solution.

As possible future work, the concept of flux limiter for the cell centered gradient could be added to improve the computation of the TVD schemes. A good work that explains this concept is presented by Park et al. (2010). Also, a projection method for the NVD principle could be created to be applied in the computation of compressible fluid flows on unstructured and adaptive grids.

Chapter 4

Error Estimators and Adaptivity

This chapter explains the error estimators used in this Thesis. They are divided into relative and absolute error estimators and they are presented in distinct sections. Also for each type of error estimator, the adaptive algorithm is explained.

4.1 Numerical Error in CFD codes

The error to be minimized by the adaptive process is the local truncation error. Since there is no possible way to accurately evaluate this error, we can observe the final error defined as the difference between the computed and the exact profile. The average total error of cell P for a generic function Φ is given by

$$E = \frac{1}{V_P} \int_P |\Phi(\mathbf{x}) - \phi(\mathbf{x})| \, dV \tag{4.1}$$

where $\phi(x)$ is the resolved profile. The discretization error is a parcel of this error; other parcels include nonlinear error sources, modeling errors and others. Consequently, the discretization error is an approximation of the total error.

It should be noted that due to the nonlinear nature of the governing equations of fluid flow, it does not follow that the cells to be refined are those with the largest error, because the error may be a symptom rather than a cause of inaccuracy. In other words, it is possible that cells with low error level become responsible for the appearance of high errors in other regions of the flow (downstream or even upstream, depending on the nature of the flow)[1].

[1] Jasak (1996) proved this mathematically by considering a transport equation for the numerical error.

4.2 Relative Error Estimators

4.2.1 Taylor-series truncation error (TSTE)

The Taylor series expansion theorem states that any smooth function $\phi \in C^\infty$ in the vicinity of a point P can be approximated as a sum of its derivatives at P. In multidimensional form:

$$\phi(x_i\,;x_{i,P}) = \phi_P + \left(\frac{\partial\phi}{\partial x_i}\right)_P (x - x_{i,P}) + \\ + \frac{1}{2}\left(\frac{\partial^2\phi}{\partial x_i \partial x_j}\right)_P (x_i - x_{i,P})(x_j - x_{j,P}) + \cdots \tag{4.2}$$

Einstein summation is implied above. Since the present FV method implementation uses a linear profile, the truncation error is dominated by the second-order term. Using the definition of the average error :

$$E_T = \frac{1}{2V_P}\int_P \left(\frac{\partial^2\phi}{\partial x_i \partial x_j}\right)_P (x_i - x_{i,P})(x_j - x_{j,P})dV \tag{4.3}$$

and assuming that the Hessian matrix is constant inside the cell P, the integration of equation (4.3) results in:

$$E_T = \frac{1}{2V_P}\left|\left(\frac{\partial^2\phi}{\partial x_i \partial x_j}\right)_P\right| (M_{ij})_P \tag{4.4}$$

where $(M_{ij})_P$ is the inertia tensor of the cell P, computed using the methodology of Mirtich (1996). When computing the E_T absolute error estimator, the Hessian matrix values are computed from a regression made with a quadratic polynomial from the cell first and second neighbors by vertex. Due to the assumption of linear variation inside the computational cells, zones with lower errors will have lower values of the Hessian matrix. The error can be nondimensionalized by using the lower order terms T_l from the Taylor series:

$$T_l = \phi_P + l_i^P \left|\left(\frac{\partial\phi}{\partial x_i}\right)_P\right| \tag{4.5}$$

where l_i^P is the length of cell P in the direction i. The isotropic version of the TSTE criterion is thus:

$$\varepsilon_P = \frac{\left|\left(\frac{\partial^2\phi}{\partial x_i \partial x_j}\right)_P\right| (M_{ij})_P}{2V_P T_l}, \tag{4.6}$$

and the anisotropic version of ε_P is obtained by retaining only the relevant Hessian terms:

$$\varepsilon_{T,i} = \delta_{ij} \frac{\left| \left(\frac{\partial^2 \phi}{\partial x_j \partial x_k} \right)_P \right| (M_{jk})_P}{2 V_P T_l}. \tag{4.7}$$

4.2.2 Coefficient of multiple determination

Since the numerical method uses a least-squares regression to reconstruct the dependent variable profile for each cell, it is natural to determine the coefficient of determination from the regression. For clarity, we will use in this section the standard notation for regression analysis. With this notation, the profile in equation (4.2) is written as:

$$Y = \beta_0 + \sum_{k=1}^{n} \beta_k X_k, \tag{4.8}$$

where Y is a variable explained by a linear combination of n explaining variables X_k (in our case the coordinate moments). The parameters β are the ones which we want to compute and consist in the intercept and in spatial derivatives of ϕ. For example, $\beta_0 = \phi_P$, $\beta_1 = (\partial\phi/\partial y)_P$, $\beta_2 = (\partial\phi/\partial x)_P$ and so on. Clearly, the profile can be easily expanded to accommodate higher-order derivatives.

Considering all the observations, an overdetermined system of equations is formed, given by $\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$, where $\mathbf{y}$ is a vector with dimension $m \times 1$ containing the observations of Y, $\mathbf{X}$ is a matrix $m \times (n+1)$ containing the values of the several explaining variables for each observation, $\mathbf{b}$ is a vector with dimension $n+1$ with the parameters to be determined and $\mathbf{e}$ is the vector with the estimation residual for each of the observations.

The estimation residual is defined as the difference between the observed $\mathbf{y}$ and the estimated value $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{b}}$:

$$e_i = Y_i - \hat{Y}_i \tag{4.9}$$

The estimate $\hat{\mathbf{b}}$ of $\mathbf{b}$ obtained by least squares is the one that minimizes the error sum of squares (with zero mean error implied):

$$\min_{\hat{\mathbf{b}}} \sum_{i=0}^{m} e_i^2 \quad \Leftrightarrow \quad \min_{\hat{\mathbf{b}}} \mathbf{e}^T \mathbf{e} \tag{4.10}$$

The coefficient of multiple determination of the regression, R^2, expresses the extent to which the variance in the original data is explained by the variance of the predicted data. This is obtained by relating the sum of squares for the original data, SS_y and

the sum of squares for the fitted data from the regression, SS_r. R^2 is then given by:

$$R^2 = \frac{SS_r}{SS_y}. \tag{4.11}$$

The quantity R^2 is limited between 0 and 1. Low values indicate a poor correlation whereas high values a good correlation. Since we want the criterion to behave in the opposite way, the criterion is formed by the one's complement of R^2:

$$c = 1 - R^2 \tag{4.12}$$

R^2 always increases as the number of explaining variables n increases, even if it introduces no improvement in the regression. To account for this, the adjusted multiple determination coefficient (adjusted for the number of variables) can be used:

$$R^2_{adj} = 1 - \frac{m-1}{m-n}(1 - R^2) \tag{4.13}$$

An alternative to the adjusted R^2, which penalizes more the loss in degrees of freedom when the number of regressors increases is called the alternative R^2:

$$R^2_{alt} = 1 - \frac{m+n}{m-n}(1 - R^2) \tag{4.14}$$

The first comparisons of these three coefficients quickly revealed that the behavior of the criterion obtained by use of the adjusted and alternative varieties were less smooth, which created non-smooth grids, with less quality. This is explained by the fact that the number of neighbors used in the moving least squares approach is not high, and the correction in the number of degrees of freedom becomes important. For that reason, we only use the unmodified R^2.

4.2.3 Least-Squares Fit and Anisotropy

The refinement criteria extracted from R^2 is isotropic because it contains information from all spatial directions. For an anisotropic version, these diagnostics are not useful since they refer to the basic profile without particular spatial direction. Other aggregate diagnostics, including detection diagnostics in parameter space, do not distinguish the spatial directions as well. To obtain this information, we pose this problem as a specification problem that tries to identify the best set of explaining variables. This is done by comparing the diagnostics of different models. Namely, we consider the

model of the base linear profile and compare it with the model of a cubic profile which contains high order terms only in a given direction. For example, the R^2 criterion for the x direction is given by:

$$c_x = R^2(\phi_x(\mathbf{x}, P_0)) - R^2(\phi(\mathbf{x}, P_0)) \tag{4.15}$$

where $\phi(\mathbf{x}, P_0)$ is the base linear profile and $\phi_x(\mathbf{x}, P_0)$ is a profile which contains high order terms, but only those where the powers of x dominate:

$$\begin{aligned}\phi_x(x, y, z; P_0) = &\underbrace{\beta_0 + \beta_1 x + \beta_2 y + \beta_3 z}_{\text{linear profile}} + \\ &\underbrace{\beta_4 x^2 + \beta_5 xy + \beta_6 xz + \beta_7 x^3 + \beta_8 x^2 y + \beta_9 x^2 z}_{\text{high order } x \text{ profile}}\end{aligned} \tag{4.16}$$

If the difference in R^2 (dR^2) of these two profiles is significant, then we have reasons to suspect that the inclusion of the high order terms in the x direction would improve the data fit. This highlights that the profile in the x direction is not linear, and induces the conclusion that cell P_0 does not possess sufficient resolution in the x direction and, consequently, should be refined in that direction. If this is performed for the remaining space directions, an anisotropic criterion is obtained.

4.2.4 Criterion filtering

Since any *a posteriori* refinement criterion is, by definition, applied to coarse grids, the results can be stiff and rapidly varying. This in turn has the effect of making the cell distribution also stiff, which in extreme situations causes numerical instability and ultimately divergence. This situation is avoided by employing the following concomitant procedures:

- The criterion is smoothed with a truncated arithmetic kernel;
- The selection of cells to be refined is expanded by a buffer layer (of one cell);
- The refinement directions are cross-checked in the vicinity, so that conflicting preferential directions are preemptively ruled out;
- As anisotropic cells become progressively stretched, it is more difficult to refine them in counter directions, so that the degree of non-orthogonality does not become severe.

This is a multi-objective procedure which requires some fine-tuning, so that the conflicting requirements of less cells vs. quality grids are both respected to some satisfactory degree. Of particular note is the impact that the last procedure has in the limit of high resolution: the anisotropy of the grid is diminished.

4.3 Absolute Error Estimators

4.3.1 Adaptive Algorithm

Error estimators are required for the adaptive mesh refinement procedure and they try to give a good approximation of the numerical error and its distribution in the computational domain. In the framework of absolute error estimators, two adaptive algorithms are tested and explained in this section:

- One proposed by Jasak and Gosman (2000b) which requires an user defined constant λ.
- Another one created by the author which is free of this constant.

Jasak Adaptive Algorithm

For the initial tests with the absolute error estimators, the adaptive algorithm proposed by Jasak and Gosman (2000a) is use, which consists of computing the mean error $mean(|E|)$ or the first norm of the error. The cells that have an error higher than $\lambda\ mean(|E|)$ are selected for refinement, where the λ is an user-defined constant that depends on the numerical error distribution in the computational domain.

The problem with this method is that λ is less than one if the error distribution is uniform and a higher value is required if the maximum error is localized and the range of error scales is wide. This means that when assigned the λ value some previous knowledge of the problem is required and as a consequence, it cannot be used easily in different cases and some tuning is required to achieve satisfactory results.

Proposed Adaptive Algorithm

With the information from the error estimator, it is possible to compute an estimation of the ideal hydraulic diameter h_i distribution for a desired local error E_0. For a second order method in space, the following equations are valid:

$$|E| \sim Kh^2 \tag{4.17}$$

$$|E_0| \sim K h_i^2 \tag{4.18}$$

where E is the error estimation and K is an unknown constant. After some algebraic manipulation:

$$h_i \sim h\sqrt{E_0/E} \tag{4.19}$$

Ideally, formula (4.19), can be used to create adaptive grids with approximately constant error if the adaptive algorithm is combined with an automatic grid generator.

The adaptive procedure used in this Thesis is based on the maximum value of the error estimator. The cells with an error higher than $\beta\ max(|E|)$ are selected for h-refinement, where β is a factor that depends of the method's order. In the case of a second order method this value β is equal to 0.25, which is the reduction factor $(h_L/h_{L+1})^2$ of the local error, in each grid refinement. The value is used in all tested cases with this algorithm.

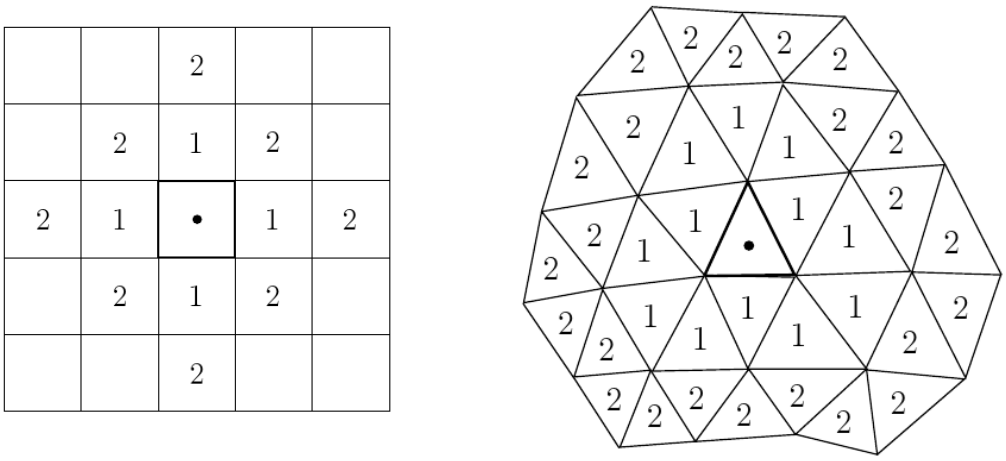
4.3.2 Regressions of High Order Polynomial

The author's own code SOL has the capability to make different types of regressions from the discrete cell values by solving a Weighted Least Squares (WLS) problem. Different types of polynomials and cell sets can be used in these regressions. Figures 4.1(a) and 4.1(b) show examples of different cell sets: figure 4.1(a) shows a cell set defined by the first and second cell neighbors by face in a cartesian grid and figure 4.1(b) shows another cell set defined by the first and second cell neighbors by vertex in a grid with triangles.

These high order regressions are required in the computation of the error estimators values and the WLS method has the versatility required in the context of the unstructured grids. The computational stencil defined by the first and second cell neighbors by vertex is used in these regressions, because this type of stencil provides a more reliable information around the reference cell. The weighted function w_p (equation (2.62)) is required to guarantee that the closest points to the reference have a higher contribution than the other points. The computational value from the reference cell is used to constrain the polynomial regression.

4.3.3 Residual Least Squares

A regression is done with a cubic order polynomial and considering the cell first and second neighbors by vertex. New face values and gradients are computed and compared with the values from the convection and diffusion schemes. One way to do this is by



(a) Cell sets by face.

(b) Cell sets by vertex.

Figure 4.1: Different examples of cell sets.

recomputing new residual values, which indicate if the values satisfy the governing equations. The Residual Least Squares (RLS) vector for each cell is computed by the following formula:

$$\mathbf{E}_R = \frac{\sum_{f=1}^{F} U_f^n \mathbf{u}_f - \sum_{f=1}^{F} \nu_f (\nabla \mathbf{u})_f . \mathbf{S}_f + \frac{V_P}{\rho} \nabla p^n}{a_p} \tag{4.20}$$

where the values $\mathbf{u}_f$ and $(\nabla \mathbf{u})_f$ are computed with the cubic regression, a_p is the matrix value used for the momentum equations, which is required to give the RLS error estimator the same dimensions of the velocity or the transported variable. This error estimator gives an indication of the local error, if the differences between the cubic profile and the numerical discretization will affect the governing equations.

4.3.4 Grid Interface Correction

It is known from the literature that the interface between cells with different levels of refinement causes a degradation in the grid quality and consequently in the local numerical solution. The main problem is that, prior to cell refinement, the estimation of the error augmentation is not possible. To solve this problem, a grid interface correction was used in the adaptive algorithm.

The selected cells for refinement are added to the same group if they are neighbors of each other. Each of these groups has an extended layer of two cells since there is always an increase of the numerical error near the grid interface. If in one of these

groups, there exist cells with different levels of refinement, so only the cells with the lowest level of refinement are selected. This procedure increases the number of the required refinement steps but prevents the accumulation of the grid interface in the same location, resulting in grids with a lower number of cells for the same error level.

Chapter 5

Results on Adaptive Grids

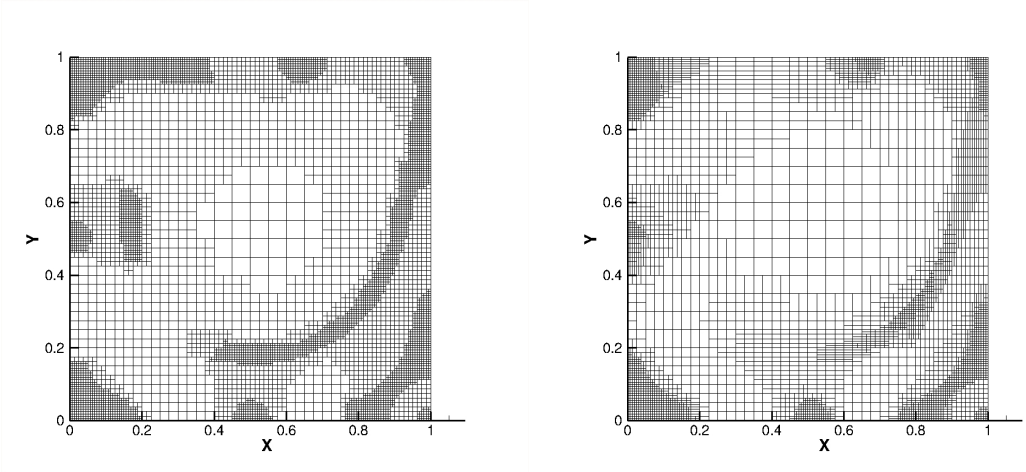
The following chapter comprises the obtained results on adaptive grids. It is divided in three sections: the first one has the adaptive grids driven by the relative error estimators which are suitable for multi-scale problems; the second one shows a small number of test cases with the Jasak adaptive algorithm for absolute error estimators; and finally the third one shows the results obtained with the proposed adaptive by the author algorithm that uses information provided by the error estimators without user defined constants.

5.1 Relative Error Estimators

5.1.1 Benchmarking Results with the adaptive method

To study the dR^2 criterion capabilities, an adaptive grid computation was done for the case of the cavity flow, it starts with a 10×10 grid and refines the cells with values higher than 0.1. After four refinement levels, we compare the results obtained for the isotropic and anisotropic criteria with the benchmark results of Botella and Peyret (1998) which were obtained with an uniform mesh of 160×160 (note that the equivalent uniform mesh for the adaptive grids has the same grid size). Figures 5.1(a) and 5.1(b) show the isotropic and anisotropic grid obtained, respectively.

The isotropic grid has 6910 cells and the anisotropic grid has 4407 cells. Figure 5.2 compares the results obtained with both meshes and the benchmark solution. The differences between the curves of both grids are due to the anisotropic grid and has lower quality than the isotropic one, although both results are close to the benchmark solution. The isotropic and anisotropic have approximately less 73% and 82.8% cells than the grid used by Botella and Peyret (1998).



(a) Isotropically refined mesh: 6910 cells. (b) Anisotropically refined mesh: 4407 cells.

Figure 5.1: Lid-driven cavity flow: isotropic and anisotropic mesh examples dR^2.

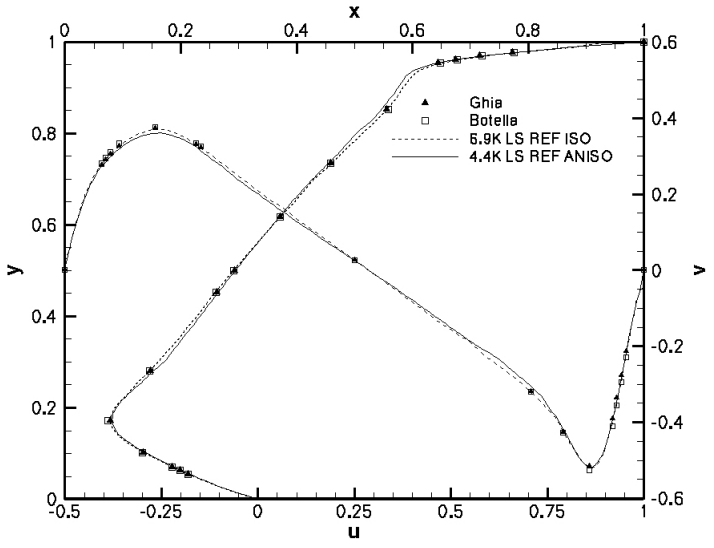
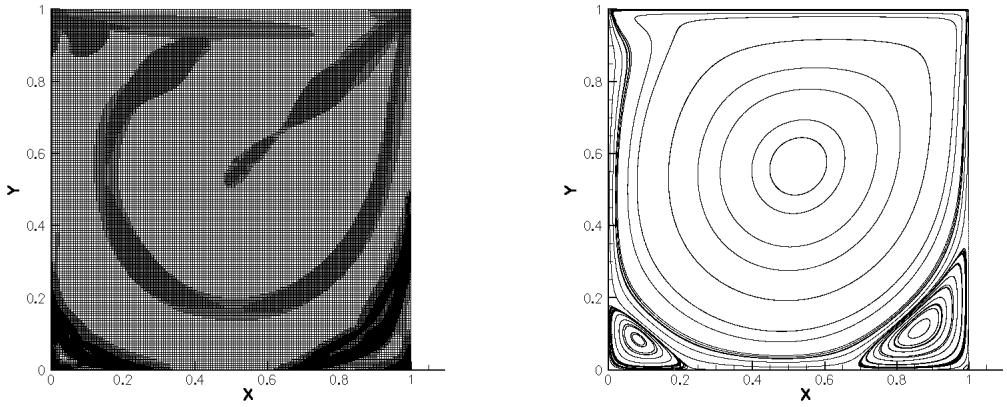


Figure 5.2: Comparison of the velocity profiles between benchmark values with adaptive refinement dR^2.

5.1.2 Results for the adaptive mesh with lid-cavity

The dR^2 criterion is sensitive to local maximums and minimums because of the low correlation at these points. In flows such as the cavity flow, this is a useful feature. We have applied this criterion with a threshold of 0.01, starting from a grid of 10×10 and figure 5.3(a) shows the grid after 15 refinement levels. It is evident that large savings are achieved with the refinement. Figure 5.3(b) shows the flow streamlines for $Re = 1000$. The grid is aligned with the flow, the boundary layers are well resolved, as well as the vortex core, the separation and attachment points. Out of the core of the secondary vortex, a belt of refined grid accompanies the inflexion of the streamlines, right where they have more pronounced curvature.



(a) Mesh. (b) Streamlines.

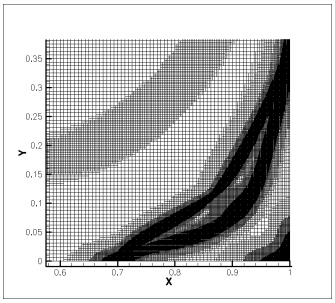
Figure 5.3: Lid-driven cavity flow: final field and mesh after 15 refinement levels.

The adaptive computation of the lid-driven cavity flow allows the corner vortices to appear with successively refinement levels. The dR^2 criterion is a logical choice for this application, as it is totally invariant in relation to scale.

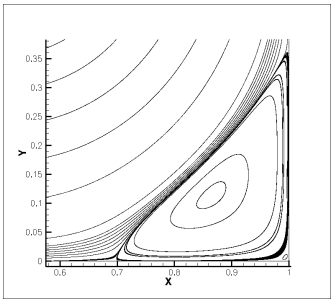
For nomenclature purposes, we will refer to the main cavity vortex as vortex number 1 and corner vortices number 2, 3, 4, 5, right or left vortices, *e.g.* vortex $3R$ denoting 3^{rd} vortex on right corner.

Figures 5.4(a) to 5.4(h) show 4 sequential zooms of the mesh and streamlines on the right corner vortices chain. The exponential decay in vortex size and strength can be qualitatively reduced from the figure zooms and obviously attention should be given to the coordinates values. Figures 5.5(a) to 5.5(h) show 4 zooms of the mesh and streamlines on the left corner vortices chain.

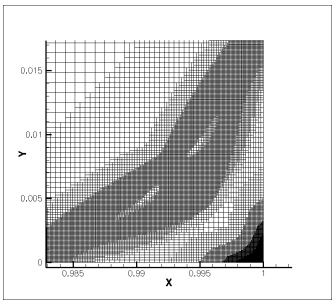
Table 5.1 lists the vortices center locations, the attachment and separation points



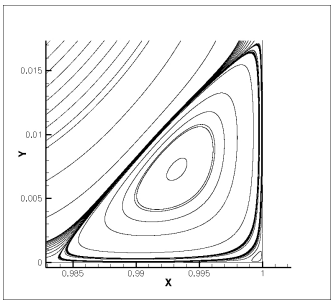
(a) Cavity right corner mesh zoom 1.



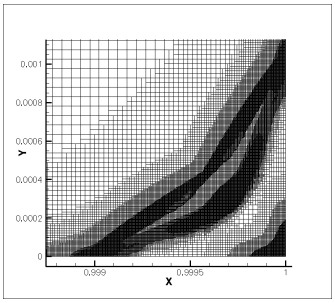
(b) Cavity vortex 2R.



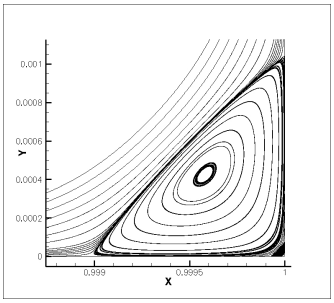
(c) Cavity right corner mesh zoom 2.



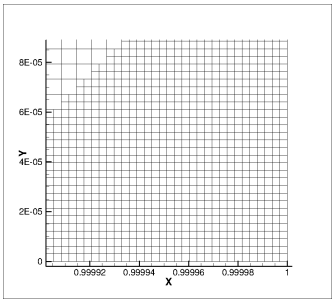
(d) Cavity vortex 3R.



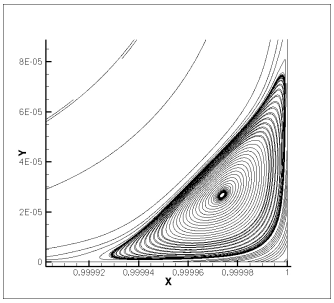
(e) Cavity right corner mesh zoom 3.



(f) Cavity vortex 4R.

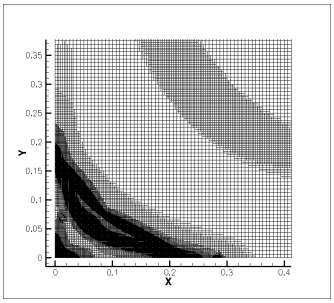


(g) Cavity right corner mesh zoom 4.

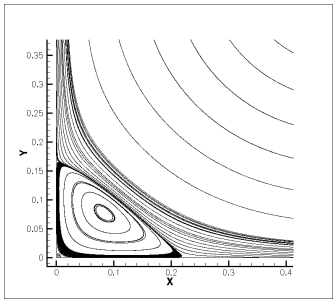


(h) Cavity vortex 5R.

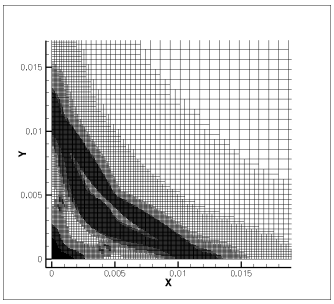
Figure 5.4: Right corner vortices mesh and streamlines, sequence of vortices.



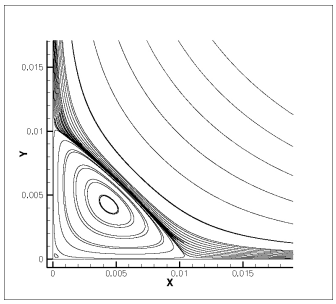
(a) Cavity left corner mesh zoom 1.



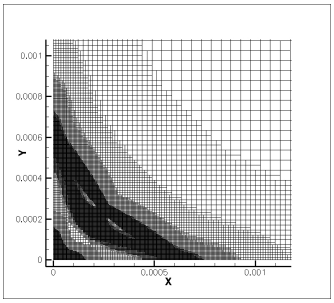
(b) Cavity vortex 2L.



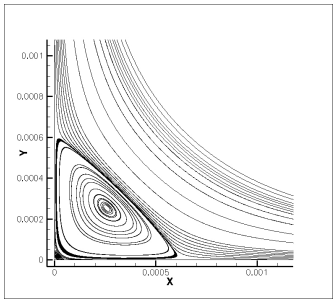
(c) Cavity left corner mesh zoom 2.



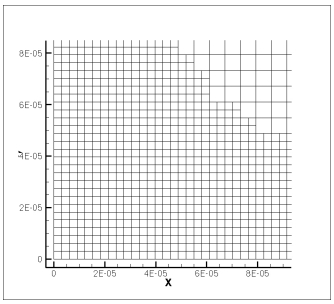
(d) Cavity vortex 3L.



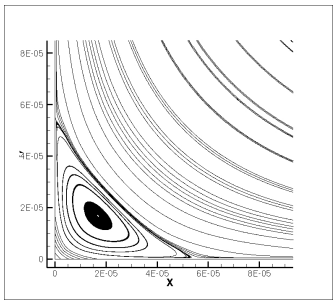
(e) Cavity left corner mesh zoom 3.



(f) Cavity vortex 4L.



(g) Cavity left corner mesh zoom 4.



(h) Cavity vortex 5L.

Figure 5.5: Left corner vortices mesh and streamlines, sequence of vortices.

and also the vorticity magnitude. Figure 5.6 explains the vortices nomenclature used.

Vortex	C_x, C_y	Vorticity	X,Y Separation	Figure
1	0.53051,0.56531	-2.02727	—	5.3(b)
2R	0.86576,0.11364	0.822856	0.69495,0.36817	5.4(b)
3R	0.99323,0.00727	-0.00484733	0.98339,0.01759	5.4(d)
4R	0.99958,0.00042	3.50858E-5	0.99895,0.00107	5.4(f)
5R	0.99997,2.68E-5	-3.71581E-7	0.99991,8.25E-5	5.4(h)
2L	0.08317,0.07708	0.386846	0.22335,0.16800	5.5(b)
3L	0.00441,0.00424	-0.00198701	0.01068,0.01038	5.5(d)
4L	0.0025,0.0025	1.2416E-5	0.00065,0.00064	5.5(f)
5L	1.69E-5,1.69E-5	-1.42392E-7	5.530E-5,5.573E-5	5.5(h)

Table 5.1: Cavity vortices parameters.

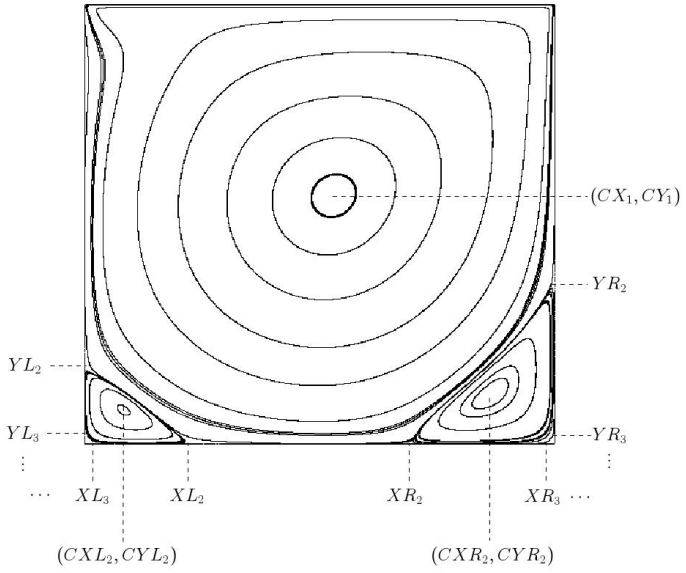


Figure 5.6: Lid-driven cavity: general layout, nomenclature.

The main cavity vortex is not exactly at the center of the cavity but slightly above this point, at $(0.530, 0.565)$, but the corner vortices are virtually centered at the cavity diagonals, see Table 5.1. The predicted vorticity decay is very strong with each consecutive vortex having less than one percent of the vorticity of the previous one. The vortices detachment length on the bottom and side walls decreases approximately with a contraction ratio of 5.0×10^{-2}.

Figures 5.7 and 5.8 show the absolute velocity along the cavity diagonal (1,1) to (0,0) and diagonal (1,0) to (0,1) as a function of the distance along the diagonal and the *log* scale used can distinguish the eddies sequence that spread over 16 orders of velocity

magnitude. The main vortex is displayed approximately from 1 to 0.1 and between 0.1 to 0.0 the sequence of vortices is shown. The length ratio between the secondary vortex and the main vortex is 0.1, and the velocity ratio is 4×10^{-2}; for the quaternary vortex, the length ratio is approximately 6×10^{-4} and the velocity ratio is 5×10^{-4}. These results are demanding from the numerical point of view and indeed are on the limit of machine precision. The spectral radius of the coefficient matrix is extreme because of the range of cell sizes, making this a difficult case to converge. Nevertheless, it can be seen that the signature of the 5^{th} vortex on the left and right corner is present in the results, despite off $O[10^{-16}]$. The last fifth vortex is not completely resolved. Any further refinement was not presented because the velocity magnitude is of the order of 10^{-16} and the truncation error saturates the solution. The proposed error criterion is able to detect phenomena over a wide range of scales without being overwhelmed by dynamically dominant phenomena.

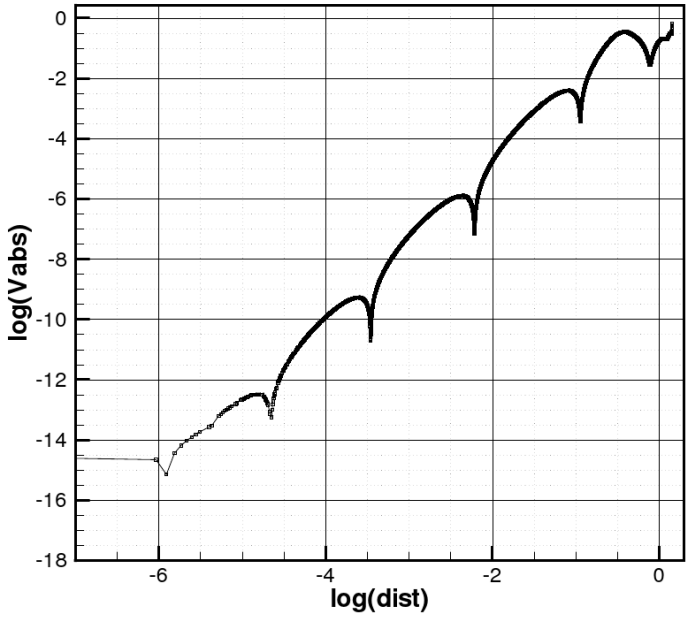


Figure 5.7: Lid-driven cavity: log plot of absolute velocity in the diagonal (0,0) to (1,1). The distance is measured along the diagonal starting from (0,0) and marked as the power of 10.

According to Moffatt (1963), for a Reynolds number less than one based on the distance from the corner the inertia forces are negligible and the Stokes flow solution displays multiple eddies of decreasing size and rapidly decreasing intensity. All eddies (for a given corner angle) are geometrically and dynamically similar but with successive changes of length and velocity scales given by the factors present by Moffatt (1963). For sharp corners with 90 degrees, the factors are 0.0614 and 4.8566×10^{-4}. The

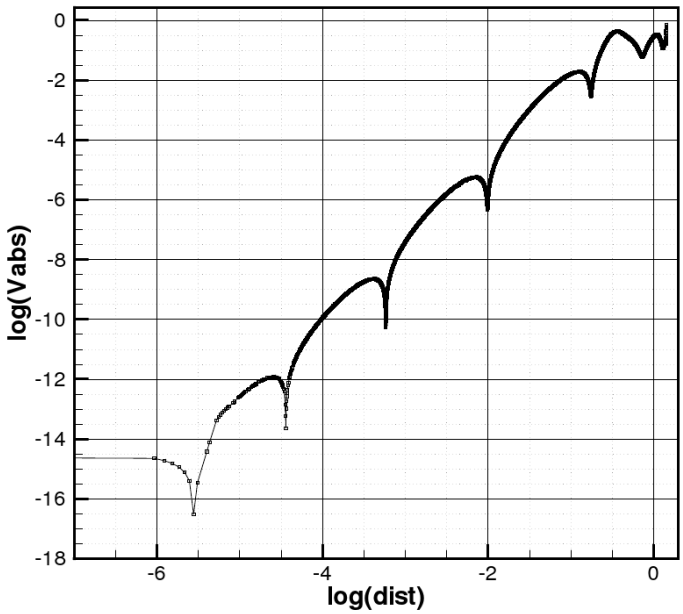


Figure 5.8: Lid-driven cavity: log plot of absolute velocity in the diagonal (1,0) to (0,1). The distance is measured along the diagonal starting from (1,0) and marked as the power of 10.

present prediction obtained with a general error estimation procedure and without any particular adaptation to capture very small eddies in the flow, gave the length and velocity factors of 0.06001 and 4.5×10^{-4}.

The prediction display a ratio of the dimensions of third and fourth successive eddies, 0.065 and 0.054. The sixth right vortex center will be at a distance from the corner 3.0226×10^{-6} and will require a minimum mesh spacing of 1.907×10^{-7} for a similar accuracy of the present results.

Table 5.2 lists other parameters to study the successive vortices in the cavity corners, where R_t is the lowest distance between the vortex center and boundary, V_t is the tangential velocity, "Ratio" stands for the geometric ratio of the separation points of the vortex X/Y, the Reynolds number $Re_{vortex} = \dfrac{V_t R_t}{\nu}$ and the V_t decay factor is the ratio of V_t between successive corners vortices. The predicted tangential velocity decay 4.3×10^{-4} is not far from the analytical value 4.8566×10^{-4} of Moffatt (1963). The vortices Reynolds number are very small $O(10^{-14})$, meaning that it will be very difficult to visualize them with experimental techniques.

When the calculations at the last refinement level comprised 2×10^5 cells, the smallest mesh size is equal to 3.05×10^{-6}, meaning that the uniform mesh with the smallest mesh size would have 1.1×10^{11} cells and consequently, the adaptive mesh contains 0.00018% of the uniform mesh.

Vortex	R_t	V_t	Re_{vortex}	Ratio	V_t decay
2R	5.2316E-2	6.85E-2	3.58371	0.82856	-
3R	4.09408E-3	2.88338E-5	1.18048E-4	0.94429	4.2093E-4
4R	2.45427E-4	1.22629E-8	3.00965E-9	0.98131	4.25296E-4
5R	1.0457E-5	5.44273E-12	5.69146E-14	1.09091	4.438371E-4
2L	4.23247E-2	2.22562E-2	9.41986E-1	1.32946	-
3L	3.45196E-3	1.15166E-5	3.97548E-5	1.0289	5.174558E-4
4L	2.25305E-4	5.19926E-9	1.17142E-9	1.01563	4.514579E-4
5L	1.2315E-5	2.74822E-12	3.38443E-14	0.95101	5.285791E-4

Table 5.2: Cavity vortices aditional parameters.

5.1.3 Relative Error Estimator - Final Remarks

A Finite-Volume 2D Navier-Stokes solver was developed for unstructured polyhedral hybrid adaptive meshes with second order accurate spatial discretization. A new error estimation criterion was developed for adaptive space discretization (h-refinement).

The algorithm was applied to the lid-driven cavity flow at $Re = 1000$ and apart from the main cavity vortex the corner vortices chain was fully resolved up to the fourth cavity corner eddy and the fifth vortex was partially resolved. The mesh adaptation was fully driven by the mesh refinement criterion that allocated meshes on the cavity corners without any external prescription.

The main cavity eddies were well resolved and the velocity predictions are virtual identical to the benchmark data, Ghia et al. (1982) and Botella and Peyret (1998). The small eddies in the creep flow sharp corner region were also quantitatively very well captured in agreement with Moffatt (1963). The new error estimation criterion and the second order developed Navier-Stokes solver have allowed to calculate in detail the cavity eddies whose characteristic length scales are five order of magnitude lower and sixteen order in tangential velocity intensity. On a macroscopic inspection of the grid, the criterion did not present preferences for large scale in detriment of small-scale phenomena.

The author is not aware of any benchmark results about these corner vortices and this may constitute the first time these vortices were resolved from the full solution of the Navier-Stokes equations without additional theoretical assumptions or forced the location of meshes on the corner regions.

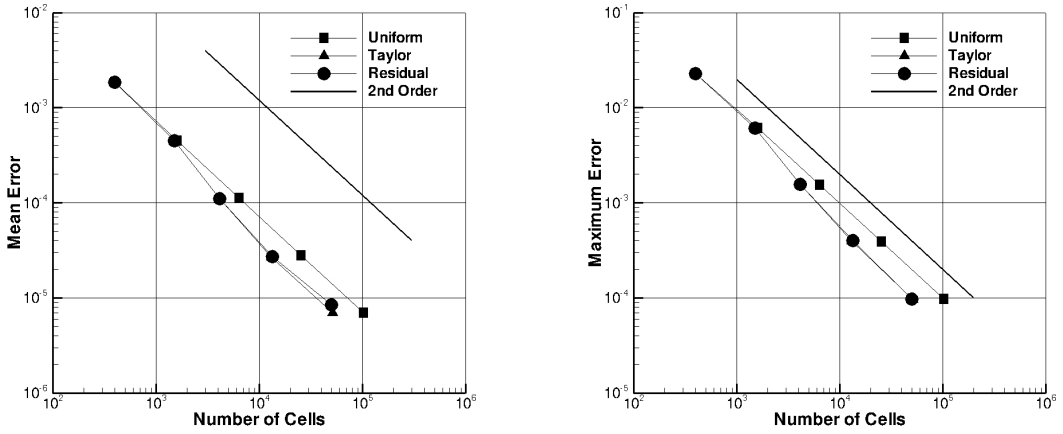
5.2 Absolute Error Estimators - Jasak Method

5.2.1 Poisson Equation - Manufactured Analytical Solution

For this test case, the Poisson equation $\nabla^2\phi = S_\phi$ was solved in a square computational domain $[0,1] \times [0,1]$. Dirichlet boundary conditions are prescribed in all boundaries, the Laplacian of the analytic solution is prescribed as the source term and it is given by:

$$\phi(x,y) = exp\left(-\frac{(x-0.5)^2 + (y-0.5)^2}{0.1^2}\right) \tag{5.1}$$

The computations start with a cartesian grid of 20×20 and λ is equal to 0.1 in the adaptive grids. Figure 5.9 shows the mean and maximum analytic error for an uniform and an adaptive quadrilateral grids for this solution.



(a) Mean error over number of cells.

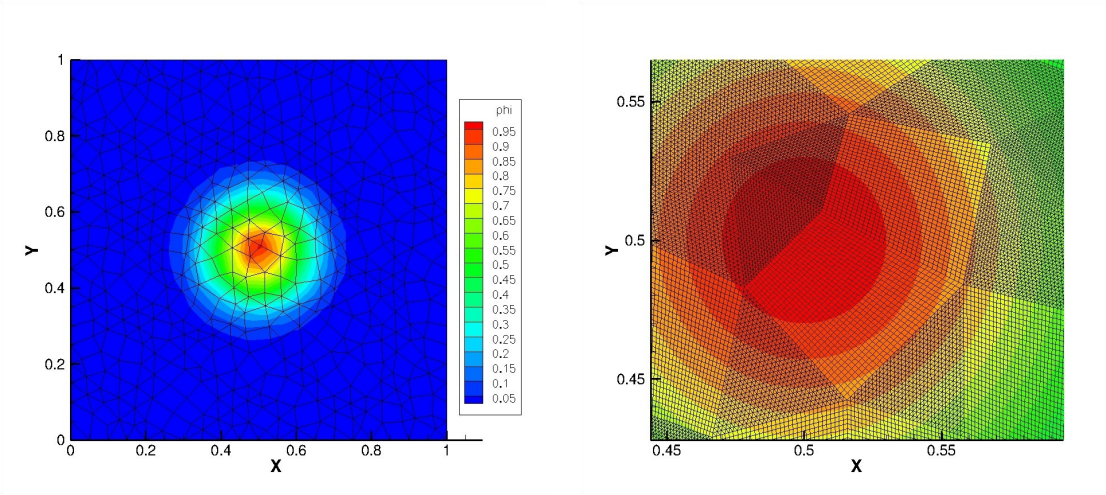
(b) Maximum error over number of cells.

Figure 5.9: Poisson equation: mean and maximum error over number of cells for uniform and adaptive grids.

Both the mean and maximum errors have second order decay for the uniform grid and the adaptive grids present second order decay after three levels of refinement. The adaptive grids keep 51% of the number of cells with a uniform grid.

The same study is done for the case of a hybrid grid composed by quadrilateral and triangles, and figure 5.10(a) shows the initially grid used which has a hydraulic diameter $h = 0.05$. Figure 5.10(b) shows a zoom from the adaptive grid after 5 levels of refinement using the Residual Least Squares criteria ($\lambda = 0.5$), with the refined cells located at the domain center.

The figure 5.11 shows the mean and maximum errors obtained with uniform and



(a) Initial grid and solution distribution.

(b) Grid after 5 levels of refinement RLS zoom.

Figure 5.10: Hybrid mesh: poisson solution for initial grid and adaptive grid.

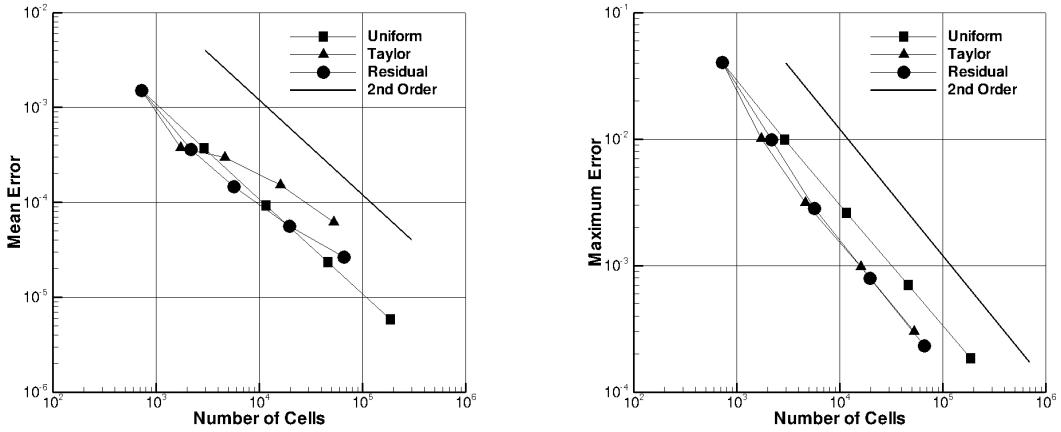
adaptive grids for each level of refinement, and the results of the uniform grid show the expected second-order convergence. Initially, the mean error of the adaptive grids decreases faster than for the uniform grid, but after the second level of refinement the reductions slows down and the mean error starts to be higher than the uniform grid, see figure 5.11(a).

This happens because the adaptive grids have worst grid quality, due to mesh skewness, cell non-orthogonality and cell distortion than the uniform grids. The characteristics of the initial grid are conserved during h-refinement as could be observed in the figure 5.10(b), which is the cause of the mesh induced errors and the higher mean error in the adaptive grids.

5.2.2 Convective-Diffusive Equation - Scalar Transport

The solution of the scalar transport equation for a imposed temperature profile is computed in a rectangle computational domain of $[0.1, 1.0] \times [0.0, 1.0]$. This problem has been used previously by Volker (2000) and Juretic (2004). A Dirichlet boundary condition is applied in all boundaries, except for $x = 1.0$ where a Neumann condition with null gradient is imposed. The analytic solution is given by:

$$T(x, y) = A \sum_{k=1}^{\infty} (\alpha_k \phi_{1k}(x) + \beta_k \phi_{2k}(x)) \psi_k(y) \tag{5.2}$$



(a) Mean error over number of cells.

(b) Maximum error over number of cells.

Figure 5.11: Hybrid mesh: mean and maximum error over number of cells uniform and adaptive grids.

where ϕ_{1k}, ϕ_{2k} and $\psi_k(y)$ are functions:

$$\phi_{1k} = \frac{sinh(b_k(1-x))\, e^{0.5xPe}}{sinh(b_k)} \tag{5.3}$$

$$\phi_{2k} = \frac{sinh(b_k x)\, e^{0.5(1-x)Pe}}{sinh(b_k)} \tag{5.4}$$

$$\psi_k(y) = sin(k\pi y) \tag{5.5}$$

and α_k, β_k and b_k are constants:

$$\alpha_k = \frac{2}{k\pi}(cos(0.45k\pi) - cos(0.55k\pi)) \tag{5.6}$$

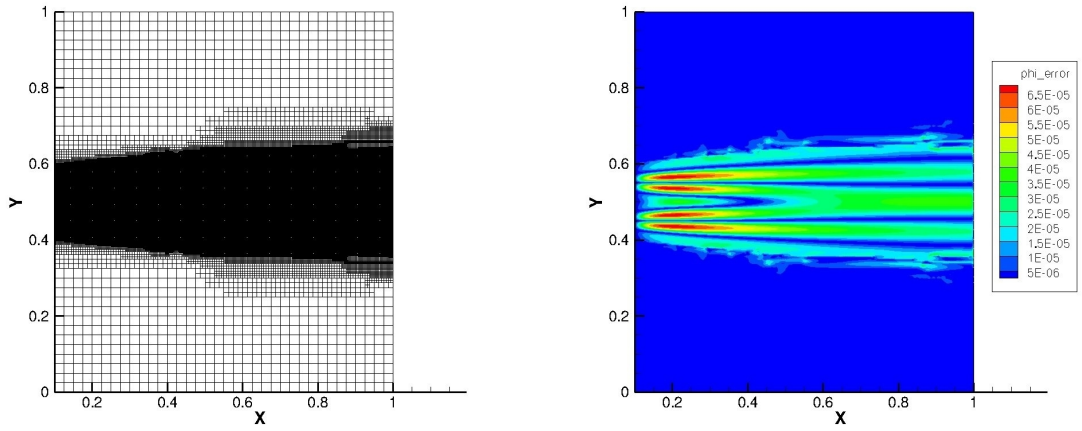
$$\beta_k = \alpha_k \frac{b_k\, e^{0.5Pe}}{0.5Pe\, sinh(b_k) + b_k\, cosh(b_k)} \tag{5.7}$$

$$b_k = \sqrt{(0.5Pe)^2 + (k\pi)^2} \tag{5.8}$$

ν is set to 0.001 $[m^2/s]$, the reference length L is set to 1.0 $[m]$, the velocity u is equal to 1.0 $[m/s]$ and the Peclet number $Pe = U|L/\nu = 1000$. Different simulations are done with uniform and adaptive grids to compare the error estimators performance. A grid with 36×40 cells is used as an initial grid.

Figure 5.12(a) shows the grid obtained after 5 levels of refinement using the Residual criteria and figure 5.12(b) shows the analytic error distribution. The higher error values

are located in the most refined cells.



(a) Adaptive grid after 5 levels of refinement Scalar.

(b) Error distribution in an adaptive grid.

Figure 5.12: Convective-diffusive equation: adaptive grid and error distribution after 5 levels of refinement with RLS.

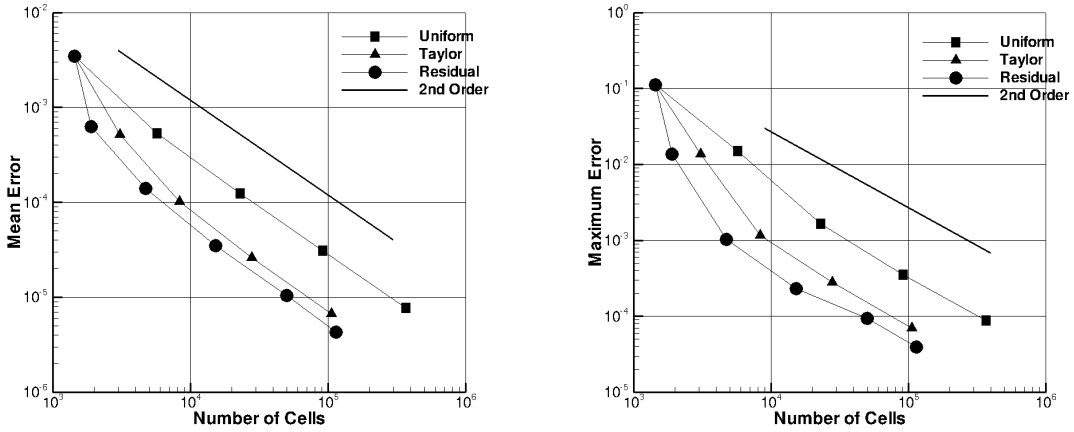
Figure 5.13 shows the mean and maximum error reduction with the number of cells for the three types of grids. In both, mean and maximum error the reduction rate is higher in the adaptive grids than in the uniform grids, after the first two levels of refinement the error curves have a second order reduction. The Residual criteria curve is located bellow the Taylor criteria in both figures 5.13(a) and 5.13(b). For the same error value, the Residual adaptive grid has less 10% cells than the uniform grid.

5.3 Absolute Error Estimators - Proposed Method

5.3.1 Analytic 2D Cavity

To study the numerical error from the FVM discretization at the adaptive grid interface, the analytic 2D cavity problem was selected. The second-order accuracy of the overall numerical scheme is verified for a prescribed mesh, which is shown in figure 5.14(a). From this starting grid, all cells are refined three times for each pair of numerical schemes: central difference diffusive and linear interpolation convective schemes which are typically used in cartesian grids (D-CDS,C-LIN)[1]; tangential correction diffusive and linear interpolation with gradient correction convective schemes

[1]The first D or C means that the respective scheme is a diffusive or a convective one.



(a) Mean error over number of cells.

(b) Maximum error over number of cells.

Figure 5.13: Convective-diffusive equation: mean and maximum error over number of cells for uniform and adaptive grids.

(D-TC,C-LIN2); and the face least squares diffusive and convective schemes (D-FLS,C-FLS).

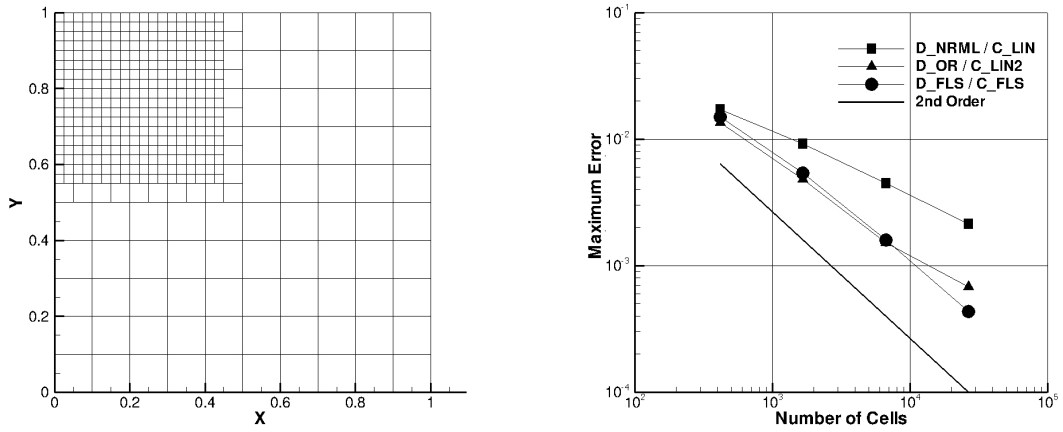
Figure 5.14(b) shows the maximum error for the three pairs of schemes used in this interface study. The mean error has second order reduction for the three schemes sets, being D-FLS,C-FLS the one with the lowest error. The maximum error has also second order reduction except for the D-NRML,C-LIN schemes which is close to first order. From these results the lowest numerical error was obtained for the D-FLS,C-FLS schemes. These are the selected schemes in the future subsections of this work.

5.3.2 Laplace Equation in a L-Shaped Domain

For this test case, the Laplace equation $\nabla^2\phi = 0.0$ was solved in a L-shaped domain $[-1,1]^2 \setminus ([0,1]\times[-1,0])$ Dirichlet boundary conditions are prescribed in all boundaries and the analytic solution is given by the following equation:

$$\phi(x,y) = r^{2/3} sin(2\varphi/3) \quad with \;\; (x,y) = r(cos\,\varphi, sin\,\varphi) \tag{5.9}$$

The computations started with a cartesian grid of 12 cells and three types of refinement are applied to this grid: one with uniform refinement and other two with the adaptive algorithm using the classic Taylor series and the RLS as error estimators. The goal is to study the main differences between the two error estimators and evaluate their effectiveness. Figure 5.15 shows the mean and maximum error for the three types



(a) Initial grid used in the interface study.

(b) Maximum error over number of cells.

Figure 5.14: Analytical cavity: initial grid and maximum error over number of cells for three pairs of diffusive and convective schemes.

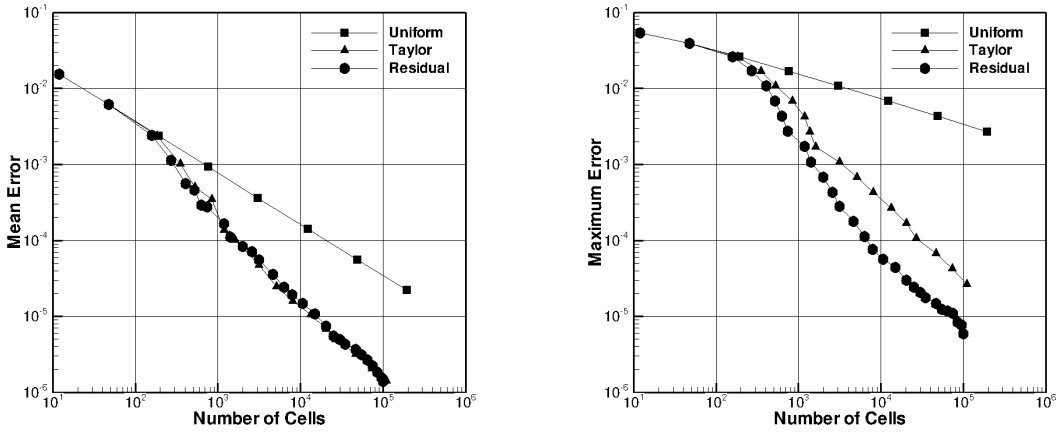
of grids, after 15 levels of refinement for the Taylor series and 22 levels of refinement for the RLS errors estimators.

For the uniform grid case, the mean and maximum error slope have an order of 4/3 and 2/3, respectively. Although, at the singularity point $(x, y) = (0, 0)$ the analytic solution is zero and the analytic gradient is infinite which causes the method to have an order accuracy lower than 2 for the uniform grids.

For both error estimators, the mean error of the adaptive grids has second order error decay. The different error slopes of the adaptive and uniform grids explain why the mean error for the two adaptive grids is much lower than the mean error for the uniform grid (more than 10 times).

Figure 5.15(b) shows the maximum error for the three cases studied. The adaptive grid with the TS estimator has a maximum error 100 times lower than the error of the uniform grid and the adaptive grid with the RLS estimator shows an improvement when compared with TS estimator by having a maximum error 1000 times lower. The ratio between the maximum and mean error, which is a measure of the adaptivity efficiency, is 0.053 for the TS estimator and 0.2336 for the RLS estimator.

Figure 5.16 shows the two final adaptive grids obtained with the TS and RLS error estimators. The adaptive grid with TS has more refined cells and the refined region has a circular pattern. This happens due to the loss of accuracy, after some adaptive levels, of the TS error estimator. The adaptive grid with RLS has a lower number of cells and a rectangular pattern is observed with an over estimation of the numerical



(a) Mean error.

(b) Maximum error.

Figure 5.15: Poisson equation: mean and maximum error over number of cells for uniform and adaptive grids (quadrilateral grid example).

error in the boundaries of the computational domain, see figure 5.16(b).

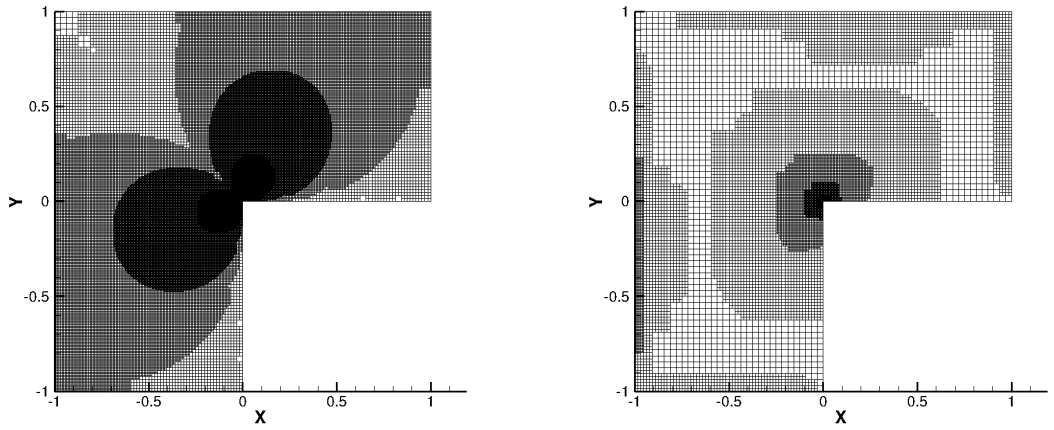
The same example is done with triangles with an initial grid with 12 cells. The mean and maximum error for the three types of grids are shown in figure 5.17. For the same number of cells, the error in the triangle based grids are slightly higher than the quadrilateral based grids. From the comparison between the adaptive and uniform grids, the same conclusions from the previous case can be drawn.

5.3.3 Convective-Diffusive Equation - Point Source in Cross-Flow

The convective-diffusive equation $U\partial\phi/\partial x = \Gamma_\phi \nabla^2\phi$ is frequently used for benchmarking, see for example Jasak and Gosman (2000b). The used analytical solution is given by:

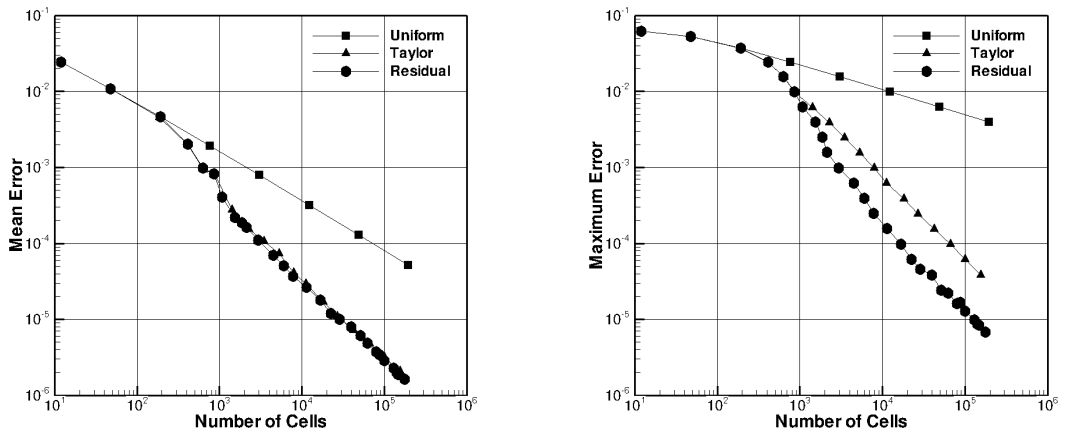
$$\phi(x,y) = \frac{S}{2\pi\Gamma} K_0\left(\frac{U\sqrt{x^2+y^2}}{2\Gamma}\right) e^{(0.5xU/\Gamma)} \tag{5.10}$$

where $S = 16.67\,[\phi/s]$ is the source magnitude, $\Gamma = 0.05\,[m^2/s]$ is the diffusive coefficient, $U = 1.0\,[m/s]$ is the imposed velocity in the x axis and K_0 is the modified Bessel function of second kind and zero order. This problem is solved in a rectangular domain $[0.0, 4.0] \times [-0.5, 0.5]$ and the line-source origin is located at $0.05\,m$ from the left boundary to avoid numerical problems from the singularity point. Dirichlet boundary conditions are prescribed in all boundaries, except for the right boundary ($x = 4.0$),



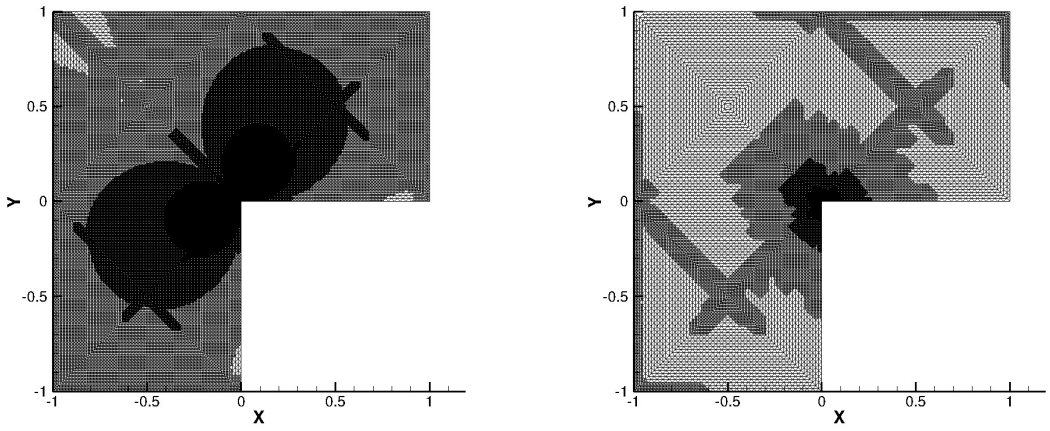
(a) Adaptive grid using the Taylor estimator. (b) Adaptive grid using the RLS estimator.

Figure 5.16: Poisson equation: adaptive grids with square cells with the Taylor and RLS estimators.



(a) Mean error. (b) Maximum error.

Figure 5.17: Poisson equation: mean and maximum error over number of cells for uniform and adaptive grids (triangle grid example).



(a) Adaptive grid using the Taylor estimator. (b) Adaptive grid using the RLS estimator.

Figure 5.18: Poisson equation: adaptive grids with triangle cells after 15 levels of refinement with the Taylor and RLS estimators.

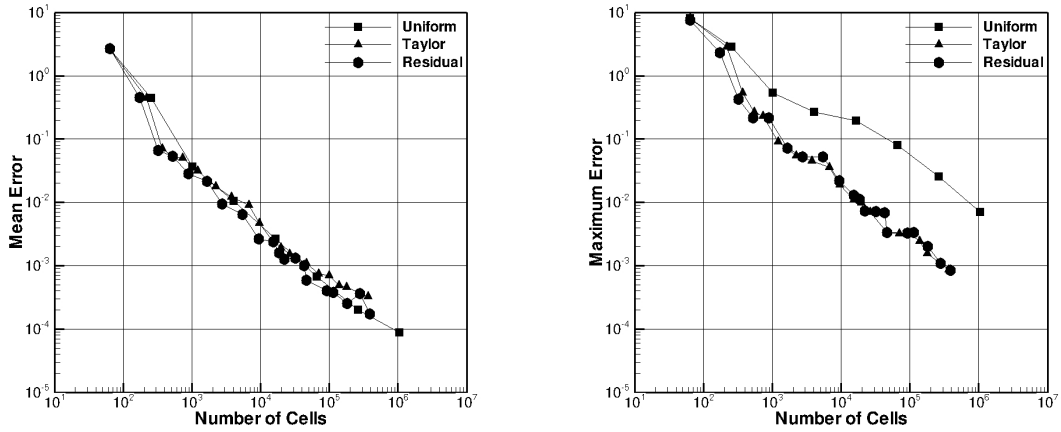
where a null gradient is imposed.

The same refinement test was done for this solution. The starting grid contains 16×4 cells and 20 levels of refinement are considered. Figure 5.19 shows the mean and maximum error for the uniform and adaptive grids.

The mean error (figure 5.19(a)) displays the same slope for the three grids and the adaptive grids do not show improvements in the mean error when compared with the uniform grid. The case with TS has a mean error curve in an upper position relatively to the RLS and uniform cases.

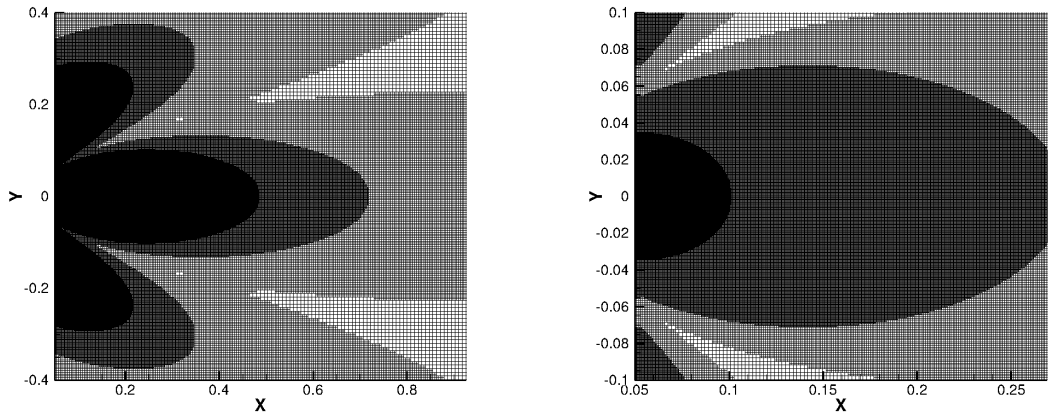
However, the adaptive grids have a lower maximum error than the uniform grid case. The final adaptive grid has a maximum error 10 times lower than the uniform grid. The RLS has a better efficiency than the TS, but the differences are smaller than in the previous case.

The error slope is not always constant due to the grid interface correction, which prevents the accumulation of the grid interfaces between different levels of refinement and avoids the loss of the grid quality and the solution overall accuracy. Figure 5.20 shows two different zooms of the grid obtained with the RLS estimator after 20 refinement steps. The cell zones with different refinement levels are well defined due the grid interface algorithm which prevents the accumulation of refined cells from the previous refinement steps. The present results are a significant improvement over Jasak and Gosman (2000b) because of the grid interface correction and the different refinement decision algorithm that creates much smoother adaptive grids with high quality.



(a) Mean error. (b) Maximum error.

Figure 5.19: Line source: mean and maximum error over number of cells for uniform and adaptive grids.



(a) Zoom 1. (b) Zoom 2.

Figure 5.20: Line source: adaptive grid obtained for the RLS estimator after 20 refinement steps.

5.3.4 Point Jet

The Point Jet analytic solution is obtained by freeing a point flow with a fixed momentum M_j in a large domain. This solution has a singularity point at the origin and consequently the computational domain was located $0.005\,m$ downstream of the jet origin to avoid accuracy problems from the singularity point. The computational domain $[0.0, 2.5] \times [0.0, 0.1]$ is used for these computations, and the domain is extended in the x direction to avoid numerical errors near the outlet boundary. Dirichlet boundary conditions are applied in all boundaries, except for $x = 2.5$ where an outlet boundary condition is applied. The initial grid is covered with 100×4 cells.

The velocity field of this solution is defined by equations (5.11) and (5.12):

$$u(x,y) = \frac{A}{B} x^{-1/3} sech^2(\frac{y}{B} x^{-2/3}) \tag{5.11}$$

$$v(x,y) = -\frac{A}{3} x^{-2/3} tanh(\frac{y}{B} x^{-2/3}) + \frac{2A}{3B} y x^{-4/3} sech^2(\frac{y}{B} x^{-2/3}) \tag{5.12}$$

where A, B and M_j are constants defined by:

$$A = (\frac{9}{2}\nu M_j)^{1/3} \quad B = (\frac{48\nu^2}{M_j})^{1/3} \quad M_j = U_0^2 h = 0.1 \ \left[m^3/s^2\right] \tag{5.13}$$

and ν is set to 0.0012 $[m^2/s]$.

Figure 5.21 shows the analytic error distribution and the grid after 14 levels of refinement using the Residual criteria. The error is distributed along the cells with different levels of refinement and the cells with higher error are located near the singularity of the problem, which is the zone with higher numerical error for an uniform mesh.

Figure 5.22 shows the mean and maximum error as a function of the number of cells. For each level of refinement, the three grids have approximately the same maximum error. Since the adaptive grid has less cells than the correspondent uniform grid, the curves from the adaptive grids are bellow the curve of the uniform grid, see figure 5.22(b).

5.3.5 Flow over a 2D Cylinder

The Navier-Stokes equations were solved for the steady flow over a $2D$ cylinder for a Reynolds number of 40. The Reynolds number for this flow is defined by $Re = UD/\nu$, where D is the cylinder diameter. The computational domain has an extension of 43

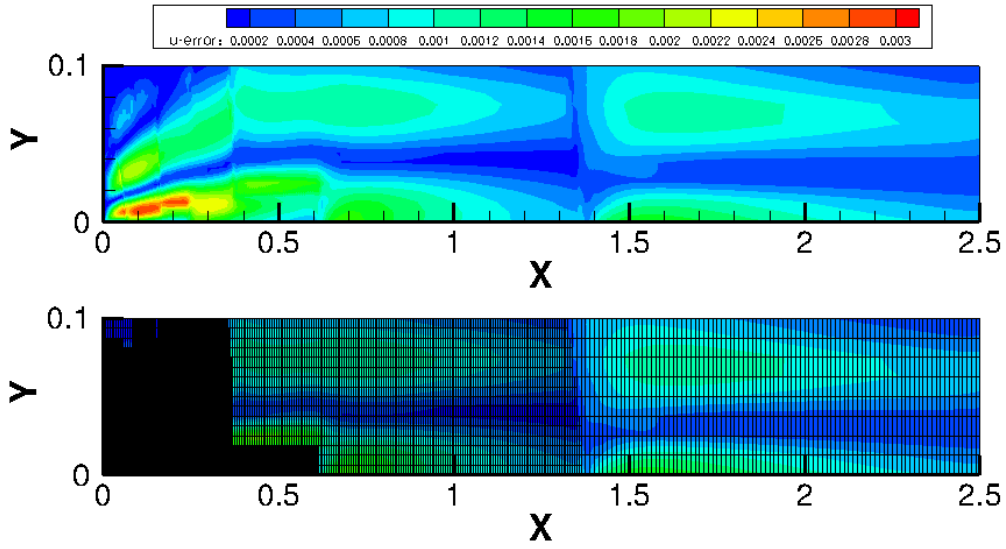
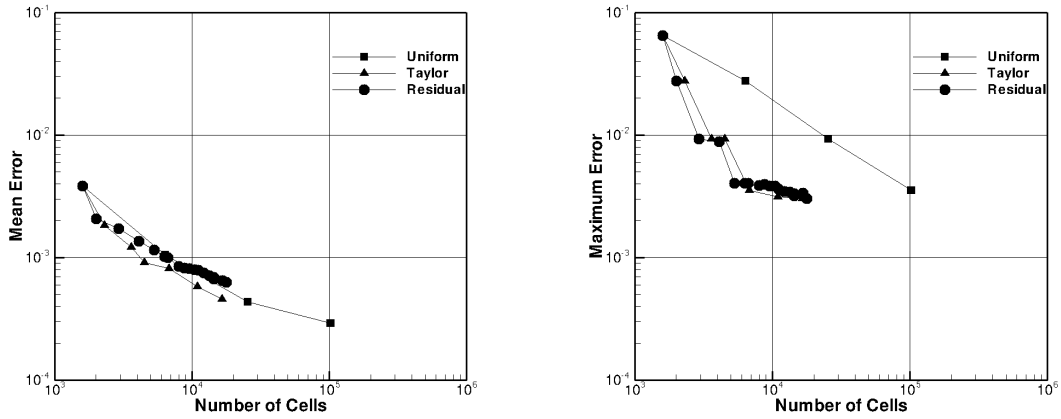


Figure 5.21: Point jet: grid obtained after 14 levels of refinement RLS.



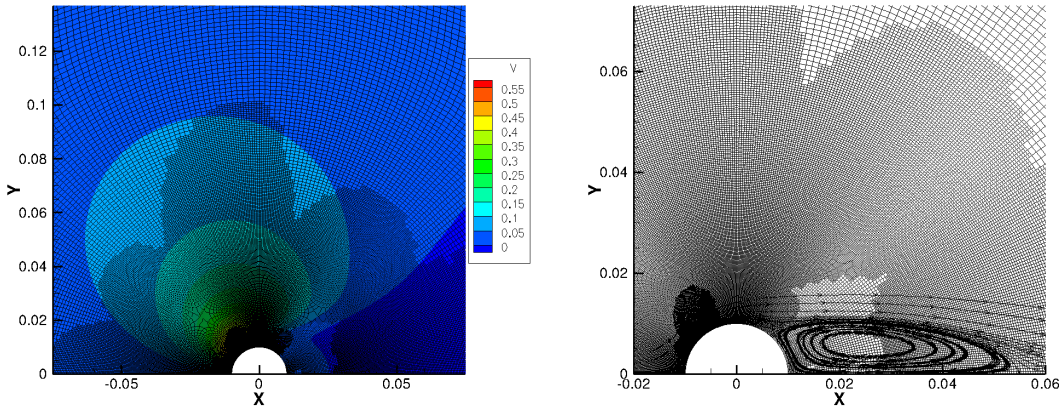
(a) Mean error.

(b) Maximum error.

Figure 5.22: Point jet: mean and maximum error over number of cells for uniform and adaptive grids.

diameter in the longitudinal direction with the cylinder located 13 diameters from the inlet. The domain extends up to 13 diameters in the vertical direction.

Figures 5.23(a) and 5.23(b) show respectively the v-velocity contour plot and the streamlines for the adaptive mesh obtained after 6 levels of refinement using the RLS error estimator. Different level sets of error distribution can be observed in this flow due to the different cell refinement levels. The refinement zone is located in a circle above the cylinder where the v velocity is maximum. This detail can be observed in the grid zoom (figure 5.23(b)). The flow rettachment length is equal to 2.21 D which is consistent with the value 2.18 D from Calhoun (2002).



(a) V-velocity contour plot.

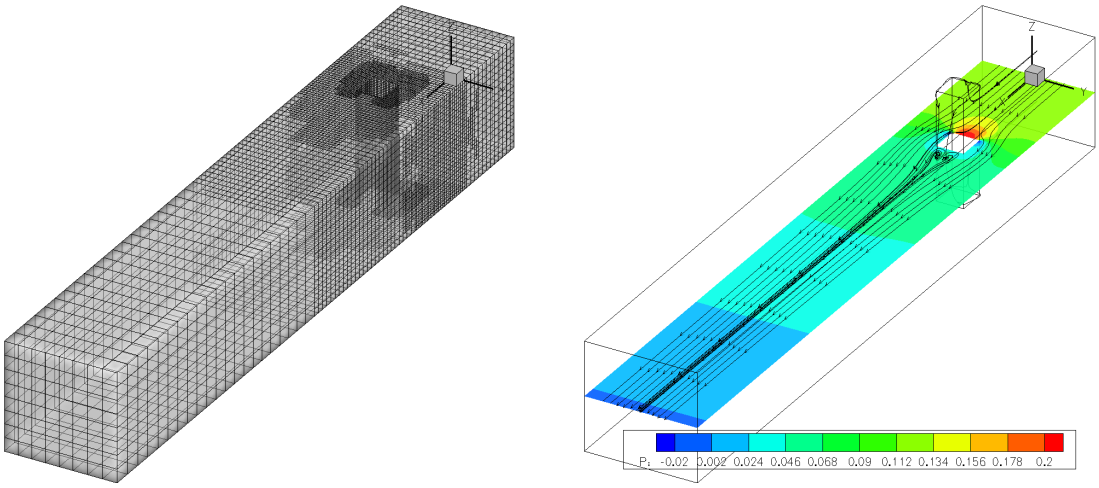
(b) Flow streamlines and adaptive grid zoom.

Figure 5.23: Cylinder flow: contour plot of the v-velocity and streamlines in the cylinder wake after 6 levels of refinement.

5.3.6 Confined 3D flow around a Squared Cylinder

The 3D Navier-Stokes equations were solved for the confined flow in a channel with a squared cylinder for a Reynolds number of 20. For this subcritical Reynolds number, the flow remains steady without the onset of the Karman sheet. This problem was previously studied by Schafer and Turek (1996); Braack and Richter (2006), where details of the computational domain and boundary conditions can be found.

The starting grid has 2912 cells and figure 5.24 shows the adaptive mesh obtained after 7 refinement levels with 83706 cells, where the solution streamlines and pressure contour plot can be observed. The adaptive grid divided the domain in four zones where the cells have different levels of refinement. The zone with the highest one includes the four squared corners which are considered singularities points of the problem.



(a) 3D view. (b) Pressure contour plot.

Figure 5.24: Squared cylinder: adaptive mesh after 6 levels of refinement - 3D view and pressure contour plot.

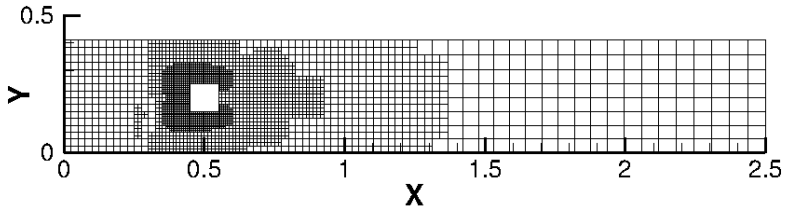


Figure 5.25: Squared cylinder: adaptive mesh after 6 levels of refinement - 2D plot.

5.3.7 Flow over a Sphere

The three dimension flow over a sphere at $Re = 200$ is computed as the last test of the RLS error estimator and the adaptive code. Two initial meshes were made one with hexahedrons and the other with tetrahedrons. The initial hexahedron grid has 46800 cells and its domain has a cylindrical form, and the computational domain of the initial tetrahedral grid is a squared prism with $39 \times 13 \times 13$ diameters comprising 126182 cells.

Figure 5.26 shows the adaptive grids for two levels of refinement, with both the hexahedral and tetrahedral grids. The final hexahedrical and tetrahedrical meshes have 1331216 and 2707026 cells, respectively, which corresponds to meshes with less 55.5% and 66.5% than the uniform refinement case. Both adaptive grids are refined near the sphere wall and in the flow's wake, which are the primary features of this problem. The cone formed by the refined cells in the flow's wake is bigger in the tetrahedral grid, since the cells far away of the sphere have a higher hydraulic diameter than in the hexahedral grid.

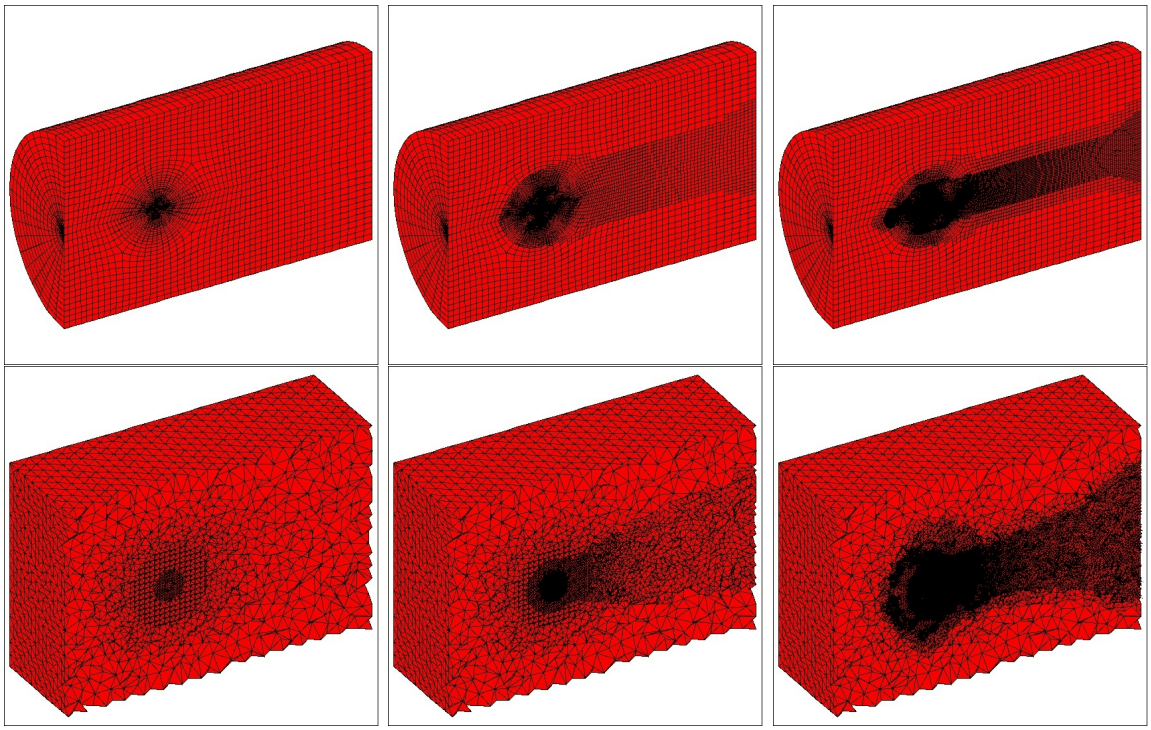


Figure 5.26: Sphere flow: example of adaptive refinement with hexahedral and tetrahedral grids.

5.3.8 Absolute Error Estimators - Final Remarks

The Residual Least Squares (RLS) error estimator has shown to be suitable for adaptive refinement of Finite-Volume methods on unstructured grids. The RLS error estimator is applicable to arbitrary unstructured grids and its calculation is strongly coupled to the governing equations and the grid quality.

The second-order FV discretization on adaptive meshes is obtained by treating the adaptive grid like an unstructured mesh and using a face centered WLS diffusive and convective schemes. These schemes deal with non-orthogonal and skewness deviations that occur in the grid interface between cells of different levels of refinement.

The defined adaptive algorithm for FVM is independent of user defined parameters and can deal with the problem of the grid quality loss in the cell interface. It produces much smoother adaptive grids having high quality, making it an alternative to other algorithms existing in the literature like the one purposed by Jasak and Gosman (2000b).

Chapter 6

Conclusions and Future Work

6.1 Conclusions

A finite volume method for the solution of the 3D Navier-Stokes equations for incompressible fluid flows is presented for unstructured meshes that may include automatic grid refinement . The different FV schemes for the numerical computation of incompressible fluid flows on unstructured grids are discussed and their implementation in the SOL code is verified for selected benchmark cases. Several aspects like grid quality, numerical accuracy and efficiency are discussed.

A novel convective TVD scheme on unstructured grids is proposed in this Thesis. The main features of this new method is that it considers the location of the face centroid, correcting the accuracy issue that comes from the grid skewness. By increasing the accuracy of the r factor, the flux limiters have a more proper use and the overshoot and undershoot values are reduced when compared with other methods. Additionally, the number of iterations required to converge the solution is reduced with this new method.

An automatic adaptive grid procedure is constructed based on a relative error estimator dR. This algorithm can produce numerical solutions on adaptive grids with the same accuracy as on a fine uniform grid but with a less number of cells. Extensive simulations of the lid square cavity flow for $Re = 1000$ showed that a sequence of corner vortices could be obtained without any external or user intervention by using an automatic grid refinement algorithm and a relative error estimator.

This error estimator is capable of local refinement on vortical structures with different geometric and velocity scales, up to sixteen orders of magnitude. Also the small eddies are well captured and in agreement with the analytical results deduced by Moffatt (1963). The new error estimator detects local extremes independently of the scale

of the computational variables, being a suitable method to be applied at multi-scale problems.

Second order diffusive and convective schemes based on the WLS method are verified for several cases with adaptive grids that are known to have a certain degree of low quality near the interfaces between the cells with different levels of refinement. The FWLS schemes have several advantages over other ones from the literature since they are free from the cell centered gradient quantity and can have higher accuracy on adaptive grids. This type of FV schemes are innovative to the author's knowledge.

The proposed Residual Least Squares (RLS) is an absolute error estimator that has been shown to be suitable for adaptive refinement on unstructured grids when using FVMs. The RLS error estimator is applicable to arbitrary unstructured grids and its computation is strongly dependent on the governing equations and on the local grid quality. The error estimator combined with the proposed adaptive algorithm is independent of user defined parameters. The grid interface correction improves the overall adaptive algorithm efficiency by considering the impact of these interfaces in the grid quality. It produces much smoother adaptive grids, making it an alternative to other algorithms in the literature.

6.2 Future Work

A natural extension of the work presented in this Thesis is the development of a h-p adaptive algorithm combined with an immersed boundary method (IBM). The author has implemented, in the SOL code, an IBM based on the WLS method that can be applied on both Cartesian and unstructured grids. It has been verified that the overall method has second order accuracy for both the mean and maximum errors unlike some of other IBMs. The developed method consists of computing a parabolic polynomial with information from the computational points near the solid boundary to correct the local velocities.

In the future, to study h-p adaptive refinement, it is necessary to develop high order schemes on unstructured grids. A fourth order accurate version of the face WLS diffusive scheme has been developed and tested for different analytic cases of the Poisson equation. The method consists in using a cubic polynomial with 10 or more cell values from the proximity of the face. Also since it is a forth order method it is required to use more that one Gauss point when computing the face integrals, in order to maintain the fourth order discretization.

Due to synergies of working in the same code, it is possible to create a fourth, sixth

and eighth order with IBM correction by using the same principle as of the second order version. The idea is to use regression polynomial with the same order as of the FV discretization. These new forth, sixth and eighth order versions of the FWLS schemes should be tested in analytical cases of the Poisson, convection-diffusion and the Navier-Stokes equations.

The FV schemes and the error estimators used in this Thesis can be applied in arbitrary polyhedral grids. By adding to the SOL code, a polyhedral grid generator with the option to prescribe a hydraulic diameter h or element size distribution over the computational domain. It is possible to have a grid optimizer that obtains a polyhedral grid with the lowest number of cells for a prescribed error level. The ideal hydraulic diameter distribution is obtained by using equation 4.19 and an absolute error estimator. The demonstration of this concept for different problems with analytical solutions could be an important contribution to the literature.

The developed grid refinement algorithms and error estimators can be applied to turbulent flows, and the addition of turbulence models to the SOL code will be important in the future. Large-Eddy simulations (LES) models are gaining popularity both in the literature and in commercial codes, although questions like accuracy issues from the discretization and modeling error on unstructured and adaptive grids are no well addressed, since the majority of the LES models are developed in the structured grid framework. The SOL code has been used to compute LES simulations of the round jet problem using parallel computing for both structured and unstructured grids to study this issue.

The proposed TVD approach should also be applied on adaptive grids since it is expected to have better boundedness properties than the other TVD approaches from the literature, due to the skewness issue at the grid interfaces. Plus, the addition of a compressible flow solver to the SOL code will be important, allowing the application of adaptive refinement in problems where discontinuities and shocks occur.

Bibliography

M. Aimsworth and J.T. Oden. *A Posteriori Error Estimates in Finite Element Analysis*. Wiley-Interscience, 2000.

M. Ainsworth and J. T. Oden. A posteriori error estimation in finite element analysis. *Computer Methods Applied Mechanics and Engineering*, 141:1–88, 1997.

Djaffar Ait-Ali-Yahia, Guido Baruzzi, Wagdi G. Habashi, Michel Fortin, Julien Dompierre, and Marie Gabrielle Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. part I: general principles. *International Journal of Numerical Methods in Fluids*, 32(6):725–744, 2002.

G. D. Van Albada, B. Van Leer, and W. W. Roberts. Comparative study of computational methods in cosmic gas dynamics. *Astronomy and Astrophysics*, 108:76–84, 1982.

D. M. S. Albuquerque, J. M. C. Pereira, and J. C. F. Pereira. Calculation of a deformable membrane airfoil in hovering flight. *CMES: Computer Modeling in Engineering and Sciences*, 72(4):337–366, 2011.

I. Babuska and W.C. Rheinbolt. A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, 12:1597–1615, 1978.

T. J. Barth and P .O. Frederickson. Higher order solution of the euler equations on unstructured grids using quadratic reconstruction. *Proceedings of the 28th AIAA Aerospace Sciences Meeting, Reno, NV, Paper AIAA 90-0013,*, 1990.

S. Benhamadouche, K. Mahesh, and G. Constantinescu. *Colocated finite-volume schemes for large-eddy simulation on unstructured meshes*. Center for Turbulence Research - Proceedings of the Summer Program. Proceedings, 2002.

M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal Computational Physics*, 53:484, 1984.

M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 52:64–84, 1989.

O. Botella and R. Peyret. Benchmark spectral results on the lid-driven cavity flow. *Computers and Fluids*, 27:421–433, 1998.

J.D. Bozemann and C. Dalton. Numerical study of viscous flow in a cavity. *Journal of Computational Physics*, 12:348–363, 1973.

M. Braack and T. Richter. Solutions of 3D Navier-Stokes benchmark problems with adaptive finite elements. *Computers and Fluids*, 35:372–392, 2006.

C. Bruner and R. Walters. Parallelization of the euler equations on unstrucutred grids. *AIAA paper 97-1894*, 1995.

R. Burggraf. Analytical and numerical studies of the structures of steady separated flows. *Journal of Fluid Mechanics*, 24:113–151, 1966.

D. Calhoun. Pressure stability in fractional step finite element methods for incompressible flows. *Journal of Computational Physics*, 170(1):112–140, 2001.

D. Calhoun. A cartesian grid method for solving the two-dimensional stream function-vorticity equations in irregular region. *Journal of Computational Physics*, 176:231–275, 2002.

V. Casulli and P. Zanolli. High resolution methods for multidimensional advectione diffusion problems in free-surface hydrodynamics. *Ocean Modelling*, 10:137–151, 2005.

M. Cheng and K.C. Hung. Vortex structure of steady flow in a rectangular cavity. *Computers and Fluids*, 35:1046–1062, 2006.

W. M. Collins and S. C. R. Dennis. Viscous eddies near a 90 and a 45 corner in flow through a curved tube of triangular cross-section. *Journal of Fluid Mechanics*, 76: 417–432, 1976.

M. Darwish, I. Sraj, and F. Moukalled. A coupled finite volume solution of incompressible flows on unstructured grids. *Journal of Computational Physics*, 228(1):180 – 201, 2009.

M. S. Darwish and F. Moukalled. TVD schemes for unstructured grids. *International Journal of Heat and Mass Transfer*, 46:559–661, 2003.

Eugene de Villers. The potential of large eddy simulation for the modeling of wall bounded flows. *PhD thesis, Imperial College, University of London*, 2006.

L. Eça, M. Hoekstra, A. Hay, and D. Pelletier. A manufactured solution for a two-dimensional steady wall-bounded incompressible turbulent flow. *International Journal of Computational Fluid Dynamics*, 21:175–188, 2007.

E. Erturk, T.C. Corke, and C. Gökçöl. Numerical solutions of 2-d steady incompressible driven cavity flow at high reynolds numbers. *International Journal for Numerical Methods in Fluids*, 48:747–774, 2005.

J. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer-Verlag, 2nd edition, 1999.

L. Franca and C. Farhat. Bubble functions prompt unusual stabilized finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 123:229–308, 1995.

L. Franca and A. Nesliturk. On a two-level finite element method for incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Engineering*, 52:433–453, 2001.

U. Ghia, K. Ghia, and C. Shin. High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

S. K. Godunov. A difference scheme for numerical computation of discontinuous solution of hydrodynamics equations. *Math. Sbornik*, 47:271–306, 1959.

T. Gratsch and K. Bathe. A posteriori error estimation techniques in practical finite element analysis. *Computers and Structures*, 83:235–265, 2005.

V. Gravemeier. Scale-separating operators for variational multiscale large eddy simulation of turbulent flows. *Journal of Computational Physics*, 212:2413–2430, 2006.

K. Gustafson and K. Halasi. Vortex dynamics of cavity flows. *Journal of Computational Physics*, 64:279–319, 1986.

K. Gustafson and K. Halasi. Cavity flow dynamics at higher Reynolds number and higher aspect ratios. *Journal of Computational Physics*, 70:271–283, 1987.

K. Gustafson and R. Leben. Vortex subdomains. *First Int. Symposium on Domain Decomposition Methods for Partial Differential Equations (R. Glowinski, G. Meurant and J. Periaux Eds.)*, SIAM, Philadelphia:370–380, 1988.

E. Hachem, B. Rivaux, T. Kloczko, H. Digonnet, and T. Coupez. Stabilized finite element method for incompressible flows with high Reynolds number. *Journal of Computational Physics*, 229:8643–8665, 2010.

F. E. Ham, F. S. Lien, and A. B. Strong. High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 179:469–494, 2002.

A. Harten. High resolution schemes for hyperbolics conservation laws. *Journal of Computational Physics*, 49:357–393, 1983.

D. C. Haworth, E. L. Thary, and M. S. Huebler. A global approach to error estimation and physical diagnostics in multidimensional fluid dynamics. *International Journal for Numerical Methods in Fluids*, 17:75–97, 1993.

C. J. Heaton. On the appearance of Moffat eddies in viscous cavity flow as the aspect ratio varies. *Physics of Fluids*, 20:103102, 2008.

R. I. Issa. Solution of the implicitly discretized fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62:40 – 65, 1986.

H. Jasak. *Error analysis and estimation in the Finite Volume method with applications to fluid flows*. PhD thesis, Imperial College, 1996.

H. Jasak and A. D. Gosman. Automatic resolution control for the finite volume method, part 1: A-posteriori error estimates. *Numerical Heat Transfer, Part B*, 38(3):237–56, 2000a.

H. Jasak and A. D. Gosman. Automatic resolution control for the finite volume method, part 2: Adaptive mesh refinement and coarsening. *Numerical Heat Transfer, Part B*, 38(3):257–71, 2000b.

H. Jasak and A. D. Gosman. Automatic resolution control for the finite volume method, part 3: Turbulent flow applications. *Numerical Heat Transfer, Part B*, 38(3):273–90, 2000c.

H. Jasak and A. D. Gosman. Residual error estimate for the finite-volume method. *Numerical Heat Transfer, Part B*, 39(1):1–19, 2001.

H. Jasak and A. D. Gosman. Elemental residual error estimate for the finite volume method. *Computer and Fluids*, 32:223–248, 2003.

H. Jasak, H. G. Weller, and A. D. Gosman. High resolution NVD differencing scheme for arbitrarily unstructured meshes. *International Journal of Numerical Methods in Fluids*, 31:431–449, 1999.

F. Juretic. *Error Analysis in Finite Volume CFD*. PhD thesis, Imperial College, 2004.

F. Juretic and A. D. Gosman. Error analysis of the Finite-Volume method with respect to mesh type. *Numerical Heat Transfer, Part B: Fundamentals*, 57(6):441–439, 2008.

M. Kobayashi, J. M. C. Pereira, and J. C. F. Pereira. A conservative Finite-Volume second-order accurate projection method on hybrid unstructured grids. *Journal of Computational Physics*, 150:40–75, 1999.

M.H. Kobayashi, J.M.C. Pereira, and J.C.F. Pereira. A 2nd order upwind least squares scheme for incompressible flows on unstructured hybrid grids. *Numerical Heat Transfer, Part B*, 34:39–60, 1998.

J. Kong, P. Xin, C.-J. Shen, Z.-Y., and L. Li. A high-resolution method for the depth-integrated solute transport equation based on an unstructured mesh. *Environmental Modelling and Software*, 49:357–393, 2013.

D. Lacasse, E. Turgeon, and D. Pelletier. On the judicious use of the k-e model, wall functions and adaptivity. *International Journal of Thermal Sciences*, 43(10): 925–938, 2004.

B. Van Leer. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32:101–136, 1979.

B. P. Leonard. The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection. *Comp. Methods Appl. Eng.*, 88:17–74, 1991.

L. X. Li, H. S. Lao, and L. J. Qi. An improved r-factor algorithm for TVD schemes. *International Journal of Heat and Mass Transfer*, 51:59–98, 2008.

F. S. Lien. Upstream monotonic interpolation for scalar transport with application to complex turbulent flows. *International Journal Numerical Methods in Fluids*, 19: 527–548, 1993.

J. P. P. Magalhães. An adaptive framework for the numerical simulation of environmental flows in complex geometries. *PhD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa*, 2011.

K. Mahesh, G. Constantinescu, and P. Moin. A numerical method for large-eddy simulation in complex geometries. *Journal of Computational Physics*, 197(1):215–240, 2004.

D. Martin and P. Colella. A cell-centered adaptive projection method for the incompressible Euler equations. *Journal of Computational Physics*, 163:271–312, 2000.

D. Martin, P. Colella, and D. Graves. A cell-centered adaptive projection method for the incompressible Navier-Stokes equations in three dimensions. *Journal of Computational Physics*, 227:1863–1886, 2008.

B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of graphics tools*, 1(2), 1996.

H.K. Moffatt. Viscous and resistive eddies near a sharp corner. *Journal of Fluid Mechanics*, 18(1):1–18, 1963.

S. Muzaferija and D. Gosman. Finite-volume CFD procedure and adaptive error control strategy for grids of arbitrary topology. *Journal of Computational Physics*, 138:766–787, 1997.

M.-J. Ni, R. Munipalli, P. Huang, N. B. Morley, and Mohamed A. Abdou. A current density conservative scheme for incompressible mhd flows at a low magnetic reynolds number. part ii: On an arbitrary collocated mesh. *Journal of Computational Physics*, 227:205–228, 2007.

Serge Nicaise. A posteriori error estimations of some cell-centered finite volume methods. *SIAM Journal on Numerical Analysis*, 43(4):1481–1503, 2006.

S. Osher. High resolution applications of the OSHER upwind scheme for the Euler equations. *AIAA paper*, 83:1943, 1983.

J. S. Park, S. H. Yoon, and C. Kim. Multi-dimensional limiting process for hyperbolic conservation laws on unstructured grids. *Journal of Computational Physics*, 229: 788–812, 2010.

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

D. Pelletier and R. Camareo. Finite element simulation of 3D turbulent free shear flows. *International Journal for Numerical Methods in Fluids*, 8(12):1563–1586, 2005.

D. Pelletier, E. Turgeon, and D. Tremblay. Verification and validation of impinging round jet simulations using an adaptive FEM. *International Journal for Numerical Methods in Fluids*, 44:737, 2004.

J. Peraire, M. Vahdati, K. Morgan, and O. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72:449–466, 1987.

J.M.C. Pereira and J.C.F. Pereira. Fourier analysis of several finite difference schemes for the one-dimensional unsteady convection-diffusion equation. *International Journal for Numerical Methods in Fluids*, 36:417–439, 2001.

H. Pina. *Métodos Numéricos*. Mc Graw-Hill, 2001.

T. Plewa, T. Linde, and V. G. Weirs. *Adaptive mesh refinement - theory and applications*. Springer-Verlag Berlin Heidelberg, 2005.

R. Reis. Cálculo paralelo das equações de Euler utilizando métodos de Newton-Krylov. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, 2005.

C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21:1525–1532, 1983.

P. L. Roe. Some contributions to the modelling of discontinuous flows. *Lectures in Applied Mechanics, Springer-Verlag, Berlin*, 22:163–193, 1985.

M. Schafer and S. Turek. Benchmark computations of laminar flow around a cylinder. *Flow simulation with high performance computers II. DFG priority research program results*, 52:547–66, 1996.

K. Segeth. A review of some a posteriori error estimates for adaptive finite element methods. *Mathematics and Computers in Simulation*, 80:1589–1600, 2008.

P. Shankar and M. Deshpande. Fluid mechanics in the driven cavity. *Annual Review of Fluid Mechanics*, 32:93–136, 2000.

P.N. Shankar. Moffat eddies in the cone. *Journal of Fluid Mechanics*, 539:113–135, 2005.

A. V. Shapeev and P. Lin. An asymptotic fitting Finite Element method with exponential mesh refinement for accurate computation of corner eddies in viscous flows. *SIAM Journal on Scientific Computing*, 31(3):1874–1900, 2009.

T. M. Shih, C. H. Tan, and B. C. Hwang. Effects of grid staggering on numerical schemes. *International Journal for Numerical Methods in Fluids*, 9:193, 1989.

J. L. Steger and R. F. Warming. Flux vector splitting of the inviscid gas dynamic equations with application to finite-difference methods. *Journal of Computational Physics*, 40:263–293, 1981.

P. K. Sweby. High resolution schemes using flux-limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21:995–1011, 1984.

M. C. Thompson and J. H. Ferziger. An adaptive multigrid technique for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 82:94–121, 1989.

Z. Tukovik and H. Jasak. A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow. *Computers and Fluids*, 55:70–84, 2012.

V. Venkatakrishnan. On the convergence of limiters and convergence to steady state solutions. *31st AIAA Aerospace Sciences Meeting, AIAA-93-0880*, 1994.

V. Venkatakrishnan. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters,. *Journal of Computational Physics*, 118:120–130, 1995.

R.W. Verstappen and A.E. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187:343–368, 2003.

H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics*. Pearson Education Limited, 2007.

J. Volker. A numerical study of a posteriori error estimators for convection-diffusion equations. *Computer Methods Applied Mechanics and Engineering*, 190:751–781, 2000.

R. F. Warming and R. M. Beam. Upwind second order difference schemes and applications in aerodynamics. *AIAA Journal*, 14(9):1241–1249, 1976.

N. P. Waterson and H. Deconinck. Design principles for bounded higher-order convection schemes - a unified approach. *Journal of Computational Physics*, 224:182–207, 2007.