GRAZ UNIVERSITY OF TECHNOLOGY

DOCTORAL SCHOOL MECHANICAL ENGINEERING

P H D T H E S I S

to obtain the degree

Dr. techn.

of Graz University of Technology

Specialty : Computational Fluid Dynamics

Defended by

Michael Buchmayr

Development of Fully Implicit Block Coupled Solvers For Incompressible Turbulent Flows

Thesis Supervisor: Prof. Wolfgang Sanz

Thesis Co-Supervisor: Dr. Luca Mangani

prepared at ANDRITZ AG Graz

January, 2014

Reviewers :	Prof. Wolfgang Sanz	-	Graz University of Technology
	Prof. Marwan Darwish	-	American University Beirut
Supervisor :	Prof. Wolfgang Sanz	-	Graz University of Technology
Co-Supervisor :	Dr. Luca Mangani	-	Hochschule Luzern
Examiners :	Prof. Wolfgang Sanz	-	Graz University of Technology
	Prof. Marwan Darwish	-	American University Beirut

Dedication

In love, admiration and gratitude to my parents, Gabi and Bruno.

Statement

The code that is presented in this PhD thesis has been elaborated in a joint effort between Luca Mangani, Marwan Darwish and the author himself. The author holds to stress that the outcome of this project could not have been achieved without the input of each of them, and that in his view all of them have equal merit to the success of the project. The author has made substantial contributions to almost all of the features outlined in this work. However, in order to present a complete view of the solver's internals, the author opted to outline also parts in which he has not been or has only marginally been involved. The author did this with the consent of his partners. In such a case, a note is made at the beginning of the respective chapter to indicate the merit of the main responsible.

Eidesstattliche Erklärung

Ich erläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....

Abstract

This work presents the development of a set of fully implicit block coupled solvers for incompressible turbulent fluid flows (steady-state, rotational frame, transient solver). The method is based on the finite volume approach for arbitrarily unstructured collocated grids. The Navier-Stokes equations are solved simultaneously employing Rhie and Chow's interpolation technique to overcome the saddle point problem of the resulting discretised block coupled system of equations. The method is based on the work of Darwish et al. and extends their approach to three-dimensional turbulent flows. The elaborated code is written in object oriented C++ making use of the open source CFD library OpenFOAM. The proposed method is benchmarked against a state-of-the-art segregated SIMPLEC solver by Mangani and it proves to outperform the segregated solver in terms of convergence speed and robustness. Most importantly it has been shown that the outlined approach exhibits excellent mesh size scalability, a property that segregated solvers, such as the benchmark solver, fail to show for intrinsic reasons. Three different solvers are presented, namely a steady-state solver for inertial reference frames, a steady-state solver for rotational reference frames, and a transient solver for moving control volumes in arbitrary Lagrange Euler (ALE) formulation. A detailed derivation of the governing equations for all of these reference frames, as well as a detailed description of their block implicit discretisation is given. Finally turbomachinery related test cases including a transient simulation of a centrifugal pump, that has been compared to experimental particle image velocimetry (PIV) measurements, are outlined to show the code's well functioning, and in order to present a validation in respect to flow physics capturing.

Acknowledgements

I would like to express my sincere gratitude to my doctoral father Prof. Wolfgang Sanz for his help and patience throughout the elaboration of this thesis. Especially in tougher times during the past years his support and advice gave me strength to overcome these moments, which I am genuinely thankful for. He has also always had an open ear for issues not related to my PhD project and took time to help me out whenever I needed it.

I am also grateful to my family and my girlfriend Magali for their patience during these years. Their support made it easy for me to pursue this challenge. Being able to share my achievements with them and to see them feeling happy for me too, gives me the greatest joy.

I am infinitly grateful for having met Prof. Marwan Darwish and Dr. Luca Mangani. They have literally saved my ass. I believe that the goal that I had set for this project, namely the elaboration of a set of block implicit coupled solvers, would have been far too difficult for me to reach without working as a team with these two experts. It has been a collaboration full of fun, confidence and mutual respect. For me working with them was enriching on a professional as well as on a personal level, and I am very thankful that today I can call them my friends. I hope to be able to continue this journey that we have started some years ago.

A special thank you goes to all the programmers that provide and maintain the OpenFOAM CFD library, and in particular to Prof. Hrvoje Jasak and Dr. Ivor Clifford for providing their block coupling features.

I also want to thank Prof. Celigoj, Prof. Brenn, Prof. Steiner, Prof. Steinbach and Prof. Ehrlacher for being role models, for getting me into continuum mechanics, and for providing me with sound foundations for this particular project and hopefully for many more to come.

Without the support of my employer I could not have undertaken this endeavour. Therefore, I want to thank the responsible persons at Andritz AG's pump department for initiating and financing this project. A special thank you goes to Dr. Arno Gehrer who has been supervising this project and who has continuously been pushing me on the right track. Along with Arno I want to thank my 'mentors' at Andritz, namely DI Josef Werderits, Dr. Etienne Parkinson, Arnold Egger and Dr. Jean-Christophe Marongiu, who gave me the possibility to work and learn in an excellent environment, and who have introduced me to the exciting field of hydraulic

machines.

I want to thank Prof. Woisetschläger and DI Stefan Hödl for providing valuable PIV measurements that I used for evaluating the code.

I also want to thank the colleagues from my office for enduring my moods during the past years.

The financial aid granted by the Austrian research fund FFG, for project 828688 - HydroSim, is greatly acknowledged.

Nomenclature

$\mathbf{A}$,a	coefficient matrix, coefficient matrix coefficient
$\mathbf{b}$,b	source vector, source vector coefficient
$\mathbf{u}$	velocity vector
u,v,w	velocity components
p	pressure
k	turbulence kinetic energy
ω	turbulence frequency
ρ	density constant
$\mathbf{D}$	Rhie-Chow numerical dissipation tensor
V,$\dot{V}$	volume scalar, volume flux scalar
S,$\mathbf{S}$	surface scalar, surface normal vector
g	geometric interpolation weighting factor
ν	kinematic viscosity scalar
ϕ	general scalar quantity, solution vector

Superscripts

u,v,w	refers to velocity components

n	current iteration
$\overline{\phi}$	linear interpolation to the face
Γ	diffusion tensor
μ	dynamic viscosity
ν	kinematic viscosity
$\mathbf{n}$	surface normal vector
$\mathbf{x}$	instantaneous position of material point
$\mathbf{X}$	reference position of material point at time t=0
phi	volume flux

Contents

List of Tables

List of Figures

Chapter 1

Introduction

At the beginning of the project that has lead to the elaboration of this thesis there was the idea of improving the quality of prediction of instabilities occurring at part load operation conditions in centrifugal pumps. This could have been done proceeding an analytical approach, however since it was commonly believed that such instabilities are triggered by unsteady flow phenomena it has been decided to opt for a numerical solution to the problem.

Furthermore the author and the decision makers in his company decided to rely on the open source CFD library OpenFOAM, which back then attracted growing interest in both industrial and scientific communities, for these numerical studies, with the initial perception that strongly needed features for simulations of required type were met by this CFD package. Furthermore, the ability to add additional or missing features to the code led to the final decision to base envisaged research on this platform.

The vision was to establish a tool chain that would allow hydraulic designers henceforth to use the CFD codes resulting from this project in everyday design. The assumption that the new tool would be accepted by designers if, and only if, it was better or at least as good as their existing commercial tool resulted in a set of goals for this project that served as a rule for development throughout the project. It has also been assumed that all of these goals were to be met for a successful conclusion of the project. These goals were:

- Calculation Speed
- Robustness
- Accuracy
- Flexibility
- Maintainability
- No License Costs

The very beginning of the OpenFOAM framework goes back to a handful researchers at Imperial College, gathering around Henry Weller in the late 90's, who started to write a new object-oriented framework for mostly finite volume based continuum mechanics. The driving force was their frustration about existing CFD codes written in a rather sequential manner. The objective was to create a modular framework, that would permit ease of extension and maintenance. The pursuance of this strategy, which still prevails in the code today, lead to a fast expansion of the code and an exponential increase in the number of users as it went viral, after having been made available under GPL licence.

Apart from the concepts of polymorphism and templating, the concept of operator overloading that allows the assembly, discretisation and solution of equations by simply defining equations at the top level of the solver, was particularly attractive. Its elegant implementation was only possible, because the coefficients for segregated approaches could be gathered into a unique homogeneous coding structure. This however put a restriction to the framework, namely that it was, until very recently, not possible to use its potential for block coupling of equations.

After a first testing phase of the framework it became apparent that four of the six goals mentioned above could be met. It was physically accurate, flexible, easy to maintain due to its object-oriented structure and the big community providing bug-fixes. Due to the GPL licence the problem of license cost could also be avoided. However concerning calculation time and robustness it was found that the framework could by far not meet the desired standards. It

was also found that both shortcomings were related to the segregated solution approach that was intrinsic to the framework.

The author decided out of necessity that for a successful conclusion of the project it was imperative to leave the paradigm of segregated solvers and turn towards the implementation of fully block coupled implicit pressure based solvers within the OpenFOAM framework. This unexpected deficiency of the OpenFOAM framework hence led to an adjustment of the original plan of solely investigating the physical origin of pump instabilities to almost pure code development. The developed fully implicit block coupled solvers will hence be the central part of this thesis, the evaluation of instabilities in pump operation will also be shown in a case study.

The author is convinced that mentioned development of a fully implicit block coupled solver would not have been possible without collaborating with a team of experts. For this work credit goes to Luca Mangani and Marwan Darwish, who have equal merit in the successful elaboration of this project. Credit also goes to Hrvoje Jasak and Ivor Clifford for providing a block matrix framework, that was of great value for this project.

The author's sincere gratitude goes to all them.

The structure of this thesis is supposed to lead the lecturer from the very derivation of governing equations to the results of an actual simulation. When developing a CFD code, one has to go through different phases or let us say work packages. This thesis has been structured in a way that reflects the author's chronology of developing the resulting CFD solvers. It starts with a description of underlaying physics and kinematics (chapter 2). After that, transport theorems for three different reference frames are derived, and the governing equations for all reference frames are derived thereof (chapter 3). The resulting governing equations have to be approached employing a numerical algorithm. The selected block coupled pressure based incompressible algorithm is therefore presented and compared to commonly used CFD algorithms (chapter 4). In what follows the discretisation of the governing equations is presented for the employed block coupled algorithm in all three derived reference frames (chapters 5,6,7). For an efficient solution of the resulting linear system of equations a special linear equation solver is necessary (block AMG solver, chapter 8). To refine physical accuracy of the derived algorithm a turbulence model was added, which is presented subsequently (chapter 9). Finally the outcome of all steps mentioned above is a very robust, fast and accurate set of solvers, results of which are presented at the end to evaluate the code and to round up this thesis (chapter 10).

Chapter 2

Continuum Mechanics

Contents

In what follows a kinematic description of continuous bodies moving in different frames of reference is given and the constitutive laws of Newtonian fluids are presented. The notions presented in this chapter serve as foundation for the derivation of the governing equations in different reference frames formulated subsequently in chaper 3.

2.1 Geometry and Kinematics of Bodies

The aim of this section is introduce certain notations from basic continuum mechanics theory. The basic ideas of the motion of continuous bodies will be outlined referring to Marsden and Hughes[5], and Stoker [6]. The notations that will be pointed out in this section will be used in chapter 3 where transport theorems will be derived.

2.1.1 Continuous Bodies and their Configurations

A simple continuous body or volume V_X normally occupies an open subset of three dimensional space R^3. Subsequently it will not be distinguished between Eucledian space and R^3. A configuration or displacement of V_X in time is a mapping. Those mappings will be useful in chapter 3 where we will need them to perform coordinate transformations through Jacobian determinants.

Motions and displacements can be specified in various descriptions:

· The **Lagrangian Description**, where the intensive variables are a function of the **material coordinate X** at time t_0 (reference time). The material coordinate **X** being constant in respect to time.

· The **Eulerian Description**, where the intensive variables are a function of the **current coordinate x** at time t. This is the coordinate of a material particle at time t, that was at **X** at reference time. Only one particle can be at a position at a time, and its motion is unique.

· The **Referential Description**, where the intensive variables are a function of the **spatial coordinate** χ at time t. Coordinates χ and **X** are related through a function of motion, which can be, but must not be, the material particle motion. In special cases coordinate χ will be the one where evaluations will be made (e.g. moving mesh calculations).

As stated before the different types of coordinate systems can be related through mapping functions. The mapping functions can be known, or have to be estimated. Mapping functions on bodies or volumes are also called configurations.

The material configuration ψ is given by:

$$\mathbf{x} = \psi(\mathbf{X}, t) \tag{2.1}$$

The material configuration follows the paths of the material points with time.

The spatial or reference configuration Ψ is given by:

$$\chi = \Psi(\mathbf{X}, t) \tag{2.2}$$

The reference configuration defines paths through R^3 in time. These paths usually do not coincide with the material points' paths.

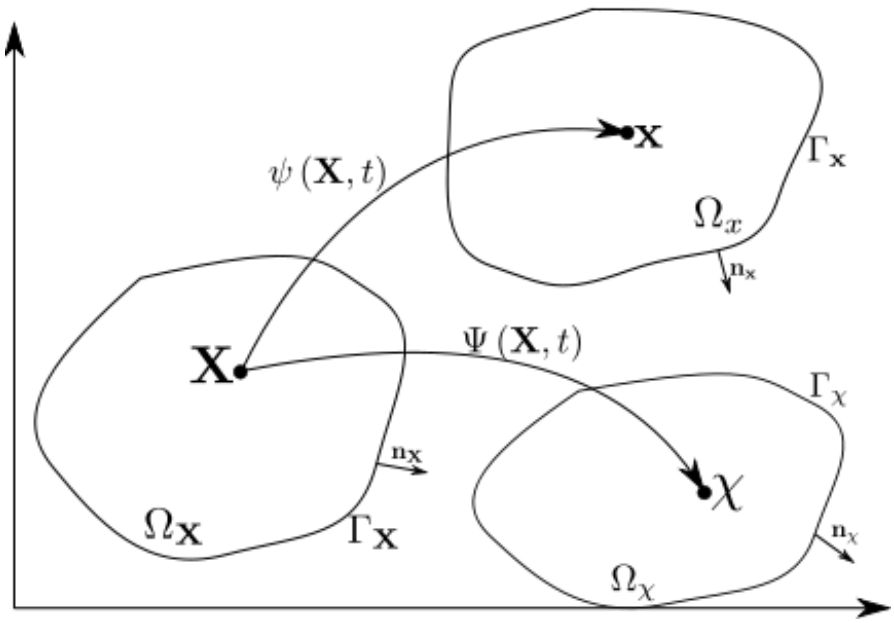

Figure 2.1: Material control volume and static reference volume

2.1.2 Material Configurations

We recall that the material configurations of a material point $\mathbf{X}$ are given by.

$$\mathbf{x} = \psi(\mathbf{X}, t) \tag{2.3}$$

Typically one does not know a priori which path a material point will follow through a domain with time. Therefore it is necessary to estimate its configuration. Commonly a linearisation approach is used.

By linearisation:

$$\begin{aligned} x_1 &= X_1 + u_1 \cdot \Delta t \\ x_2 &= X_2 + u_2 \cdot \Delta t \\ x_3 &= X_3 + u_3 \cdot \Delta t \end{aligned} \tag{2.4}$$

2.1.3 Reference Configurations

We recall that the reference configurations of a material point $\mathbf{X}$ are given by.

$$\chi = \Psi(\mathbf{X}, t) \tag{2.5}$$

Sometimes it is interesting or even necessary to know how fluid properties evolve with time over a selected path. In CFD this is e.g. the case when the meshes in a calculation domain move or deform. Then the fluid properties can not be evaluated at a static point, such as it is the case with non-moving and non-deforming numerical grids.

If it is difficult to define a smooth function for the reference configuration beforehand, the simplest approach is again linearisation.

$$\begin{aligned} \chi_1 &= X_1 + u_{g1} \cdot \Delta t \\ \chi_2 &= X_2 + u_{g2} \cdot \Delta t \\ \chi_3 &= X_3 + u_{g3} \cdot \Delta t \end{aligned} \tag{2.6}$$

At this point a very special configuration shall be examined, which will be needed to derive a transport theorem for **rotating control volumes**.

$$\chi = \mathbf{R}(\mathbf{X} - \mathbf{r}) + \mathbf{r} \tag{2.7}$$

$\mathbf{R}$ is a 3-dimensional rotation matrix and $\mathbf{r}$ the distance from the origin to the centre of rotation. For generality the rotation matrix is chosen as a rotation about an arbitrary axis in R^3. Hence the configuration reproduces exactly the rotation of the angle $\Theta = \omega \Delta t$ at a point about a unit rotation vector $\mathbf{n}$. The function of motion is known. The rotation matrix reads:

$$\mathbf{R} = \tag{2.8}$$

$$\begin{pmatrix} c(\Theta) + {n_x}^2(1 - c(\Theta)) & n_x n_y(1 - c(\Theta)) - n_z s(\Theta) & n_x n_z(1 - c(\Theta)) + n_y s(\Theta) \\ n_x n_y(1 - c(\Theta) + n_z s(\Theta)) & c(\Theta) + {n_y}^2(1 - c(\Theta)) & n_y n_z(1 - c(\Theta)) - n_x s(\Theta) \\ n_x n_z(1 - c(\Theta)) - n_y s(\Theta)) & n_y n_z(1 - c(\Theta)) - n_x s(\Theta) & c(\Theta) + {n_z}^2(1 - c(\Theta)) \end{pmatrix}$$

2.2 Constitutive Laws

Newton-Stokes Hypothesis

In order to relate the movement of the fluid to the resulting stresses acting on the fluid, a constitutive law is needed. Therefore Newton's assumption, which linearly relies the strain rate tensor and stresses, combined with Stokes' hypothesis yields a framework for isotropic fluids such as water or ideal gas.

The total stress tensor σ_{ij} can be written as follows [7]:

$$\sigma_{ij} = -p\delta_{ij} - \frac{2}{3}\mu S_{kk}\delta_{ij} + 2\mu S_{ij} \tag{2.9}$$

The first term on the RHS is the pressure that acts normal to the confined fluid volume, the second term represents the dilatation of the fluid volume and the third term represents the shear stresses that arise from the linearised strain rate of the fluid volume.

Hereby the strain rate tensor S_{ij} is given as:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.10}$$

and following Einstein's summation:

$$S_{kk} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \tag{2.11}$$

Fourier Heat Conduction

In the scope of this thesis we shall consider isotropic heat conduction and will choose Fourier's approach that states that the heat flux is proportional to the temperature gradient and a scalar coefficient λ.

$$q_j = -\lambda \frac{\partial T}{\partial x_j} \tag{2.12}$$

Specific Heat Coefficients

In order to be able to link the internal energy e and the enthalpy h, we shall use constant specific heat coefficients at constant pressure c_p or at constant volume c_v under the assumption of ideal gas, or an incompressible fluid (i.e. water) respectively.

$$e = c_v T \tag{2.13}$$

$$h = c_p T \tag{2.14}$$

Chapter 3

Derivation Of Governing Equations For Incompressible Fluid Flows

Contents

In the previous chapter (2) the foundations of continuum mechanics were presented. These foundation included the kinematics of bodies in different frames of reference as well as the constitutive laws for Newtonian fluids. In this chapter these notions are taken up and transport theorems are derived for all reference frames. Finally, using these transport theorems, the governing equations for three reference frames are derived. In the next chapter (4) a block-coupled pressure-based incompressible numerical algorithm for the derived governing equations is presented.

The governing equations of fluid mechanics are based on three principles:

- Mass is conserved
- Newton's second law (momentum is conserved)
- Energy is conserved

For turbomachinery applications we generally deal with problems that involve rotating machines. Flows in such machines can be treated using different numerical approaches. Therefore, the governing equations will be outlined in what follows, for inertial frames of reference, moving control volumes and rotating reference frames. In a first step the respective transport theorems for each type of configuration will be derived, and thereof the governing equations will be established.

3.1 Non-Moving Reference Volumes In Inertial Frame Of Reference

3.1.1 The Reynolds Transport Theorem for Non-Moving Reference Volumes in Inertial Frame of Reference

The Reynolds Transport Theorem (RTT) for non-moving reference volumes is derived at this point in order to subsequently derive the governing equations of fluid mechanics in an inertial frame for non-moving numerical grids. The Reynolds Transport Theorem can be considered as the time rate of change of any given quantity Φ, which is the integral of an intensive property ϕ over a material control volume. A material control volume contains material points and deforms as those material points move, so that the material points always stay inside the volume.

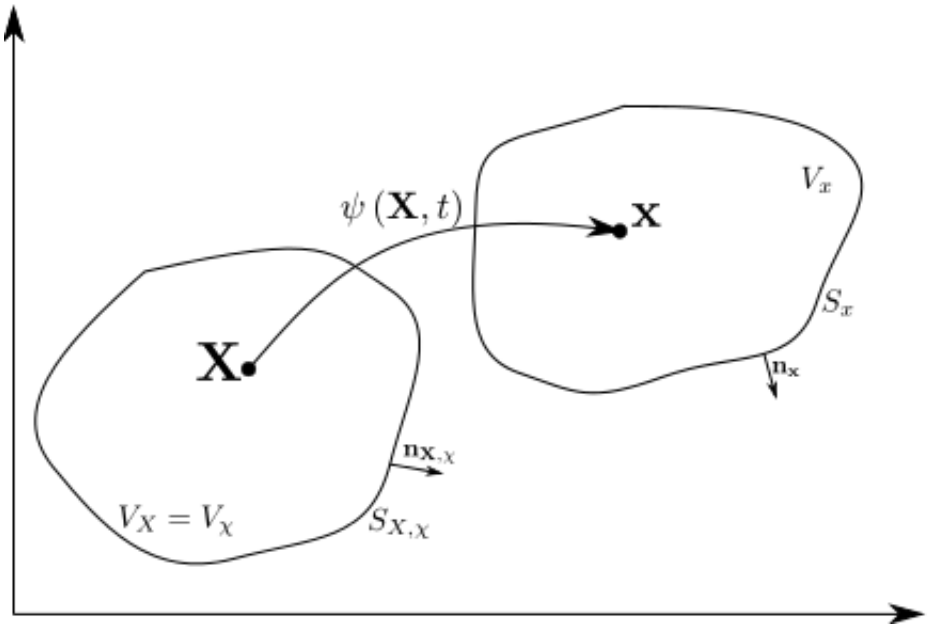

Figure 3.1: Material control volume and static reference volume

Let us consider a quantity of an extensive fluid property Φ. This quantity is the integral of a volume specific quantity (the intensive fluid property ϕ) over a given material volume. Or

more general:

$$\Phi(t) = \int\limits_{V_x(t)} \phi(\mathbf{x}, t)\, \mathrm{dV_x} \tag{3.1}$$

Hence, the time rate of change of the fluid property is:

$$\frac{d\Phi(t)}{dt} = \frac{d}{dt} \int\limits_{V_x(t)} \phi(\mathbf{x}, t)\, \mathrm{dV_x} \tag{3.2}$$

The limits of integration vary with time. Therefore the time derivative cannot be taken directly inside the integral. We can overcome this problem by introducing a mapping function, that relates the material volume at time t with a reference material volume at time t_0. The mapping function ψ of a domain is a function of the reference position of the material points and time.

$$\mathbf{x} = \psi(\mathbf{X}, t) \tag{3.3}$$

The deformation gradient matrix $\mathbf{F}$ is defined as:

$$\mathbf{F} = \frac{\partial x_i}{\partial X_j} \mathbf{e}_i \otimes \mathbf{e}_j \tag{3.4}$$

$$\mathbf{F} = \begin{pmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_3}{\partial X_2} \\ \frac{\partial x_1}{\partial X_3} & \frac{\partial x_2}{\partial X_3} & \frac{\partial x_3}{\partial X_3} \end{pmatrix} \tag{3.5}$$

The Jacobian determinant $\mathbf{J}_m$, is defined as:

$$\mathbf{J}_m = det(\mathbf{F}) \tag{3.6}$$

$$\mathbf{J}_m = \begin{vmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_3}{\partial X_2} \\ \frac{\partial x_1}{\partial X_3} & \frac{\partial x_2}{\partial X_3} & \frac{\partial x_3}{\partial X_3} \end{vmatrix} \text{ given } \mathbf{x} = \psi(\mathbf{X}, t) \tag{3.7}$$

The material volume at time t relates to the reference material volume through the Jacobian determinant:

$$\mathrm{dV_x} = \mathbf{J}_m \mathrm{dV_X} \tag{3.8}$$

The intensive quantity ϕ can be expressed in terms of the material coordinates by:

$$\phi(\mathbf{x}, t) = \phi\left[\psi(\mathbf{X}, t), t\right] = \phi(\mathbf{X}, t) \tag{3.9}$$

Using the above derivations equation (3.2) can be expressed as follows:

$$\frac{d\Phi(t)}{dt} = \frac{d}{dt} \int_{V_X} \{\phi(\mathbf{X}, t)\mathbf{J}_m\} \, \mathrm{dV_X} \tag{3.10}$$

Now V_X is independent of time, and the material derivative can be taken inside the integral. Using the chain rule we obtain:

$$\frac{d\Phi(t)}{dt} = \int_{V_X} \left\{ \frac{d\phi(\mathbf{X}, t)}{dt} \mathbf{J}_m + \phi(\mathbf{X}, t) \frac{d\mathbf{J}_m}{dt} \right\} \mathrm{dV_X} \tag{3.11}$$

Now equation (3.11) shall be simplified. The material derivative of ϕ can be decomposed into a local time derivative and a convective derivative using the chain rule.

$$\frac{d\phi(\mathbf{X}, t)}{dt} = \frac{d\phi(\mathbf{x}, t)}{dt} = \frac{\partial \phi(\mathbf{x}, t)}{\partial t} + (\mathbf{u} \cdot \nabla_x)\phi \tag{3.12}$$

To simplify the given set of equations the following equality shall be used.

$$\frac{d\mathbf{J}_m}{dt} = \mathbf{J}_m \left(\nabla_X \cdot \mathbf{u}\right) \tag{3.13}$$

By deriving the equation (3.13) it shall be shown, which assumptions have to be made to get there and which terms have been neglected. Through a linearised approach $\mathbf{x} = \psi(\mathbf{X}, t)$ can be expressed as follows:

$$\begin{aligned} x_1 &= X_1 + u_1 \cdot dt \\ x_2 &= X_2 + u_2 \cdot dt \\ x_3 &= X_3 + u_3 \cdot dt \end{aligned} \tag{3.14}$$

One always has to keep in mind that the so-derived transport theorem is an approximation, due to the fact that the deformation is assumed to be linear in respect to the velocity of any material point at reference time. But this is only the first assumption that we have to make to derive the transport theorem. The second assumption will be outlined later (equation (3.21)).

Let us now fill up the deformation tensor $\mathbf{F}$ by taken the derivative of the material particles' motion over the reference coordinates.

$$\frac{\partial x_i}{\partial X_j} = \begin{cases} 1 + \frac{\partial u_i}{\partial X_j} dt & \text{if i=j} \\ \frac{\partial u_i}{\partial X_j} dt & \text{if i} \neq \text{j} \end{cases} \tag{3.15}$$

Hence the deformation tensor reads,

$$\mathbf{F} = \begin{pmatrix} 1 + \frac{\partial u_1}{\partial X_1} dt & \frac{\partial u_2}{\partial X_1} dt & \frac{\partial u_3}{\partial X_1} dt \\ \frac{\partial u_1}{\partial X_2} dt & 1 + \frac{\partial u_2}{\partial X_2} dt & \frac{\partial u_3}{\partial X_2} dt \\ \frac{\partial u_1}{\partial X_3} dt & \frac{\partial u_2}{\partial X_3} dt & 1 + \frac{\partial u_3}{\partial X_3} dt \end{pmatrix} \tag{3.16}$$

And the Jacobian becomes,

$$\mathbf{J}_m = det(\mathbf{F}) = \left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_2}{\partial X_2} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right) + O(dt^2) \tag{3.17}$$

$$\begin{aligned} \frac{d\mathbf{J}_m}{dt} &= \frac{\partial u_1}{\partial X_1}\left(1 + \frac{\partial u_2}{\partial X_2} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right) + \frac{\partial u_2}{\partial X_2}\left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right) \\ &\quad + \frac{\partial u_3}{\partial X_3}\left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_2}{\partial X_2} dt\right) + O(dt) \end{aligned} \tag{3.18}$$

$$\tag{3.19}$$

At this stage we know the time derivative of the Jacobian. Now it stays to show how the ominous equality of equation (3.13) is obtained.

$$\begin{aligned} \mathbf{J}_m\left(\nabla_X \cdot \mathbf{u}\right) &= \left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_2}{\partial X_2} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right)\left(\frac{\partial u_1}{\partial X_1} + \frac{\partial u_2}{\partial X_2} + \frac{\partial u_3}{\partial X_3}\right) \\ \mathbf{J}_m\left(\nabla_X \cdot \mathbf{u}\right) &= \frac{\partial u_1}{\partial X_1}\left(1 + \frac{\partial u_2}{\partial X_2} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right) + \frac{\partial u_2}{\partial X_2}\left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right) \\ &\quad + \frac{\partial u_3}{\partial X_3}\left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_2}{\partial X_2} dt\right) \\ &\quad + \frac{\partial u_1}{\partial X_1}\left(\frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_2}{\partial X_2} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right) \\ &\quad + \frac{\partial u_2}{\partial X_2}\left(\frac{\partial u_2}{\partial X_2} dt\right)\left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_3}{\partial X_3} dt\right) \\ &\quad + \frac{\partial u_3}{\partial X_3}\left(\frac{\partial u_3}{\partial X_3} dt\right)\left(1 + \frac{\partial u_1}{\partial X_1} dt\right)\left(1 + \frac{\partial u_2}{\partial X_2} dt\right) \end{aligned} \tag{3.20}$$

Hence,

$$\begin{aligned}\mathbf{J}_m\left(\nabla_X\cdot\mathbf{u}\right) &= \frac{d\mathbf{J}_m}{dt} - O(dt) \\ &+\frac{\partial u_1}{\partial X_1}\left(\frac{\partial u_1}{\partial X_1}dt\right)\left(1+\frac{\partial u_2}{\partial X_2}dt\right)\left(1+\frac{\partial u_3}{\partial X_3}dt\right) \\ &+\frac{\partial u_2}{\partial X_2}\left(\frac{\partial u_2}{\partial X_2}dt\right)\left(1+\frac{\partial u_1}{\partial X_1}dt\right)\left(1+\frac{\partial u_3}{\partial X_3}dt\right) \\ &+\frac{\partial u_3}{\partial X_3}\left(\frac{\partial u_3}{\partial X_3}dt\right)\left(1+\frac{\partial u_1}{\partial X_1}dt\right)\left(1+\frac{\partial u_2}{\partial X_2}dt\right)\end{aligned} \tag{3.21}$$

(3.22)

$$\frac{d\mathbf{J}_m}{dt} = \mathbf{J}_m\left(\nabla_X\cdot\mathbf{u}\right) + O(dt) \tag{3.23}$$

Thus, equation (3.13) is first order accurate in time, when a linearisation is chosen to describe the particle motion $\psi(\mathbf{X},t)$ (see (3.14).

$$\frac{d\mathbf{J}_m}{dt} \approx \mathbf{J}_m\left(\nabla_X\cdot\mathbf{u}\right) \tag{3.24}$$

With the help of this relation equation (3.11) can be rearranged.

$$\frac{d\Phi(t)}{dt} = \int\limits_{V_X}\left\{\frac{\partial\phi(\mathbf{x},t)}{\partial t} + (\mathbf{u}\cdot\nabla_X)\phi(\mathbf{x},t) + \phi(\mathbf{x},t)(\nabla_X\cdot\mathbf{u})\right\}\mathbf{J}_m\,\mathrm{dV_X} \tag{3.25}$$

$$\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)}\left\{\frac{\partial\phi(\mathbf{x},t)}{\partial t} + (\mathbf{u}\cdot\nabla_X)\phi(\mathbf{x},t) + \phi(\mathbf{x},t)(\nabla_X\cdot\mathbf{u})\right\}\mathrm{dV_x} \tag{3.26}$$

$$\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)}\left\{\frac{\partial\phi(\mathbf{x},t)}{\partial t} + \nabla_X\cdot(\mathbf{u}\phi(\mathbf{x},t))\right\}\mathrm{dV_x} \tag{3.27}$$

Employing Gauss' Theorem equation (3.27) reads,

$$\boxed{\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \frac{\partial \phi(\mathbf{x},t)}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\cdot(\mathbf{u}\phi(\mathbf{x},t))\,\mathrm{dS_x}} \tag{3.28}$$

Equation (3.28) is commonly referred to as Reynolds Transport Theorem (RTT).

Using the RTT it is possible to evaluate the total time derivative of an extensive fluid quantity inside a moving control volume through integration over a volume $V_x(t)$. Hence the Lagrangian description of a quantity in a moving volume is replaced by a Eulerian description of a quantity that locally changes with time.

Recall that the RTT is obtained by the linearisation of the material particle motion (3.14), and by neglecting higher order terms in equation (3.23).

The substantial (or material) derivative:

At this stage the definition of the so-called substantial (or material derivative) shall be recalled, since it is often used to transform the RTT when it is injected into the governing equations.

It is obtained using the chain rule:

$$\frac{D\phi}{Dt} = \frac{d\phi}{dt} = \frac{\partial\phi}{\partial t} + \frac{\partial\phi}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial\phi}{\partial x_2}\frac{dx_2}{dt} + \frac{\partial\phi}{\partial x_3}\frac{dx_3}{dt} \tag{3.29}$$

Or in Einstein notation:

$$\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + u_j\frac{\partial\phi}{\partial x_j} \tag{3.30}$$

The RTT as given in equation (3.27) can now be reformulated. Consider that the intensive variable ϕ is the product of the density ρ with another intensive variable Υ. Then the RTT reads,

$$\int\limits_{V_x(t)} \left\{ \frac{\partial \rho \Upsilon(\mathbf{x},t)}{\partial t} + \nabla \cdot (\mathbf{u}\rho\Upsilon(\mathbf{x},t)) \right\} \mathrm{dV_x} \tag{3.31}$$

This can be rearranged using the chain rule.

$$\int\limits_{V_x(t)} \left\{ \rho \left(\frac{\partial \Upsilon(\mathbf{x},t)}{\partial t} + \mathbf{u} \cdot \nabla(\Upsilon(\mathbf{x},t)) \right) + \Upsilon \left(\frac{\partial \rho(\mathbf{x},t)}{\partial t} + \nabla \cdot (\rho(\mathbf{x},t)\mathbf{u}) \right) \right\} \mathrm{dV_x} \tag{3.32}$$

As will be seen in the following section, the term in the second round brackets represents the continuity equation, which is equal to zero. Thus the RTT can be rewritten to yield,

$$\int\limits_{V_x(t)} \left\{ \rho \frac{D\Upsilon(\mathbf{x},t)}{Dt} \right\} \mathrm{dV_x} \tag{3.33}$$

Hence, whenever the intensive variable is a product of ρ with some other extensive variable, one ends up with an equation like equation (3.33. The author believes that this is probably the reason why the substancial derivative is also called material derivative, because this simplification of the RTT is due to the adherence of some intensive variable to a set of mass points in a confined volume. The momentum and energy equations are the most prominent examples of intensive variables that are a product of ρ and another primitive variable.

3.1.2 Governing Equations for Non-Moving Reference Volumes in Inertial Frame of Reference

From equation (3.28), we can now derive the governing equations for non-moving reference volumes in inertial frame of reference.

- **Conservation of Mass**

  $\phi = \rho$

$$\frac{d}{dt}\int\limits_{V_x(t)} \rho \mathrm{dV_x} = 0 \tag{3.34}$$

or,

$$\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\cdot(\rho\mathbf{u})\,\mathrm{dS_x} = 0 \tag{3.35}$$

or,

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{3.36}$$

- **Conservation of Momentum**

  $\phi = \rho\mathbf{u}$

$$\frac{d}{dt}\int\limits_{V_x(t)} \rho\mathbf{u}\,\mathrm{dV_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot\underline{\underline{\sigma}}\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\mathrm{dV_x} \tag{3.37}$$

or,

$$\int\limits_{V_x(t)} \frac{\partial\rho\mathbf{u}}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\cdot(\rho\mathbf{u}\mathbf{u})\,\mathrm{dS_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot\underline{\underline{\sigma}}\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\mathrm{dV_x} \tag{3.38}$$

or,

$$\int\limits_{V_x(t)} \rho\frac{D\mathbf{u}}{Dt}\,\mathrm{dV_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot\underline{\underline{\sigma}}\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\mathrm{dV_x} \tag{3.39}$$

or,

$$\rho\frac{D\mathbf{u}}{Dt} = \nabla\cdot\underline{\underline{\sigma}} + \rho\mathbf{g} \tag{3.40}$$

- **Conservation of Energy**

  $\phi = e + \frac{\mathbf{u}^2}{2}$

$$\frac{d}{dt}\int\limits_{V_x(t)} \rho\left(e+\frac{\mathbf{u}^2}{2}\right)\,\mathrm{dV_x} = \oint\limits_{V_x(t)} \mathbf{n}\cdot(\underline{\underline{\sigma}}\cdot\mathbf{u})\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\cdot\mathbf{u}\mathrm{dV_x} - \oint\limits_{V_x(t)} \mathbf{n}\cdot\mathbf{q}\mathrm{dS_x} + \int\limits_{V_x(t)} Q\,\mathrm{dV_x} \quad (3.41)$$

or,

$$\int\limits_{V_x(t)} \frac{\partial\rho\left(e+\frac{\mathbf{u}^2}{2}\right)}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\left(\rho\cdot\left(e+\frac{\mathbf{u}^2}{2}\right)\right)\,\mathrm{dS_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot(\underline{\underline{\sigma}}\cdot\mathbf{u})\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\cdot\mathbf{u}\mathrm{dV_x} - \oint\limits_{S_x(t)} \mathbf{n}\cdot\mathbf{q}\mathrm{dS_x} + \int\limits_{V_x(t)} Q\,\mathrm{dV} \quad (3.42)$$

or,

$$\int\limits_{V_x(t)} \rho\frac{D\left(e+\frac{\mathbf{u}^2}{2}\right)}{Dt}\,\mathrm{dV_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot(\underline{\underline{\sigma}}\cdot\mathbf{u})\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\cdot\mathbf{u}\mathrm{dV_x} - \oint\limits_{S_x(t)} \mathbf{n}\cdot\mathbf{q}\mathrm{dS_x} + \int\limits_{V_x(t)} Q\,\mathrm{dV_x} \quad (3.43)$$

or,

$$\rho\frac{D\left(e+\frac{\mathbf{u}^2}{2}\right)}{Dt} = \nabla\cdot(\underline{\underline{\sigma}}\cdot\mathbf{u}) + (\rho\mathbf{g})\cdot\mathbf{u} - \nabla\cdot\mathbf{q} + Q \quad (3.44)$$

Governing Equations with Newton-Stokes Hypothesis

Now the constitutive laws of the fluid shall be included (following Newton's and Stokes hypothesises), and then we shall reformulate the energy equation to obtain a simple and nice to handle form of it, as written down by Steiner [8] or Anderson [9]. We shall also derive an equation for the total enthalpy, since it is of practical importance. To make our lives easier the Navier-Stokes equations shall be outlined in Einstein notation.

$$\frac{D\rho}{Dt} + \rho\frac{\partial u_j}{\partial x_j} = 0 \tag{3.45}$$

$$\rho\frac{Du_i}{Dt} = \frac{\partial}{\partial x_j}\left(-p\delta_{ij} - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) + \rho g_i \tag{3.46}$$

$$\rho\frac{D\left(e + \frac{\mathbf{u}^2}{2}\right)}{Dt} = \frac{\partial}{\partial x_j}\left(u_i\left(-p\delta_{ij} - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right)\right) + (\rho g_i)u_i - \frac{\partial q_j}{\partial x_j} + Q \tag{3.47}$$

Conservation of Energy

It is very often desirable to get rid of the $\rho\frac{D\frac{\mathbf{u}^2}{2}}{Dt}$ term in the energy equation (3.44).

It can easily be seen that this is equivalent to:

$$\rho\frac{D\frac{\mathbf{u}^2}{2}}{Dt} = \rho u_i\frac{Du_i}{Dt} \tag{3.48}$$

Which is nothing else than the velocity dotted with the LHS of the momentum equation (3.40). Hence:

$$\rho\frac{D\frac{\mathbf{u}^2}{2}}{Dt} = u_i\left(\frac{\partial}{\partial x_j}\left(-p\delta_{ij} - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) + \rho g_i\right) \tag{3.49}$$

It can be seen that when equation (3.49) is inserted into equation (3.44) the part of the potential energy in the energy equation will vanish.

$$\rho\frac{De}{Dt} = \frac{\partial}{\partial x_j}\left(u_i\sigma_{ij}\right) - u_i\left(\frac{\partial}{\partial x_j}\sigma_{ij}\right) - \frac{\partial q_j}{\partial x_j} + Q \tag{3.50}$$

Applying the chain rule to the first two terms on the RHS yields:

$$\rho\frac{De}{Dt} = \frac{\partial u_i}{\partial x_j}\left(-p\delta_{ij} - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) - \frac{\partial q_j}{\partial x_j} + Q \tag{3.51}$$

And finally the energy equation can be written in a short and convenient form:

$$\rho\frac{De}{Dt} = -p\frac{\partial u_j}{\partial x_j} - \frac{2}{3}\mu\left(\frac{\partial u_k}{\partial x_k}\right)^2 + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} - \frac{\partial q_j}{\partial x_j} + Q \tag{3.52}$$

Now 5 equations have been derived (continuity, momentum in three directions, energy) for 7 unknowns $(u_1, u_2, u_3, p, e, q, \rho)$. In this work only fluids with constant density shall be considered, hence only two more relations need to be added, which will link the temperature T to the internal energy e and the heat flux q. Therefore we shall apply Fourier's law of heat transfer and specific heat coefficients as outlined in equations(2.13,2.14,2.12).

$$\rho c_v\frac{DT}{Dt} = -p\frac{\partial u_j}{\partial x_j} - \frac{2}{3}\mu\left(\frac{\partial u_k}{\partial x_k}\right)^2 + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} - \lambda\frac{\partial^2 T}{\partial x_j \partial x_j} + Q \tag{3.53}$$

For the special case of incompressible flow with constant density we obtain:

$$\rho c_v\frac{DT}{Dt} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} + \lambda\frac{\partial^2 T}{\partial {x_j}^2} + Q \tag{3.54}$$

Conservation of Enthalpy

Sometimes it is more convenient to write the energy equation in terms of enthalpy.

The specific total enthalpy is defined as:

$$h = e + \frac{p}{\rho} \tag{3.55}$$

Hence the extensive total enthalpy reads:

$$H_{tot} = \int\limits_{V_x(t)} \rho h_{tot} \, \mathrm{dV_x} \tag{3.56}$$

Following Steiner [8], the substantial derivatives of equation(3.55,3.56) can be related with the internal energy as follows:

$$\rho \frac{Dh}{Dt} = \rho \frac{De}{Dt} + \rho \frac{D\left(\frac{p}{\rho}\right)}{Dt} \tag{3.57}$$

Rearranging this equation applying the chain rule to the third term on the RHS yields:

$$\rho \frac{De}{Dt} = \rho \frac{Dh}{Dt} - \frac{Dp}{Dt} + \frac{p}{\rho}\frac{D\rho}{Dt} \tag{3.58}$$

Using the conservation of continuity (equation (3.45)) gives:

$$\rho \frac{De}{Dt} = \rho \frac{Dh}{Dt} - \frac{Dp}{Dt} - p\frac{\partial u_j}{\partial x_j} \tag{3.59}$$

The LHS of equation(3.59) is already familiar to us, since it is equivalent to the LHS of equation (3.52). Therefore the RHS of equation(3.59) just has to be set equal to the RHS of equation (3.52) to obtain,

$$\rho \frac{Dh}{Dt} - \frac{Dp}{Dt} - p\frac{\partial u_j}{\partial x_j} = -p\frac{\partial u_j}{\partial x_j} - \frac{2}{3}\mu \left(\frac{\partial u_k}{\partial x_k}\right)^2 + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} - \frac{\partial q_j}{\partial x_j} + Q \tag{3.60}$$

Rearranging above equation and applying the chain rule finally leads to a short and convenient form for the enthalpy equation.

$$\rho\frac{Dh}{Dt} = \frac{\partial p}{\partial t} + u_j\frac{\partial p}{\partial x_j} - \frac{2}{3}\mu\left(\frac{\partial u_k}{\partial x_k}\right)^2 + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} - \frac{\partial q_j}{\partial x_j} + Q \tag{3.61}$$

Introducing Fourier's law and specific heat coefficients at constant pressure c_p, an equation that closes the system of Navier-Stokes equations is obtained.

$$\rho c_p\frac{DT}{Dt} = \frac{\partial p}{\partial t} + u_j\frac{\partial p}{\partial x_j} - \frac{2}{3}\mu\left(\frac{\partial u_k}{\partial x_k}\right)^2 + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} + \lambda\frac{\partial^2 T}{\partial {x_j}^2} + Q \tag{3.62}$$

For the special case of incompressible flow with constant density the following formulation is obtained.

$$\rho c_p\frac{DT}{Dt} = \frac{\partial p}{\partial t} + u_j\frac{\partial p}{\partial x_j} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} + \lambda\frac{\partial^2 T}{\partial {x_j}^2} + Q \tag{3.63}$$

3.2 Moving Reference Volumes In Inertial Frame Of Reference

3.2.1 The Transport Theorem for Moving Reference Cells in Inertial Frame of Reference

The aim of this section is to obtain a transport theorem with which an integration over a static material reference volume can be replaced by an integration over a moving spatial reference volume (moving mesh cell).

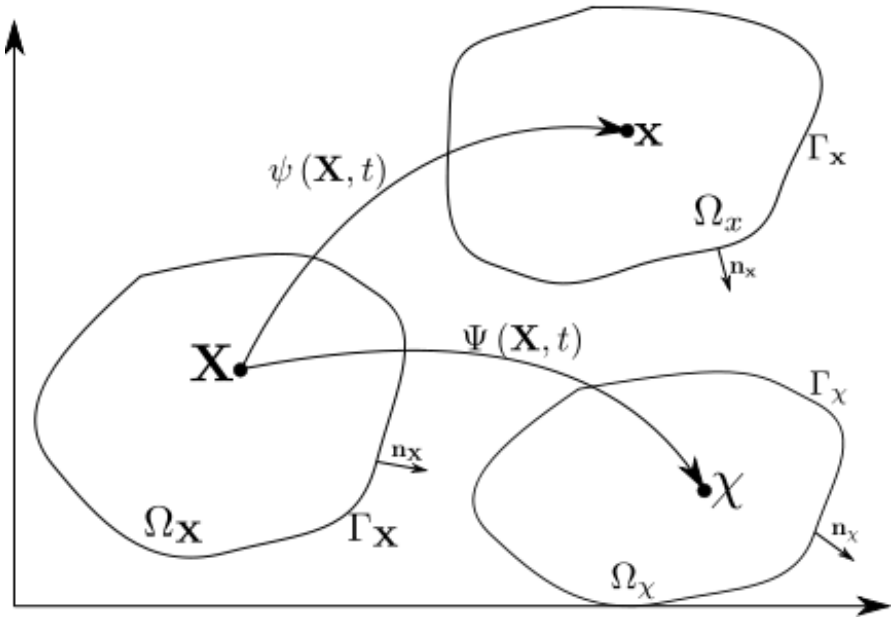

Figure 3.2: Material control volume V_x, moving reference control volume V_χ and static control volume V_X

The transport theorem (RTT equation(3.27)) that we have obtained previously shall now be adapted in a way that allows us to calculate the local time rate of change of a control volume, which moves arbitrarily in space. Therefore we shall recall the RTT, equation(3.27).

$$\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \left\{ \frac{\partial \phi(\mathbf{x},t)}{\partial t} + \nabla \cdot (\phi(\mathbf{x},t)\mathbf{u}) \right\} \mathrm{dV_x} \tag{3.64}$$

The local rate of change $\frac{\partial\phi(\mathbf{x},t)}{\partial t}$ can be seen as the local time rate of change of a static control volume in absolute frame of reference. This control volume does not change its position in time, nor does it change its shape. If we now want to evaluate the integral of the intensive variable ϕ over a moving control volume V_x we have to relate the local time rate of change from the Eulerian reference volume (at fixed coordinate $\mathbf{X}$) with the local time rate of change seen from within a moving reference volume (at coordinate χ). Following Gehrer [10] this can be done as follows,

$$\left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} = \frac{\phi\left(\chi(t),t+dt\right)-\phi\left(\chi(t),t\right)}{dt} = \frac{\phi\left(\mathbf{X}+d\mathbf{x},t+dt\right)-\phi\left(\mathbf{X},t\right)}{dt} \tag{3.65}$$

By doing a Taylor series expansion we get,

$$\left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} = \frac{\phi\left(\mathbf{X},t\right)+\frac{\partial\phi(\mathbf{X},t)}{\partial t}dt+\frac{\partial\phi(\mathbf{X},t)}{\partial x_1}dx_1+\frac{\partial\phi(\mathbf{X},t)}{\partial x_2}dx_2+\frac{\partial\phi(\mathbf{X},t)}{\partial x_2}dx_3-\phi\left(\mathbf{X},t\right)}{dt} \tag{3.66}$$

Through a linearised approach $\chi = \Psi(\mathbf{X},t)$ can be expressed as follows:

$$\begin{aligned}\chi_1 &= X_1 + u_{g,1}dt\\ \chi_2 &= X_2 + u_{g,2}dt\\ \chi_3 &= X_3 + u_{g,3}dt\end{aligned} \tag{3.67}$$

From equation(3.67) it can be seen that $d\mathbf{x}$ in equation(3.66) can be replaced by $\mathbf{u}_g \cdot dt$. Equation(3.66) can be further simplified to yield,

$$\left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} = \frac{\frac{\partial\phi(\mathbf{X},t)}{\partial t}dt+\frac{\partial\phi(\mathbf{X},t)}{\partial x_1}u_{g1}dt+\frac{\partial\phi(\mathbf{X},t)}{\partial x_2}u_{g2}dt+\frac{\partial\phi(\mathbf{X},t)}{\partial x_3}u_{g3}dt}{dt} \tag{3.68}$$

Thereof we obtain the relation for the local time rate of change between the absolute and the

moving reference control volumes.

$$\left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} = \left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\mathbf{x}} + \nabla\phi\left(\mathbf{x},t\right)\cdot\mathbf{u}_g \tag{3.69}$$

Or respectively,

$$\boxed{\left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\mathbf{x}} = \left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} - \nabla\phi\left(\mathbf{x},t\right)\cdot\mathbf{u}_g} \tag{3.70}$$

inserting this into equation(3.64) yields,

$$\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \left\{ \left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} - \nabla\phi\left(\mathbf{x},t\right)\cdot\mathbf{u}_g + \nabla\cdot\left(\phi(\mathbf{x},t)\mathbf{u}\right) \right\} \mathrm{dV_x} \tag{3.71}$$

The second term on the RHS of equation(3.71) can be transformed to yield,

$$\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \left\{ \left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} - \underbrace{\nabla\cdot\left(\phi\left(\mathbf{x},t\right)\mathbf{u}_g\right)}_{*} + \underbrace{\phi\nabla\cdot\mathbf{u}_g}_{**} + \nabla\cdot\left(\phi(\mathbf{x},t)\mathbf{u}\right) \right\} \mathrm{dV_x} \tag{3.72}$$

The additional term that had been introduced due to the motion of the reference volume has been split in two parts in equation (3.72). The first term (*) accounts for the movement of the control volume, the second term (**) accounts for the control volume dilatation.

Equation (3.72) can be further simplified to yield:

$$\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \left\{ \left(\frac{\partial\phi(\mathbf{x},t)}{\partial t}\right)_{\chi} + \nabla\cdot\left(\phi(\mathbf{x},t)\left(\mathbf{u}-\mathbf{u}_g\right)\right) + \phi\nabla\cdot\mathbf{u}_g \right\} \mathrm{dV_x} \tag{3.73}$$

And finally a transport theorem for a moving control volume in the inertial frame of reference is obtained.

$$\boxed{\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \left(\frac{\partial \phi(\mathbf{x},t)}{\partial t}\right)_\chi \mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\cdot\left(\phi(\mathbf{x},t)\left(\mathbf{u}-\mathbf{u}_g\right)\right)\,\mathrm{dS_x} + \int\limits_{V_x(t)} \phi\nabla\cdot\mathbf{u}_g\,\mathrm{dV_x}} \tag{3.74}$$

This description is commonly known as Arbitrary Lagrange Euler (ALE) description, since the Lagrangian description is obtained if $\mathbf{u}_g = \mathbf{u}$, and the Eulerian description is obtained if $\mathbf{u}_g = 0$.

3.2.2 Governing Equations for Moving Reference Volumes in Inertial Frame of Reference

In this subsection the governing equations for moving reference volumes in inertial frame of reference shall be derived. This means that the equations are valid for the integration over volumes that move within the inertial reference frame's domain.

- **Conservation of Mass**

  $\phi = \rho$

$$\frac{d}{dt}\int\limits_{V_x(t)} \rho\mathrm{dV_x} = 0 \tag{3.75}$$

or,

$$\int\limits_{S_x(t)} \frac{\partial\rho}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\cdot\left(\rho\left(\mathbf{u}-\mathbf{u}_g\right)\right)\,\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\nabla\cdot\mathbf{u}_g\,\mathrm{dV_x} = 0 \tag{3.76}$$

or,

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\left(\rho\left(\mathbf{u}-\mathbf{u}_g\right)\right) + \rho\nabla\cdot\mathbf{u}_g = 0 \tag{3.77}$$

In the case of non-deforming meshes and rigid body motion the continuity equation in integral form reduces to.

$$\boxed{\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\cdot(\rho\mathbf{u})\,\mathrm{dS_x} = 0} \tag{3.78}$$

- **Conservation of Momentum**

  $\phi = \rho\mathbf{u}$

$$\frac{d}{dt}\int\limits_{V_x(t)} \rho\mathbf{u}\,\mathrm{dV_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot\underline{\underline{\sigma}}\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\mathrm{dV_x} \tag{3.79}$$

or,

$$\int\limits_{V_x(t)} \frac{\partial\rho\mathbf{u}}{\partial t}\,\mathrm{dV_x} + \oint\limits_{V_x(t)} \mathbf{n}\cdot\left(\rho\mathbf{u}\left(\mathbf{u}-\mathbf{u}_g\right)\right)\,\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{u}\nabla\cdot\mathbf{u}_g\,\mathrm{dV_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot\underline{\underline{\sigma}}\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\mathrm{dV_x} \tag{3.80}$$

or,

$$\frac{\partial\rho\mathbf{u}}{\partial t} + \nabla\cdot\left(\rho\mathbf{u}\left(\mathbf{u}-\mathbf{u}_g\right)\right) + \rho\mathbf{u}\nabla\cdot\mathbf{u}_g = \nabla\cdot\underline{\underline{\sigma}} + \rho\mathbf{g} \tag{3.81}$$

In the case of non-deforming meshes and rigid body motion the momentum equation in integral form reduces to.

$$\boxed{\int\limits_{V_x(t)} \frac{\partial\rho\mathbf{u}}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n}\cdot\left(\rho\mathbf{u}\left(\mathbf{u}-\mathbf{u}_g\right)\right)\,\mathrm{dS_x} = \oint\limits_{S_x(t)} \mathbf{n}\cdot\underline{\underline{\sigma}}\mathrm{dS_x} + \int\limits_{V_x(t)} \rho\mathbf{g}\mathrm{dV_x}} \tag{3.82}$$

3.3 Non-Moving Reference Volumes In Rotating Frame Of Reference

In this section the equations of motion for incompressible flows with constant density shall be derived. Therefore the kinematics of relative rotational motion, which relates velocities and accelerations seen from an inertial frame and a rotational reference frame will be outlined following Hauger et al. [11]. Subsequently the Reynolds transport theorem will be adapted to obtain a transport theorem for non-moving reference volumes in rotational frame notation.

3.3.1 The Kinematics of Relative Motion

Following closely Hauger [11] we shall start deriving the motion of a point in space in respect to an inertial Cartesian coordinate system (with coordinates x,y,z) and a relative coordinate system (with coordinates ξ, η, ζ and basis vectors $\mathbf{e}_\xi$, $\mathbf{e}_\eta$, $\mathbf{e}_\zeta$). The relative coordinate system performs a motion decomposed into a translational and rotational part. Figure 3.3 shows a material point **P** and its positioning in respect to each coordinate system.

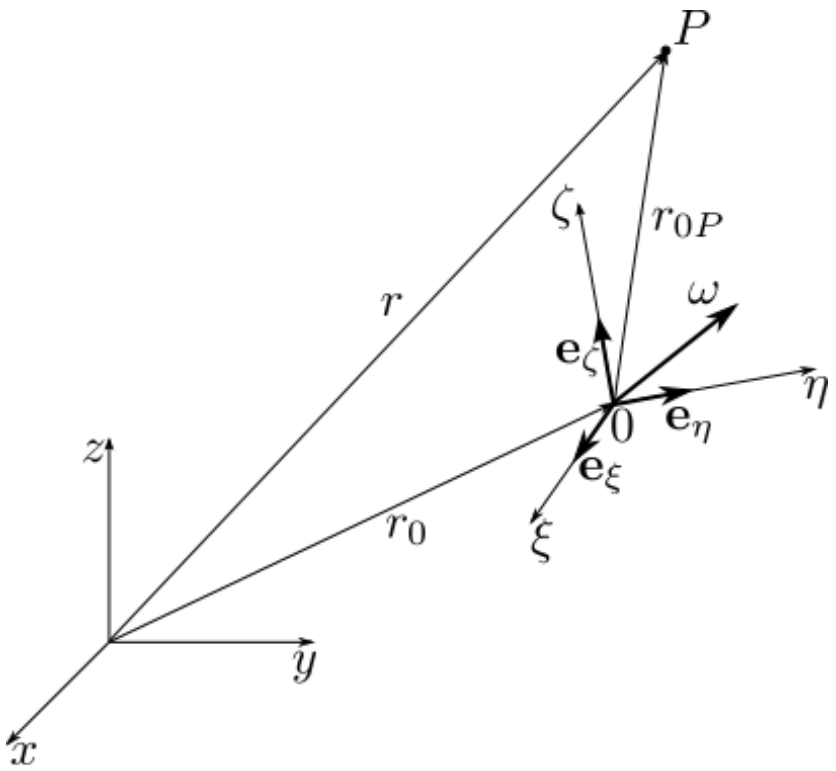

Figure 3.3: Material point in inertial and relative reference frames

The position vector is defined as,

$$\mathbf{r} = \mathbf{r}_0 + \mathbf{r}_{0P} \tag{3.83}$$

where $\mathbf{r}_0$ relates the origins of both coordinate systems and $\mathbf{r}_{0P}$ defines the position of the material point seen from the relative reference frames perspective. Written in basis vector notation this reads,

$$\mathbf{r}_{0P} = \xi\mathbf{e}_\xi + \eta\mathbf{e}_\eta + \zeta\mathbf{e}_\zeta \tag{3.84}$$

The absolute velocity (the velocity measured in the inertial reference system) of a material point that moves in space can be written as,

$$\mathbf{u}_a = \dot{\mathbf{r}} = \dot{\mathbf{r}}_0 + \dot{\mathbf{r}}_{0P} \tag{3.85}$$

The first term on the RHS of equation (3.85) is the translational velocity of the origin of the relative reference frame in respect to the inertial reference frame.

$$\dot{\mathbf{r}}_0 = \mathbf{u}_0 \tag{3.86}$$

Since the direction of the relative system's basis vectors changes with time due to a rotation relative to the inertial frame the second term on the RHS yields,

$$\dot{\mathbf{r}}_{0P} = (\dot{\xi}\mathbf{e}_\xi + \dot{\eta}\mathbf{e}_\eta + \dot{\zeta}\mathbf{e}_\zeta) + (\xi\dot{\mathbf{e}}_\xi + \eta\dot{\mathbf{e}}_\eta + \zeta\dot{\mathbf{e}}_\zeta) \tag{3.87}$$

The relative system rotates with an angular velocity Ω. Hence the time rate of change of the relative basis vectors reads,

$$\dot{\mathbf{e}}_\xi = \Omega \times \mathbf{e}_\xi \quad \dot{\mathbf{e}}_\eta = \Omega \times \mathbf{e}_\eta \quad \dot{\mathbf{e}}_\zeta = \Omega \times \mathbf{e}_\zeta \tag{3.88}$$

The second term on the RHS of equation (3.87) can therefore be rewritten,

$$\begin{aligned}(\xi\dot{\mathbf{e}}_\xi + \eta\dot{\mathbf{e}}_\eta + \zeta\dot{\mathbf{e}}_\zeta) =& \xi\Omega \times \mathbf{e}_\xi + \eta\Omega \times \mathbf{e}_\eta + \zeta\Omega \times \mathbf{e}_\zeta \\ & \Omega \times (\xi\mathbf{e}_\xi + \eta\mathbf{e}_\eta + \zeta\mathbf{e}_\zeta) = \Omega \times \mathbf{r}_{0P}\end{aligned} \tag{3.89}$$

The first term on the RHS of equation (3.87) denotes the time rate of change of the material point $\mathbf{P}$ in respect to the relative reference system (*).

$$\frac{d^*\mathbf{r}_{0P}}{dt} = (\dot{\xi}\mathbf{e}_\xi + \dot{\eta}\mathbf{e}_\eta + \dot{\zeta}\mathbf{e}_\zeta) \tag{3.90}$$

Inserting equations (3.89)(3.90)(3.86) in equation (3.85) yields:

$$\mathbf{u}_a = \mathbf{u}_0 + \Omega \times \mathbf{r}_{0P} + \frac{d^*\mathbf{r}_{0P}}{dt} \tag{3.91}$$

The first two terms on the RHS of equation (3.91) are the relative translational and rotational velocities, which describe the movement of the relative reference system in respect to the inertial reference system. These two velocities shall be called leading velocities $\mathbf{u}_l$. The time rate of change of $\mathbf{r}_{0P}$ is nothing else than the velocity measured from within the relative reference system and shall be denoted relative velocity $\mathbf{u}_r$.

The absolute velocity of a material point $\mathbf{P}$ can therefore be written as:

$$\mathbf{u}_a = \mathbf{u}_l + \mathbf{u}_r \tag{3.92}$$

With,

$$\begin{aligned} \mathbf{u}_l &= \mathbf{u}_0 + \Omega \times \mathbf{r}_{0P} \\ \mathbf{u}_r &= \frac{d^*\mathbf{r}_{0P}}{dt} \end{aligned} \tag{3.93}$$

Before deriving the relation of the absolute and relative acceleration of a material point,the relation between the time rate of change of a general vector $\mathbf{v}$ in the inertial and relative reference frames shall be pointed out. The notations of Figure 3.4 are taken to derive this relation.

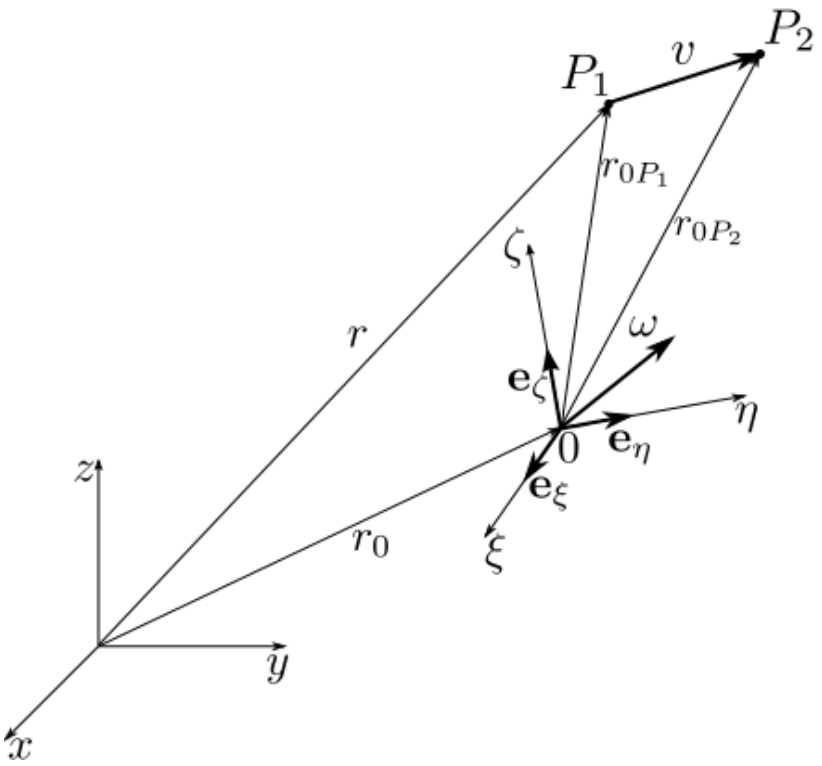

Figure 3.4: General vector in inertial and relative reference frames

A general vector is defined as:

$$\mathbf{v} = (\mathbf{r}_{0P2} - \mathbf{r}_0) - (\mathbf{r}_{0P1} - \mathbf{r}_0) = \mathbf{r}_{0P2} - \mathbf{r}_{0P1} \tag{3.94}$$

The time rate of change of a vector in absolute reference frame written with basis vectors is:

$$\dot{\mathbf{v}} = (\dot{v}_\xi \mathbf{e}_\xi + \dot{v}_\eta \mathbf{e}_\eta + \dot{v}_\zeta \mathbf{e}_\zeta) + (v_\xi \dot{\mathbf{e}}_\xi + v_\eta \dot{\mathbf{e}}_\eta + v_\zeta \dot{\mathbf{e}}_\zeta) \tag{3.95}$$

The first term on the RHS of equation (3.95) is the time rate of change of a vector $\mathbf{v}$ seen from the relative frame of reference. The second term on the right hand side can be reformulated using equation (3.88). This yields,

$$\dot{\mathbf{v}} = \frac{d^*\mathbf{v}}{dt} + \Omega \times \mathbf{v} \tag{3.96}$$

Now that the relation, between the time rate of change of a general vector seen from an inertial and a relative point of view, has been derived, the derivation of the relation for the accelerations can be laid out.

The absolute acceleration, which is defined as the time rate of change of the absolute velocity in respect to the inertial reference frame.

$$\mathbf{a}_a = \dot{\mathbf{u}}_l + \dot{\mathbf{u}}_r = \dot{\mathbf{u}}_0 + \dot{(\Omega \times \mathbf{r}_{0P})} + \dot{\mathbf{u}}_r \tag{3.97}$$

Equation (3.97) shall now be developed, starting by introducing the relation of equation (3.96) to the third term on the RHS of equation (3.97),

$$\begin{aligned}
\mathbf{a}_a &= \dot{\mathbf{u}}_0 + \dot{(\Omega \times \mathbf{r}_{0P})} + \frac{d^*\mathbf{u}_r}{dt} + \Omega \times \mathbf{u}_r \\
&= \dot{\mathbf{u}}_0 + \dot{\Omega} \times \mathbf{r}_{0P} + \Omega \times \dot{\mathbf{r}_{0P}} + \frac{d^*\mathbf{u}_r}{dt} + \Omega \times \mathbf{u}_r \\
&= \dot{\mathbf{u}}_0 + \dot{\Omega} \times \mathbf{r}_{0P} + \Omega \times \left(\frac{d^*\mathbf{r}_{0P}}{dt} + \Omega \times \mathbf{r}_{0P} \right) + \frac{d^*\mathbf{u}_r}{dt} + \Omega \times \mathbf{u}_r \\
&= \dot{\mathbf{u}}_0 + \dot{\Omega} \times \mathbf{r}_{0P} + \Omega \times (\mathbf{u}_r + \Omega \times \mathbf{r}_{0P}) + \frac{d^*\mathbf{u}_r}{dt} + \Omega \times \mathbf{u}_r \\
&= \underbrace{\dot{\mathbf{u}}_0 + \dot{\Omega} \times \mathbf{r}_{0P} + \Omega \times (\Omega \times \mathbf{r}_{0P})}_{\mathbf{a}_l} + \underbrace{\frac{d^*\mathbf{u}_r}{dt}}_{\mathbf{a}_r} + \underbrace{2\Omega \times \mathbf{u}_r}_{\mathbf{a}_c}
\end{aligned} \tag{3.98}$$

In short notation,

$$\mathbf{a}_a = \mathbf{a}_l + \mathbf{a}_r + \mathbf{a}_c \tag{3.99}$$

with,

$$\mathbf{a}_l = \dot{\mathbf{u}}_0 + \dot{\Omega} \times \mathbf{r}_{0P} + \Omega \times (\Omega \times \mathbf{r}_{0P}) \tag{3.100}$$

The so-called leading acceleration $\mathbf{a}_l$ is the acceleration that the material point would experience

if it was rigidly connected with the relative coordinate system. The relative acceleration $\mathbf{a}_r$ is the acceleration that would be measured from within the relative coordinate system. The remaining term is the well-known Coriolis acceleration $\mathbf{a}_c$.

3.3.2 The Reynolds Transport Theorem for Non-Moving Reference Volumes in Relative Frame of Reference

The Reynolds Transport Theorem for non-moving reference volumes in relative frame of reference is derived almost identically to the RTT described in section (3.1). The only difference being the mapping function that relates the fixed coordinates $\mathbf{X}$ of the non-moving volumes in the relative reference frame with the Lagrangian coordinates of the material points, which change as these material points move.

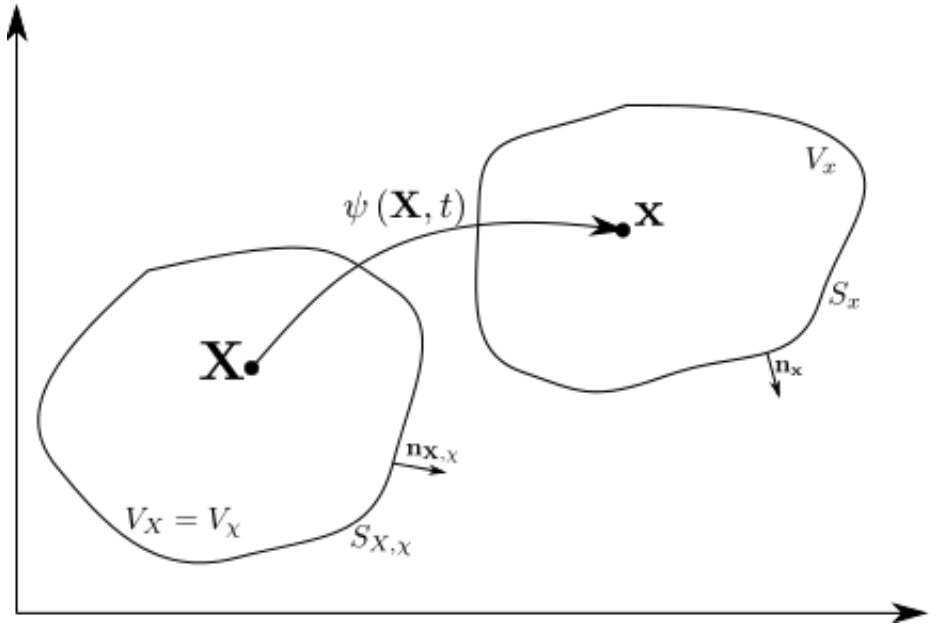

Figure 3.5: Material control volume and static reference volume in a relative reference frame

The mapping function that relates the moving material control volume with the static reference volume is given as,

$$\mathbf{x} = \psi(\mathbf{X}, t) \tag{3.101}$$

Keeping in mind that, in contrast to the derivation of the RTT for inertial reference frames, the coordinate $\mathbf{X}$ is now being fixed within the relative coordinate system. The name of the coordinate has not been changed for the sake of brevity.

Through a linearised approach the mapping function $\mathbf{x} = \psi(\mathbf{X}, t)$ can be expressed as follows:

$$\begin{aligned} x_1 &= X_1 + u_{r,1} \cdot \Delta t \\ x_2 &= X_2 + u_{r,2} \cdot \Delta t \\ x_3 &= X_3 + u_{r,3} \cdot \Delta t \end{aligned} \tag{3.102}$$

The mapping function is almost identical to the mapping function chosen in section (3.1), the only difference being that the mapping is defined with the relative velocity $\mathbf{u}_r$ instead of the inertial velocity $\mathbf{u}$.

From here onwards the derivation of the RTT for relative frames of reference is identical to that for inertial frames. Only the inertial velocity $\mathbf{u}$ has to be substituted with the relative velocity $\mathbf{u}_r$, this yields:

$$\boxed{\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \frac{\partial \phi(\mathbf{x}, t)}{\partial t} \, \mathrm{dV_x} + \oint\limits_{V_x(t)} \nabla \cdot (\phi(\mathbf{x}, t)\mathbf{u}_r) \, \mathrm{dV_x}} \tag{3.103}$$

$$\boxed{\frac{d\Phi(t)}{dt} = \int\limits_{V_x(t)} \frac{\partial \phi(\mathbf{x}, t)}{\partial t} \, \mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n} \cdot (\phi(\mathbf{x}, t)\mathbf{u}_r) \, \mathrm{dS_x}} \tag{3.104}$$

3.3.3 Governing Equations for Non-Moving Reference Volumes in Relative Rotational Frame of Reference

In this subsection we shall derived the governing equations for non-moving reference volumes in relative rotational frame of reference. This means that the equations are valid for the integration over volumes which are fixed in respect to a relative rotational coordinate system. The relative rotational frame of reference is a special case of the relative frame of reference, where the origin of the relative coordinate system doesn't move ($\mathbf{u}_0 = 0$). Hence equation(3.91) simplifies to,

$$\mathbf{u}_a = \mathbf{u}_r + \Omega \times \mathbf{r}_{0P} \tag{3.105}$$

Furthermore the rotational velocity Ω shall not vary in time.

$$\dot{\Omega} = 0 \tag{3.106}$$

From equation (3.28), we can now derive the governing equations for non-moving reference volumes in relative rotational frame of reference.

- **Conservation of Mass**

  $\phi = \rho$

$$\frac{d}{dt} \int\limits_{V_x(t)} \rho \mathrm{dV_x} = 0 \tag{3.107}$$

or,

$$\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t} \, \mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n} \cdot (\rho \mathbf{u}_r) \, \mathrm{dS_x} = 0 \tag{3.108}$$

Using the relation of equation (3.105) the continuity equation can also be written in terms of absolute velocities.

$$\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t}\,\mathrm{dV_x} + \int\limits_{V_x(t)} \nabla \cdot (\rho(\mathbf{u}_r))\,\mathrm{dS_x} = 0$$

$$\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t}\,\mathrm{dV_x} + \int\limits_{V_x(t)} \nabla \cdot (\rho(\mathbf{u}_a - (\Omega \times \mathbf{r}_{0P})))\,\mathrm{dV_x} = 0$$

$$\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t}\,\mathrm{dV_x} + \int\limits_{V_x(t)} \nabla \cdot (\rho \mathbf{u}_a) - \underbrace{\nabla \cdot (\rho(\Omega \times \mathbf{r}_{0P}))}_{0}\,\mathrm{dV_x} = 0$$

Finally the continuity equation reads:

$$\boxed{\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t}\,\mathrm{dV_x} + \int\limits_{V_x(t)} \nabla \cdot (\rho \mathbf{u}_a)\,\mathrm{dV_x} = 0} \tag{3.109}$$

or,

$$\boxed{\int\limits_{V_x(t)} \frac{\partial \rho}{\partial t}\,\mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n} \cdot (\rho \mathbf{u}_a)\,\mathrm{dS_x} = 0} \tag{3.110}$$

- **Conservation of Momentum**

  $\phi = \rho \mathbf{u}_a$

Starting from the Lagrangian perspective, for a mass point the momentum equation reads,

$$\frac{d\left(\rho \mathbf{u}_a\right)}{dt} = \sum \mathbf{F} \tag{3.111}$$

Taking the continuity equation into account,

$$\underbrace{\frac{d\rho}{dt}}_{0} \mathbf{u}_a + \rho \frac{d\mathbf{u}_a}{dt} = \sum \mathbf{F} \tag{3.112}$$

$$\rho \mathbf{a}_a = \sum \mathbf{F} \tag{3.113}$$

Equations (3.99)(3.100) inserted into equation (3.113) gives:

$$\rho \left(\dot{\mathbf{u}}_0 + \dot{\Omega} \times \mathbf{r}_{0P} + \Omega \times (\Omega \times \mathbf{r}_{0P}) + \frac{d^* \mathbf{u}_r}{dt} + 2\Omega \times \mathbf{u}_r \right) = \sum \mathbf{F} \tag{3.114}$$

For a non-moving relative rotational coordinate origin and a constant rotational velocity the first two terms inside the brackets vanish.

$$\rho \left(\Omega \times (\Omega \times \mathbf{r}_{0P}) + \frac{d^* \mathbf{u}_r}{dt} + 2\Omega \times \mathbf{u}_r \right) = \sum \mathbf{F} \tag{3.115}$$

And rearranged:

$$\rho \left(\frac{d^* \mathbf{u}_r}{dt} + \Omega \times (\Omega \times \mathbf{r}_{0P}) + 2\Omega \times \mathbf{u}_r \right) = \sum \mathbf{F} \tag{3.116}$$

Now the obtained Lagrangian equation for a single mass point are transformed to an integral form.

$$\rho\left(\frac{d^*}{dt}\int\limits_{V_x(t)}\mathbf{u}_r\,\mathrm{dV_x}+\int\limits_{V_x(t)}\Omega\times(\Omega\times\mathbf{r}_{0P})+2\Omega\times\mathbf{u}_r\,\mathrm{dV_x}\right)=\sum\mathbf{F}\tag{3.117}$$

Applying the RTT of equation (3.104) to the first term inside the brackets yields:

$$\rho\left(\int\limits_{V_x(t)}\frac{\partial\mathbf{u}_r}{\partial t}+\nabla\cdot(\mathbf{u}_r\mathbf{u}_r)\;\mathrm{dV_x}+\int\limits_{V_x(t)}\Omega\times(\Omega\times\mathbf{r}_{0P})+2\Omega\times\mathbf{u}_r\,\mathrm{dV_x}\right)=\sum\mathbf{F}\tag{3.118}$$

Now the LHS of above equation can be transformed in order to obtain a set of momentum equations for rotational frames written in terms of absolute velocity.

$$\rho\left(\int\limits_{V_x(t)}\frac{\partial\mathbf{u}_r}{\partial t}+\nabla\cdot(\mathbf{u}_r\left[\mathbf{u}_a-\Omega\times\mathbf{r}_{0P}\right])+\Omega\times(\Omega\times\mathbf{r}_{0P})+2\Omega\times\mathbf{u}_r\,\mathrm{dV_x}\right)$$

$$\rho\left(\int\limits_{V_x(t)}\frac{\partial\mathbf{u}_r}{\partial t}+\nabla\cdot(\mathbf{u}_r\mathbf{u}_a)-\underbrace{\nabla\cdot\mathbf{u}_r\left(\Omega\times\mathbf{r}_{0P}\right)}_{0}-\mathbf{u}_r\cdot\nabla\left(\Omega\times\mathbf{r}_{0P}\right)+\Omega\times(\Omega\times\mathbf{r}_{0P})+2\Omega\times\mathbf{u}_r\,\mathrm{dV_x}\right)$$

$$\rho\left(\int\limits_{V_x(t)}\frac{\partial\mathbf{u}_r}{\partial t}+\nabla\cdot(\mathbf{u}_r\mathbf{u}_a)-\Omega\times\mathbf{u}_r+\Omega\times(\Omega\times\mathbf{r}_{0P})+2\Omega\times\mathbf{u}_r\,\mathrm{dV_x}\right)$$

$$\rho\left(\int\limits_{V_x(t)}\frac{\partial\mathbf{u}_r}{\partial t}+\nabla\cdot(\mathbf{u}_r\mathbf{u}_a)+\Omega\times\mathbf{u}_r+\Omega\times(\Omega\times\mathbf{r}_{0P})\;\mathrm{dV_x}\right)$$

$$\rho\left(\int\limits_{V_x(t)}\frac{\partial\mathbf{u}_r}{\partial t}+\nabla\cdot(\mathbf{u}_r\mathbf{u}_a)+\Omega\times(\mathbf{u}_r+\Omega\times\mathbf{r}_{0P})\;\mathrm{dV_x}\right)$$

$$\rho \left(\int\limits_{V_x(t)} \frac{\partial \mathbf{u}_a}{\partial t} + \underbrace{\frac{\partial (\Omega \times \mathbf{r}_{0P})}{\partial t}}_{0} + \nabla \cdot (\mathbf{u}_r \mathbf{u}_a) + \Omega \times (\mathbf{u}_r + \Omega \times \mathbf{r}_{0P}) \, \mathrm{dV_x} \right)$$

$$\rho \left(\int\limits_{V_x(t)} \frac{\partial \mathbf{u}_a}{\partial t} + \nabla \cdot (\mathbf{u}_r \mathbf{u}_a) + \Omega \times (\mathbf{u}_r + \Omega \times \mathbf{r}_{0P}) \, \mathrm{dV_x} \right)$$

And finally,

$$\rho \left(\int\limits_{\Omega_x(t)} \frac{\partial \mathbf{u}_a}{\partial t} + \nabla \cdot (\mathbf{u}_r \mathbf{u}_a) + \Omega \times \mathbf{u}_a \, \mathrm{dV_x} \right) \tag{3.119}$$

The momentum equations for rotational reference frames and non-moving meshes, with constant rotational velocity and fixed rotation center can now be written as:

$$\boxed{\rho \left(\int\limits_{\Omega_x(t)} \frac{\partial \mathbf{u}_a}{\partial t} + \nabla \cdot (\mathbf{u}_r \mathbf{u}_a) + \Omega \times \mathbf{u}_a \, \mathrm{dV_x} \right) = \oint\limits_{\partial \Omega_x(t)} \mathbf{n} \cdot \underline{\underline{\sigma}} \mathrm{dS_x} + \int\limits_{\Omega_x(t)} \rho \mathbf{g} \mathrm{dV_x}} \tag{3.120}$$

Or,

$$\boxed{\rho \left(\int\limits_{\Omega_x(t)} \frac{\partial \mathbf{u}_a}{\partial t} \, \mathrm{dV_x} + \oint\limits_{\partial \Omega_x(t)} \mathbf{n} \cdot (\mathbf{u}_r \mathbf{u}_a) \, \mathrm{dS_x} + \int\limits_{\Omega_x(t)} \Omega \times \mathbf{u}_a \, \mathrm{dV_x} \right) = \oint\limits_{\partial \Omega_x(t)} \mathbf{n} \cdot \underline{\underline{\sigma}} \, \mathrm{dS_x} + \int\limits_{\Omega_x(t)} \rho \mathbf{g} \mathrm{dV_x}} \tag{3.121}$$

Chapter 4

Numerical Algorithms for Incompressible Flows

Contents

All roads lead to Rome. The same holds true for the solution of the Navier-Stokes equations. The solution procedure depends on many things, like the physical properties of the flow, the discretisation used, demanded accuracy, computational restrictions or whether steady or unsteady solutions are sought. There might even be many more parameters that define which algorithm is the most practical for a given problem. In the scope of this work, focus is laid on incompressible, single-phase flows, as they occur in hydraulic turbomachines. Effects like cavitation (multi-phase flow) shall not be taken into account. Also the gravitational force can be discarded since the Froud number, which relates inertial forces to gravitational forces, is very big in turbomachinery flows. For incompressible flows of constant density the to be solved variables of the Navier-Stokes equations reduce to velocity and pressure. This pressure-velocity

coupling is at the heart of commonly used incompressible algorithms. The resulting system of equations can be solved sequentially or simultaneously. Both approaches have their strengths and weaknesses. In this work a simultaneous coupled approach is presented, but before diving into the deep waters of numerics, the author believes that it makes sense to give an overview and background of mentioned procedures.

4.1 Historical Background

In computational fluid dynamics (CFD) two main paradigms exist on how to efficiently solve incompressible flows, namely sequential (segregated) algorithms and simultaneous (coupled) algorithms. No general answer can be found to the question, which of them is better, since both approaches have their problem-depending advantages and shortcomings. Segregated algorithms evaluate the momentum and continuity equation sequentially and couple them weakly by means of sub-looping. On the other hand coupled (also called block coupled) approaches solve both the continuity and the momentum equations simultaneously, which results in a strong variable coupling.

Ever since computers have been available for research, scientists have been developing methods for the numerical solution of the Navier-Stokes equations. This system of equations has the curious property that, for incompressible flows, the coupling of the primitive variables (pressure and velocity) between equations is weak, in the sense that the continuity equation does not contain the pressure variable. When compressible flows are treated, it is possible to come up with a system of equations, where each variable (primitive or conservative) is present in every governing equation. This property leads to coupled algorithms that are neatly derived and understood in a purely mathematical sense. However for incompressible flows, the coupling of equations is far of being a trivial matter (which shall not mean that compressible flows are trivial). The most productive period of algorithm development in the field of CFD was probably between the mid 60s and the mid 80s. A multitude of solvers that are still used today originate from that time. A glimpse on the developments that were made in the course

of those years can be found in a review[12] about Brian Spalding, a CFD pioneer, who was the head of a CFD group at Imperial College London back than. Segregated solvers, but also coupled solvers for incompressible flows have been developed and investigated there at the same time. Along with L.S. Caretto, who was working as a visiting researcher at IC at that time, Spalding published a paper that outlined the SIVA (simultaneous variable arrangement) algorithm [13], which was a pressure-based coupled solver for primitive variables. However shortly after, in the very same year, a young PhD student, named Suhas Patankar, joined the group, and came up with a segregated solution procedure that would impact CFD algorithms for decades, namely the SIMPLE algorithm (Semi Implicit Method for Pressure Linked Equations) [14]. The algorithm gave accurate results in a reasonable amount of time, without using a lot of memory space. This was clearly a huge advantage in regards to coupled algorithms that needed considerably more memory space. This fact probably also led to the premature abandonment of the development of block coupled solvers at Imperial College. Although the superiority of block coupled algorithms for strongly coupled flows was shown e.g. by Vanka[15], the SIMPLE algorithm kept dominating CFD until approximately the year 2000, when memory space became less of an issue. Today, SIMPLE and its derivatives can still be found in almost every commercial CFD software package, although it is being more and more replaced. When memory space became less of an issue and with increasing knowledge about the numerical solution of non-linear coupled systems of equations, some groups of researchers started to investigate again the simultaneous solution of the Navier-Stokes equations. Thereby Galpin and Raithy [16], Deng et al. [17] and Vanka [15] made a major step forward, which led to the introduction of block coupled algorithms into commercial software packages at the end of the 20th century. The algorithm that is outlined in this thesis leans on the scientific findings of mentioned researchers, transferring them to another block coupled discretisation framework proposed by Darwish [18].

4.2 Sequential Solution

Whenever the equations' inter-variable coupling is loose and when the system of equations is non-linear it might be preferable to treat each of the equations separately leaving only a single variable as unknown and temporarily treating all other variables as known, taking previously found values for them. The sequentially obtained solution field, however, will not satisfy the set of equations, because the variables that were assumed to be known, had been substituted by guessed approximate values. For this reason these guessed values have to be updated after each iteration in order to obtain a final solution field that satisfies the constituting equations, which makes outer iteration cycles necessary.

For the solution of the Navier-Stokes equations the most prominent of these solvers is the 'Semi-Implicit Method for Pressure Linked Equations' (SIMPLE) by Patankar [14]. In this sequential (segregated) algorithm the momentum equations are solved implicitly for each velocity component at a time, holding the remaining velocity components and the pressure constant and treating them explicitly. This leads to solution fields for velocities and pressure that do not satisfy the continuity equation. In order to assure mass conservation a pressure correction equation has to be solved. With the obtained pressure correction field, the pressure as well as the velocity component are updated, and a mass conserving solution is obtained. However, now the momentum equations are not satisfied any more, but the newly obtained velocities represent a better guess for the temporarily constant variables of the momentum equations. In this manner outer iterations have to be performed with the objective of updating non-linearities and solution fields until solution fields are obtained which satisfy the whole set of equations. Since this method projects a solution arising from solving the momentum equations, which subsequently has to be corrected, the SIMPLE algorithm can be seen as a projection method.

A variety of SIMPLE-based projection methods exists, which can be seen as enhancements of the original approach. The main concept, however, remains unchanged. The most prominent variations of SIMPLE are the SIMPLER (SIMPLE revised) [19] and the SIMPLEC (SIMPLE corrected/consistent) [20] algorithms. Now the derivation and solution procedure of the SIMPLE algorithm based on Ferzinger [21] shall be outlined. Ferzinger therein denotes intermediate

(outer iteration) steps with m. By means of sub-looping the intermediate solution fields at iteration m will approach the final iteration value at time $n+1$ which satisfies the governing equations.

At time $n+1$ the following discretised momentum equation shall be fulfilled:

$$a_C^{u_i} u_{i,C}^{n+1} + \sum_{f_{int}} a_{nei}^{u_i} u_{i,nei}^{n+1} = Q_{u,i}^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i}\right)_C \tag{4.1}$$

Implicitly discretising the momentum equations, treating only a single velocity component in each equation as unknown, and setting the remaining variables constant yields for a control volume C following discretisation for the momentum equations:

$$a_C^{u_i} u_{i,C}^{m^*} + \sum_{f_{int}} a_{nei}^{u_i} u_{i,nei}^{m^*} = Q_{u,i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_C \tag{4.2}$$

Therein the source term $Q_{u,i}^{m-1}$ contains the discretised terms of the velocity components which are treated as constants at the intermediate outer iteration step m. The solution (at m^*) obtained by equation(4.2) does not satisfy the continuity equation and therefore the corrections have to be added to the used variables $\mathbf{u}^{m^*}$ and p^{m-1}.

$$u_i^m = u_i^{m^*} + u_i' \qquad p^m = p^{m-1} + p' \tag{4.3}$$

Injecting the correction equations(4.3) into equation(4.2) yields the following equation:

$$a_C^{u_i} \left(u_{i,C}^m - u_{i,C}'\right) + \sum_{f_{int}} a_{nei}^{u_i} \left(u_{i,nei}^m - u_{i,nei}'\right) = Q_{u,i}^{m-1} - \left(\frac{\delta p^m}{\delta x_i}\right)_C + \left(\frac{\delta p'}{\delta x_i}\right)_C \tag{4.4}$$

This can be reordered to give:

$$a_C^{u_i} u'_{i,C} + \sum_{f_{int}} a_{nei}^{u_i} u'_{i,nei} + \left(\frac{\delta p'}{\delta x_i}\right)_C = \underbrace{a_C^{u_i} u_{i,C}^m + \sum_{f_{int}} a_{nei}^{u_i} u_{i,nei}^m + \left(\frac{\delta p^m}{\delta x_i}\right)_C - Q_{u,i}^{m-1}}_{\to 0} \tag{4.5}$$

The terms on the RHS of equation (4.4) represent the momentum equations at the intermediate time step m. Since we are looking for a correction equation that fulfils the momentum equations the RHS should vanish. Reordering the LHS and setting the RHS to zero yields:

$$u'_{i,C} = \underbrace{-\frac{\sum_{f_{int}} a_{nei}^{u_i} u'_{i,nei}}{a_C^{u_i}}}_{\tilde{u}'_{i,C}} - \frac{1}{a_C^{u_i}} \left(\frac{\delta p'}{\delta x_i}\right)_C \tag{4.6}$$

An additional condition that has to be introduced to the correction equation is the continuity equation. At time m the velocity field shall abide the continuity equation:

$$\frac{\delta u_i^m}{\delta x_i} = \frac{\delta u_i^{m^*}}{\delta x_i} + \frac{\delta u'_i}{\delta x_i} = 0 \tag{4.7}$$

Injecting equation(4.6) into equation(4.7) yields a correction equation for the pressure.

$$\frac{\delta}{\delta x_i}\left[\frac{1}{a_C^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)\right]_C = \left[\frac{\delta u_i^{m*}}{\delta x_i}\right]_C - \underbrace{\left[\frac{\delta \tilde{u}'_i}{\delta x_i}\right]_C}_{<<} \tag{4.8}$$

Since the neighbour velocity corrections $u'_{i,nei}$ are unknown at this point and since it is difficult to obtain them, SIMPLE neglects the second term on the RHS of equation(4.8). Other algorithms like SIMPLEC and SIMPLER are less brutal and try to find an estimate of this term.

With this set of equations we can now write down a procedure to obtain a solution, which satisfies the governing equations through sub-looping.

Solution procedure:

- Assemble and calculate preliminary velocity fields $u_i^{m^*}$ (equation (4.2))

- Assemble and calculate a pressure correction field p' (equation (4.8))
- Calculate velocity correction fields u_i' (equation (4.6))
- Update preliminary velocity and pressure fields with corrections (equation (4.3))
- Go back to step one and repeat outer iterations until $u_i^{m-1} \approx u_i^m$ and $p^{m-1} \approx p^m$
- If $u_i^{m-1} \approx u_i^m$ and $p^{m-1} \approx p^m$: assemble and calculate turbulence equations
- Update properties and volume flux and go to next time step $n+1$

Figure 4.1 shows a conceptual scheme of the SIMPLE solution procedure including under-relaxation factors and turbulence model.

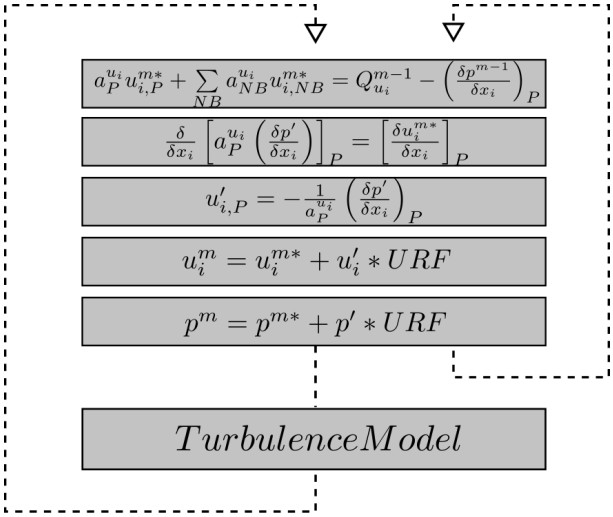

Figure 4.1: Concept of SIMPLE solution procedure

When the inter-variable coupling is important, the loose coupling between variables can be very detrimental to convergence rate and stability. For this reason only fractions of the correction fields are added to the preliminary solution fields, while carrying out the sub-loops to assure stable convergence. However, the optimal magnitudes for the necessary under-relaxation factors have to be found by testing since they vary depending on the case treated.

Additionally it is beneficial to implicitly under-relax the intermediate momentum equations(4.2).

Following Ferzinger [21] and Patankar [19] implicit relaxation for the intermediate momentum equations(4.2) yields:

$$\frac{a_C^{u_i}}{\alpha} u_{i,C}^{m^*} + \sum_{f_{int}} a_{nei}^{u_i} u_{i,nei}^{m^*} = Q_{u,i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_C + \frac{1-\alpha}{\alpha} a_C^{u_i} u_{i,C}^{m-1} \tag{4.9}$$

In respect to equation (4.2) this signifies that a term has been added that is somewhat similar to a time derivative term of momentum.

$$a_C^{u_i} \frac{1-\alpha}{\alpha} \left(u_{i,C}^{m^*} - u_{i,C}^{m-1}\right) \tag{4.10}$$

An thought time derivation of momentum would give:

$$\frac{\delta V_C}{\delta t} \left(u_{i,C}^{m^*} - u_{i,C}^{m-1}\right) \tag{4.11}$$

A plausibility analysis juxtaposing equations (4.10,4.11),and assuming that α is constant, we obtain:

$$\frac{\delta V_C}{\delta t} \propto a_C^{u_i} \tag{4.12}$$

If $a_C^{u_i}$ is dominated by convection we can further write:

$$\frac{\delta V_C}{\delta t} \propto \delta S_C \tag{4.13}$$

So implicit under-relaxation can be seen as a time derivative with an imaginary time step δt of:

$$\delta t \propto \frac{\delta V_C}{\delta S_C} \propto \delta x_C \tag{4.14}$$

Thereof implicit under-relaxation can be interpreted as an imaginary time derivative in which different time steps are used for different nodes. Furthermore it can be seen that with increasing mesh sizes and decreasing grid spacing the imaginary time step also decreases. This also means that for a given iteration the fluid flow information propagates very slowly depending on the mesh spacing δx_C. Furthermore this signifies that a simulation on a fine mesh will need more iterations than the same simulation on a coarse mesh, using the same under-relaxation factors. For this reason sequential algorithms, which have to use implicit relaxation, always have the inherent shortcoming of bad mesh size scaling in terms of convergence time per control volume.

4.3 Coupled Algorithms

When the inter-variable coupling of a set of equations is tight, the natural way of treating these equations is solving them simultaneously, all at the same time. The big advantage of this approach is that a variable at a specific point in the domain directly feels the influence of all variables. When dealing with a set of linear equations nobody would think of any other solution procedure then a coupled one. However, when treating sets of non-linear equations, the question arises whether a sequential solution procedure wouldn't be faster, since the linearised set of non-linear equations has to be updated several times and solving a huge set of linearised equations simultaneously multiple times could be slower than including the non-linear updating into a sequential solution process. For tightly coupled problems, it seems that solving linearised sets of non-linear equations simultaneously is beneficial.

As already mentioned in section 4.1 the main proponents of coupled solutions for the Navier-Stokes equations were Galpin and Raithy [16], Deng et al. [17] and Vanka [15]. The coupled solution approach elaborated in the scope of this thesis leans on the method for unstructured, collocated grids originally proposed by Darwish [18] for two-dimensional flows.

The discretisation and solution procedure of a set of algorithms that have been elaborated, will be outlined in the following sections.

A problem that all proposed coupled algorithms have in common is the saddle point problem [22], which arises from the non-existence of the pressure in the continuity equation. The absence of the pressure in this equation leads to zero entry in the diagonal of the discretised matrix system. A cure for this condition was proposed by Rhie and Chow [23].

A generic simultaneous solution procedure for the incompressible Navier-Stokes equations is given below, and a conceptual scheme of the simultaneous procedure presented in this work is shown in Figure 4.2.

Solution procedure (for steady-state calculation):

- Assemble and calculate governing equations simultaneously
- Assemble and calculate turbulence equations
- Update properties and volume flux and go to next iteration $n+1$

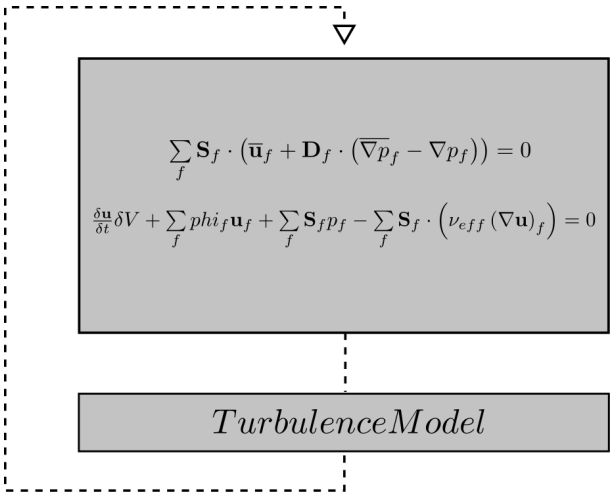

Figure 4.2: Concept of block coupled solution procedure

At this point a more detailed description shall be given additionally, because the block coupled pressure based incompressible algorithm outlined here shall be the model for subsequent descretisation (chapters 5,6,7).

For steady state simulations it is sufficient to update non-linearities and preliminary field values within the same loop. The resulting procedure is given in Figure 4.3.

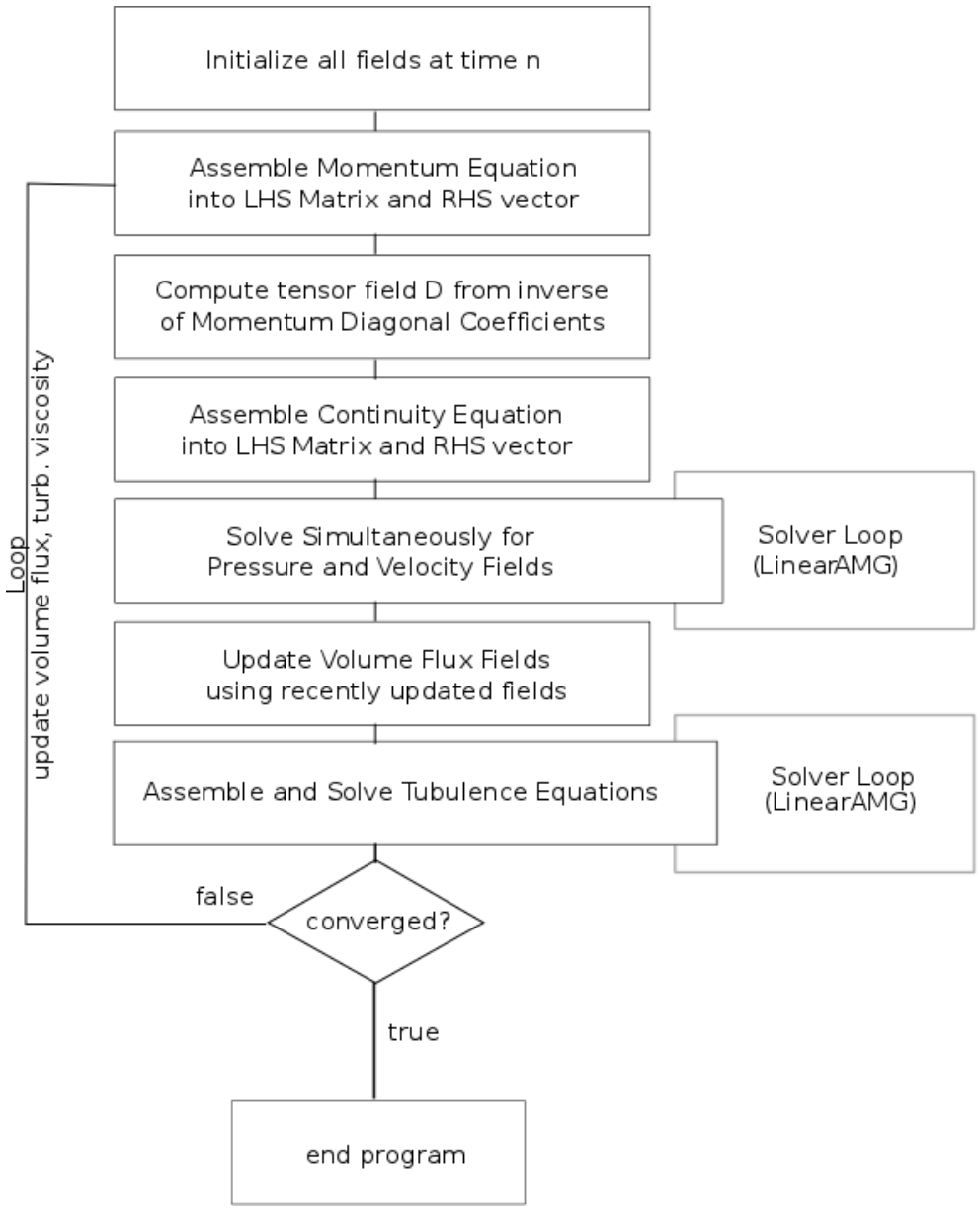

Figure 4.3: Concept of steady state block coupled solution procedure

For transient simulations an additional inner loop is needed to make sure that for each time step non-linearities and non-orthogonal correction parts have converged sufficiently. Figure 4.4 shows such a procedure.

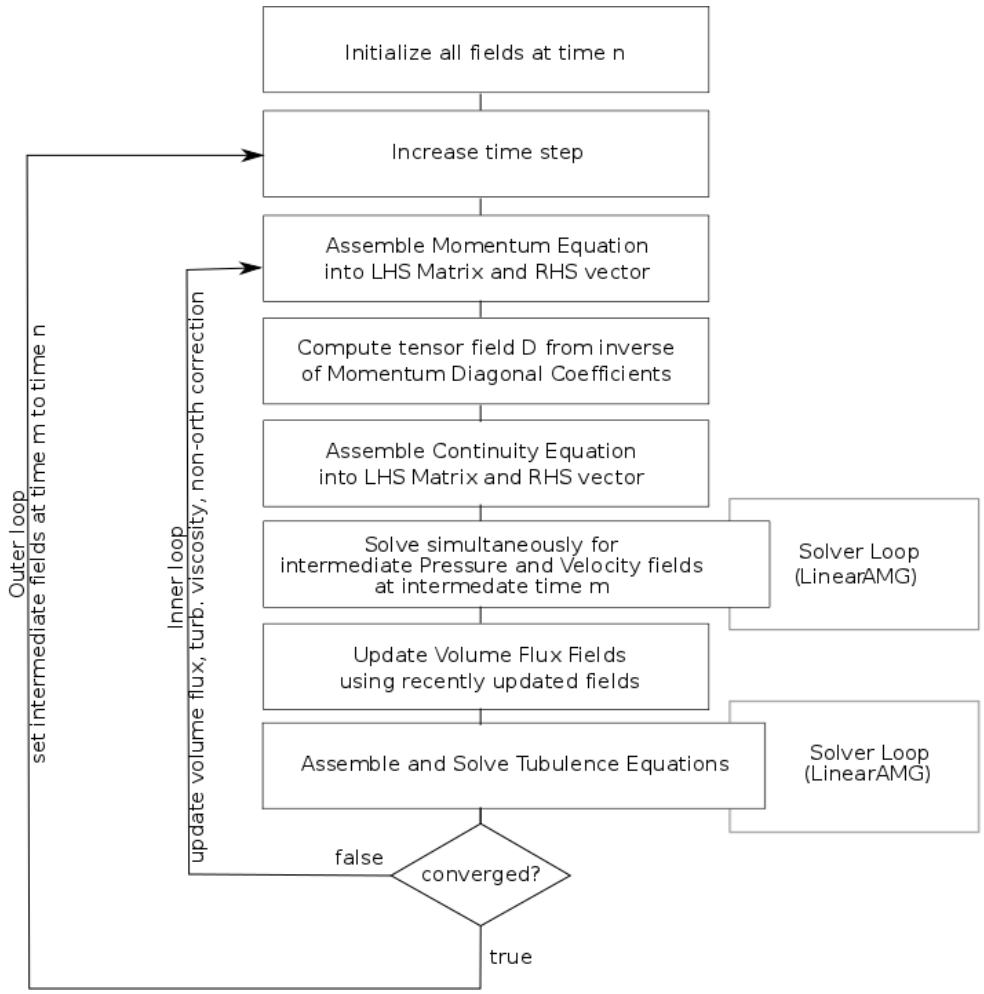

Figure 4.4: Concept of transient block coupled solution procedure

4.4 Sequential vs. Simultaneous Solution

The difference between loose and tight coupling is determinant for the properties defining simultaneous and sequential algorithms. It can be basically said that simultaneous solution procedures deduce their advantages compared to sequential solution algorithms from the tight inter-variable coupling. Their disadvantages arise due to their block coupled matrix structure, which tends to make the solution procedure more challenging. Starting with the benefits obtained by block coupling it can be said that robustness and convergence speed naturally tend to be better when using block coupled algorithms. This is especially significant when solutions are sought starting with a very bad initial guess. In these cases segregated solvers suffer a lot because the error introduced by the initial solution is carried along with the iterations, and the smoothing process until a reasonable guess is found can take very long. CFD developers often argue that this advantage becomes obsolete when dealing with transient simulations, because the solution fields at the previous time step are very close to the fields for the next time step. This, however, only holds true when no abrupt transient change such as pressure and velocity fluctuations occur. Furthermore the simultaneous solution procedure's robustness is superior because the tight inter-variable coupling hinders the variables to start oscillating. To overcome the issue of robustness sequential solution algorithms commonly have to introduce under-relaxation. This under-relaxation however leads to convergence rates which are mesh size dependent. Since simultaneous solution procedures do not need to introduce under-relaxation they show mesh size independent convergence rates. The shortcomings of simultaneous solutions arising from the block coupled matrix solution system are related to the solution of the linear equation system itself. Due to the solution of all variables in the same matrix system, the memory requirement is higher than that of sequential solution methods. Additionally the block matrix' condition number is much worse than those of the sequential method's matrices. This is because of the occurrence of additional implicit extra-diagonal coefficients that decrease the diagonal dominance of the block matrix. It can even happen that the block coupled system of equations is not diagonally dominant any more. Due to this property only very special linear solvers can be used for block matrix systems, whereas for the solution of sequential algorithms'

matrix systems usually almost any iterative solver can be used. Figure 4.5 shows a comparison between simultaneous and sequential solution methods.

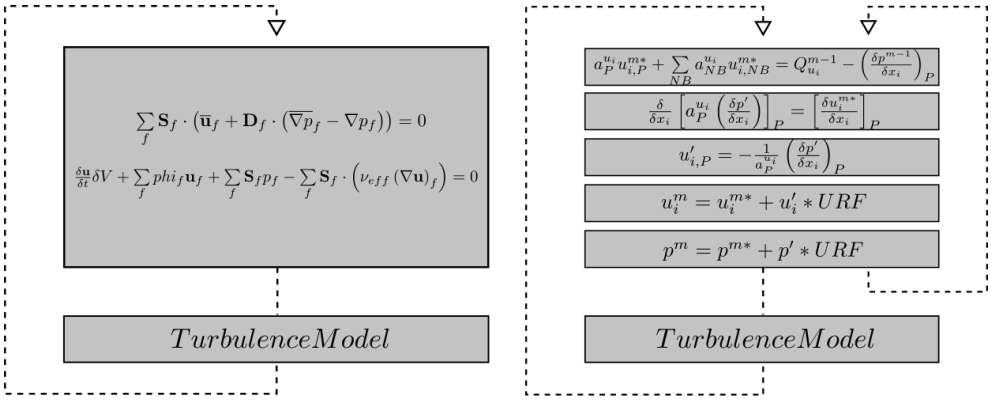

Figure 4.5: Concept comparison between simultaneous and sequential algorithm

Chapter 5

Discretisation Of Governing Equations

Contents

In the previous chapter (4) the concept of a block-coupled pressure-based incompressible algorithm was presented. According to this concept, the discretisation of all terms of the governing equations for three different reference frames (chapter 3) shall be outlined in what follows. For this purpose the addressing concept for unstructured meshes, which is the basis for discretisation of the OpenFOAM CFD library, is also presented in some detail.

5.1 Finite Volume Discretisation in OpenFOAM

Finite volume discretisation in OpenFOAM follows the arbitrarily unstructured collocated grid approach. This means that all variables are solved at points at the cell centres. Moreover, these points are located at the centroids of each cell [1], such that:

$$\int_V (\mathbf{x} - \mathbf{x}_C)\, dV = 0 \tag{5.1}$$

The mean value of the values at each point inside a control volume, is projected to the centroid of the volume. This approximation is commonly known as 'Mean Value Theorem', and it is second order accurate as can be derived with following Taylor series expansion.

$$\begin{aligned} &\phi_C = \frac{1}{V_C}\int_V (\phi)\, dV \\ &\frac{1}{V_C}\int_V \left[\phi_C + (\mathbf{x} - \mathbf{x}_C)\cdot\nabla(\phi)_C + O(\Delta x^2)\right] dV \\ &\phi_C \frac{1}{V_C}\int_V dV + \frac{1}{V_C}\int_V (\mathbf{x} - \mathbf{x}_C)\, dV \cdot (\nabla\phi)_C + \frac{1}{V_C}\int_V O(\Delta x^2) dV \end{aligned} \tag{5.2}$$

If ϕ_C is evaluated at the centroid, the second term on the RHS vanishes.

$$\phi_C \approx \phi_C + \frac{1}{V_C}\int_V O(\Delta x^2) dV \tag{5.3}$$

Figure 5.1 shows the general arrangement of the grid structure, as it can be found in OpenFOAM. In order to diminish memory space OpenFOAM's grid structure consists of owner cells (centroid $\mathbf{x}_{own}$) and neighbour cells (centroid $\mathbf{x}_{nei}$), which share a common surface. Each surface has a surface normal vector pointing from the owner to the neighbour. Cells may also have boundary faces (face center $\mathbf{x}_{bou}$). Notations, as employed in OpenFOAM can be seen in Figure 5.1. They shall subsequently be used to derive and analyse the discretised terms of the

governing equations.

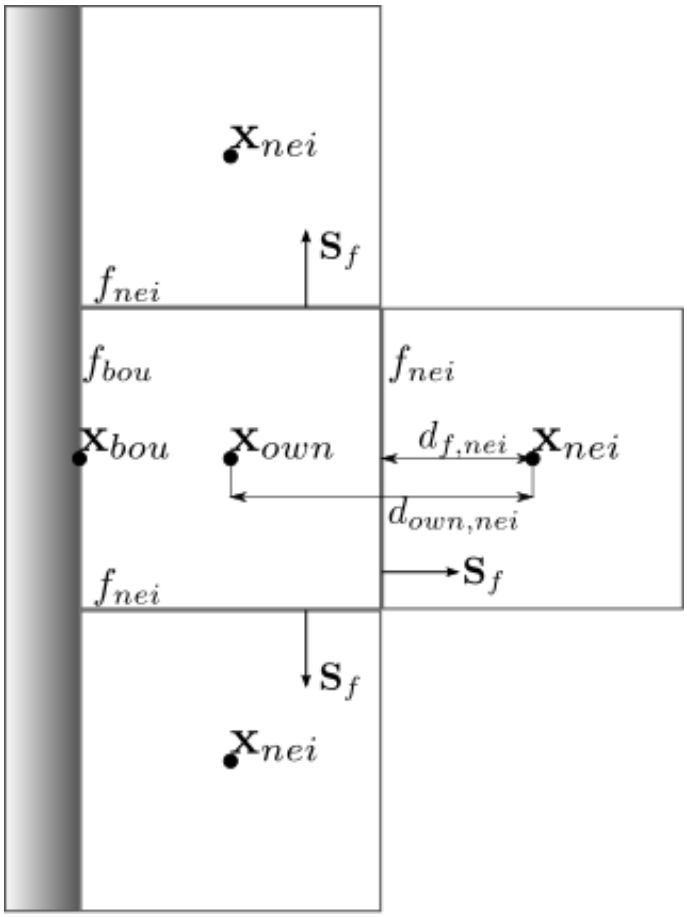

Figure 5.1: Collocated grid arrangement used in OpenFOAM

The basic mesh structure in OpenFOAM is computationally stored in 4 lists. This structure has been chosen in order to minimize memory storage.

- A **point list**: A list of point coordinates
- A **face list**: A list of point indices, corresponding to the point list
- An **owner list**: A list of cells owning a face, where the list's index corresponds to the owned face and the list's value corresponds to the index of the owning cell. The owner list does not only contain information about internal faces, but it contains also information about boundary faces. Hence at the beginning the list's index corresponds to internal faces. After that the index corresponds to the label of the respective boundary face.

- A **neighbour list**: A list of cells neighbouring faces without being their owner, where the list's index corresponds to the neighboured face and the list's value corresponds to the label of the neighbouring cell.

Figure 5.2 shows an example of how this list addressing works.

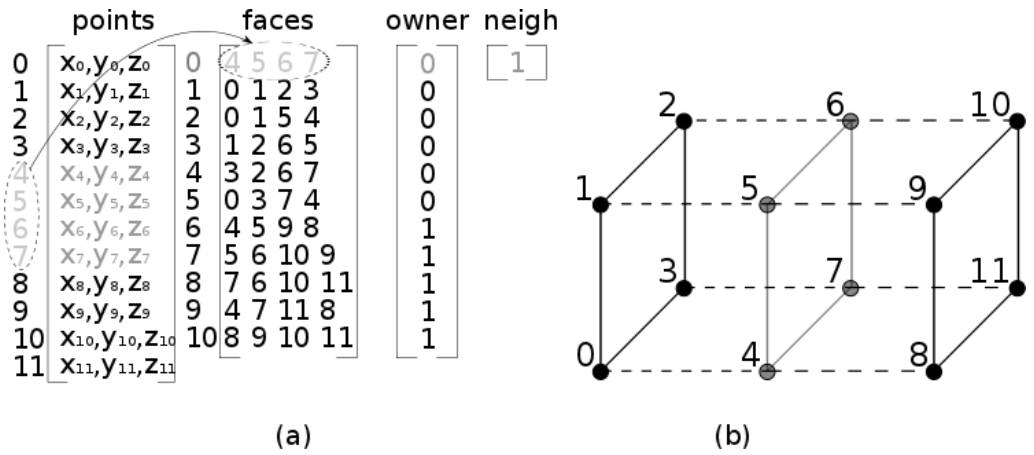

Figure 5.2: (a) list adressing structure; (b) corresponding 2 cell domain

This means that each face is shared by two cells. Information and values at each face are however only stored once to avoid duplicating information. However it becomes necessary to differentiate between the two cells sharing a common face, introducing the notion of owner and neighbour cells. The reason for this differentiation become more apparent having a look at the following examples.

It is important to know that surface normal vectors S_f always point from a face's owner cell to its neighbour cell. Furthermore surface related values , such as the linear distance weighting factor g_f (see equation (5.14)), are stored only once per face. Hence it is important to know, who the owner (factor g_f) and the neighbour (factor $1 - g_f$) cell of the face are. Figure 5.3 shall help to explain how mesh information is stored in OpenFOAM. For each face, the adjacent cell with the lower index is also the owner of the face.

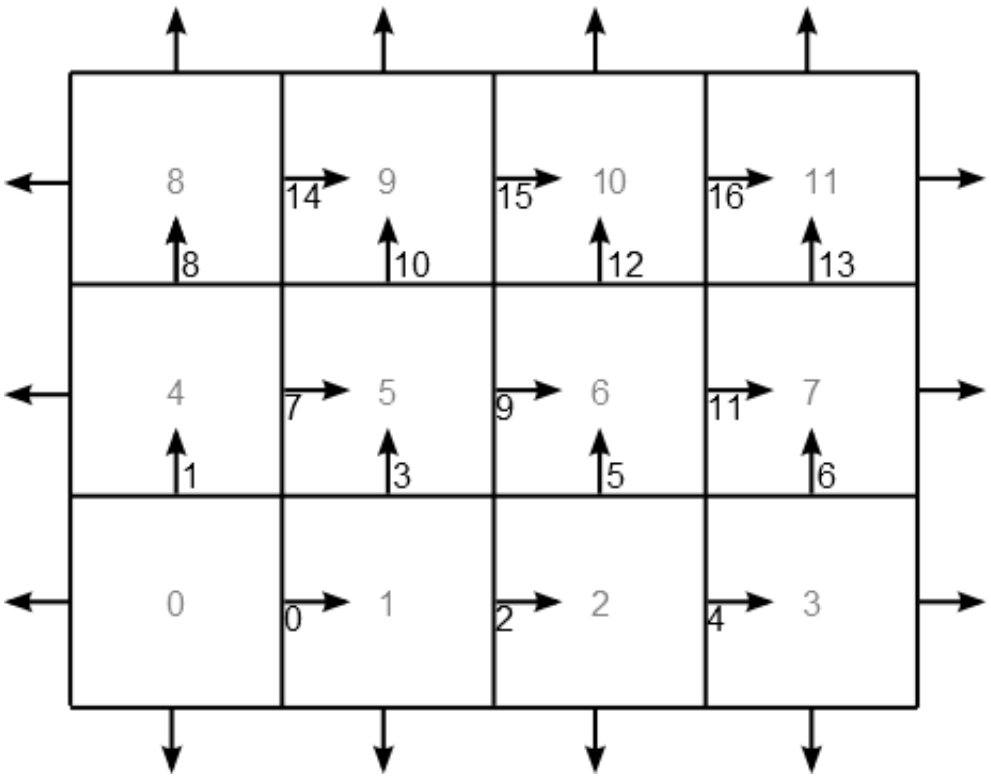

Figure 5.3: Mesh arrangement, surface normal vector pointing from owner to neighbour cell

One has to be aware, that the use of arbitrarily unstructured meshes, as nice as it may be from a user's perspective, brings some disadvantages in terms of discretisation. Because of memory restrictions, a cell object in OpenFOAM only knows its first neighbours, but not its second (the neighbours' neighbours) or even third neighbours.

This restriction is a big disadvantage compared to structured hexahedral grids, where due to the mesh structure, information on cell indices can easily be extracted. For instance when an integration over a cell surface is carried out in OpenFOAM, first a face value is calculated at the surface center, which depends only on the two adjacent cells. This procedure can have a considerably inferior accuracy compared to an integration where cell values of far neighbours are also extrapolated to the face edges. Keep in mind that the small computational molecule, that usually comes along with an arbitrarily unstructured collocated grid approach has some shortcomings that cannot be circumvented without deteriorating memory usage and computational speed.

5.2 Discretisation of Governing Equations for Inertial Frames

The aim of this section is to identify all matrix coefficients and source terms that determine the discretised and linearized system of equations that shall be solved. Since we want to solve the equations for conservation of mass and momentum simultaneously the resulting matrix will be a matrix of submatrices (matrix coefficients) containing the following entries.

$$\begin{pmatrix} a_{own}^{pp} & a_{own}^{pu} & a_{own}^{pv} & a_{own}^{pw} \\ a_{own}^{up} & a_{own}^{uu} & a_{own}^{uv} & a_{own}^{uw} \\ a_{own}^{vp} & a_{own}^{vu} & a_{own}^{vv} & a_{own}^{vw} \\ a_{own}^{wp} & a_{own}^{wu} & a_{own}^{wv} & a_{own}^{ww} \end{pmatrix} \begin{pmatrix} p_{own} \\ u_{own} \\ v_{own} \\ w_{own} \end{pmatrix} + \sum_{f_{int}} \begin{pmatrix} a_{nei}^{pp} & a_{nei}^{pu} & a_{nei}^{pv} & a_{nei}^{pw} \\ a_{nei}^{up} & a_{nei}^{uu} & a_{nei}^{uv} & a_{nei}^{uw} \\ a_{nei}^{vp} & a_{nei}^{vu} & a_{nei}^{vv} & a_{nei}^{vw} \\ a_{nei}^{wp} & a_{nei}^{wu} & a_{nei}^{wv} & a_{nei}^{ww} \end{pmatrix} \begin{pmatrix} p_{nei} \\ u_{nei} \\ v_{nei} \\ w_{nei} \end{pmatrix}$$
$$+ \sum_{f_{bou}} \begin{pmatrix} a_{bou}^{pp} & a_{bou}^{pu} & a_{bou}^{pv} & a_{bou}^{pw} \\ a_{bou}^{up} & a_{bou}^{uu} & a_{bou}^{uv} & a_{bou}^{uw} \\ a_{bou}^{vp} & a_{bou}^{vu} & a_{bou}^{vv} & a_{bou}^{vw} \\ a_{bou}^{wp} & a_{bou}^{wu} & a_{bou}^{wv} & a_{bou}^{ww} \end{pmatrix} \begin{pmatrix} p_{own} \\ u_{own} \\ v_{own} \\ w_{own} \end{pmatrix} = \begin{pmatrix} b_{own}^{p} \\ b_{own}^{u} \\ b_{own}^{v} \\ b_{own}^{w} \end{pmatrix} + \begin{pmatrix} b_{bou}^{p} \\ b_{bou}^{u} \\ b_{bou}^{v} \\ b_{bou}^{w} \end{pmatrix} \tag{5.4}$$

This leads to a sparse linear system of equations of the form:

$$\mathbf{A} \cdot \Phi = \mathbf{B} \tag{5.5}$$

The matrix consists of matrix coefficients **a**, and it is sparse having as many off-diagonal entries

per row as the owner cell has neighbours.

$$\begin{pmatrix} \left(\mathbf{a}_{11}\right) & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \left(\mathbf{a}_{nei}\right) & \cdot & \left(\mathbf{a}_{own,bou}\right) & \left(\mathbf{a}_{nei}\right) & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \left(\mathbf{a}_{nn}\right) \end{pmatrix} \cdot \begin{pmatrix} \left(\phi_1\right) \\ \cdot \\ \left(\phi_{own}\right) \\ \cdot \\ \left(\phi_n\right) \end{pmatrix} \cdot = \begin{pmatrix} \left(\mathbf{b}_1\right) \\ \cdot \\ \left(\mathbf{b}_{own,bou}\right) \\ \cdot \\ \left(\mathbf{b}_n\right) \end{pmatrix} \tag{5.6}$$

The boundary conditions will mostly give contributions to the diagonal block matrix coefficients $\mathbf{a}_{ii}$ or to the source vectors $\mathbf{b}_i$ or both. However, there are some boundary conditions that are less easy to implement, since they yield off-diagonal coefficients. In the following sections we will treat only the derivation of coefficients arising from a cell's neighbouring cells. Boundary conditions will be treated in detail in chapter 6.

5.2.1 Discretisation of Momentum Equation

The momentum Equation in integral form can be written as:

$$\int_V \frac{\partial \mathbf{u}}{\partial t} \mathrm{d}V + \int_V \nabla \cdot (\mathbf{u}\mathbf{u})\, \mathrm{d}V = -\int_V \nabla p \mathrm{d}V + \int_V \nabla \cdot (\nu_{eff} \nabla \mathbf{u})\, \mathrm{d}V \tag{5.7}$$

Note that in OpenFOAM the pressure variable actually corresponds to a pressure divided by a constant density.

Gauss' Theorem leads to:

$$\int_V \frac{\partial \mathbf{u}}{\partial t} \mathrm{d}V + \oint_S \mathbf{n} \cdot (\mathbf{u}\mathbf{u})_f \, \mathrm{dS} = -\oint_S \mathbf{n} p_f \, \mathrm{dS} + \oint_S \mathbf{n} \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}\right)_f\right) \mathrm{dS} \tag{5.8}$$

Semi-discretised form:

$$\frac{\delta \mathbf{u}}{\delta t}\delta V + \sum_{faces} \mathbf{S}_f \cdot (\mathbf{uu})_f = -\sum_{faces} \mathbf{S}_f p_f + \sum_{faces} \mathbf{S}_f \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}\right)_f\right) \tag{5.9}$$

Introducing the volume face flux $phi_f = \mathbf{S}_f \cdot \mathbf{u}_f$ this equation can be transformed to:

$$\frac{\delta \mathbf{u}}{\delta t}\delta V + \sum_{faces} phi_f \mathbf{u}_f + \sum_{faces} \mathbf{S}_f p_f - \sum_{faces} \mathbf{S}_f \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}\right)_f\right) = 0 \tag{5.10}$$

And it can be written in terms of specific forces.

$$\frac{\delta \mathbf{u}}{\delta t}\delta V + \sum_{faces} \mathbf{F}_f^C + \sum_{faces} \mathbf{F}_f^P - \sum_{faces} \mathbf{F}_f^D = 0 \tag{5.11}$$

Time discretisation

For transient simulations the discretisation in time should be of the same order as the spatial discretisation. The first time discretisation scheme that comes into mind when thinking of second order time discretisation is usually the Cranck-Nicholson scheme. Using the Cranck-Nicholson scheme implies that spatial discretisation has to be carried out implicitly for the current time step as well as explicitly for the former time step. This circumstance, however, imposes a challenge, since the dual discretisation also has to be accounted for at boundaries, at interfaces, during the flux update and when performing non-linear corrections. In order to keep a simple coding structure, it is therefore preferable to use the backward differencing formulae (BDF) for time discretisation, since the spatial discretisation has to be performed only for the current time step.

The easiest backward differencing formula (BDF1) is the Euler backward method (equation 5.12). However, for transient simulations where temporal accuracy is important, the BDF2 method (equation 5.13) has been used in this work. This method is second order accurate, and can be seen as a trade-off between accuracy and simplicity of coding.

In any case, for both approaches the non-linear coefficients and non-orthogonal terms, that arise from the second order spatial derivatives, have to be updated through sub-looping.

For steady state calculations the time derivative of the momentum equations vanishes. However, to accelerate convergence a so-called false transient term can be added. For the false transient term usually the backward differencing formula (BDF1) is used. For steady state computations the sub-looping for the non-linear components and non-orthogonal terms is not necessary. At convergence the solution field at time n+1 is equal to that of n. Hence, the contribution of the false transient term cancels out, and the non-linear and non-orthogonal coefficients take a constant value.

The first order implicit backward Euler time scheme(BDF1) reads:

Ddt term:

$$\frac{\delta \mathbf{u}}{\delta t}\delta V + \mathbf{f}(t) \approx \frac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{\Delta t}\Delta V + \mathbf{f}^{n+1} \tag{5.12}$$

The BDF2 formula for time discretisation reads, **Second Order Ddt term:**

$$\frac{\delta \mathbf{u}}{\delta t}\delta V + \mathbf{f}(t) \approx \frac{\frac{3}{2}\mathbf{u}^{n+1} - 2\mathbf{u}^{n} + \frac{1}{2}\mathbf{u}^{n-1}}{\Delta t}\Delta V + \mathbf{f}^{n+1} \tag{5.13}$$

Since the effective dynamic viscosity depends on the turbulence variables, it is not known at time (n+1). Therefore it also has to be updated through sub-looping for transient simulations.

Spatial discretisation

Discretisation in space makes use of Gauss' divergence theorem and some kind of surface interpolation. Due to OpenFOAM's mesh addressing method implicit surface interpolation only depends on the surfaces' adjacent cells.

Keeping in mind that either the Second-Order-Backward or the first order Euler time scheme is used, spatial discretisation of all terms only shall be carried out implicitly for time (n+1). Due to the explicit non-orthogonal contribution for second order derivatives both, implicit and explicit contributions, will be obtained.

All spatial discretisation schemes use some common interpolation factors, these factors shall be outlined next.

The linear interpolation weighting factors at surfaces g_f in OpenFOAM can be expressed by means of bow length evaluation:

$$g_f = \frac{mag\left[\mathbf{S}_f \cdot (\mathbf{x}_{nei} - \mathbf{x}_f)\right]}{mag\left[\mathbf{S}_f \cdot (\mathbf{x}_f - \mathbf{x}_{own})\right] + mag\left[\mathbf{S}_f \cdot (\mathbf{x}_{nei} - \mathbf{x}_f)\right]} \tag{5.14}$$

The distance vectors $\mathbf{d}$ are used to evaluate gradients at surfaces. The vectors are pointing from owner centroids to neighbour centroids.

$$\mathbf{d} = \mathbf{x}_{nei} - \mathbf{x}_{own} \tag{5.15}$$

Pressure gradient term:

The pressure gradient is discretised using Gauss' Theorem and a linear interpolation between two adjacent cells. This method is second order accurate in space.

The semi-discretised pressure gradient term reads,

$$\sum_{faces} \mathbf{S}_f p_f \tag{5.16}$$

And the pressure force over a surface is:

$$\mathbf{F}_f^P = \mathbf{S}_f p_f \tag{5.17}$$

The pressure at face f between two cells is,

$$p_f = p_{own}\, g_f + p_{nei}\, (1 - g_f) \tag{5.18}$$

An implicit contribution in form of matrix coefficients for each cell is finally obtained.

$$\begin{aligned}
a_{own}^{up} &= \sum_{f_{int}} S_{f_x}\, g_f \qquad a_{nei}^{up} = S_{f_x}\, (1 - g_f) \\
a_{own}^{vp} &= \sum_{f_{int}} S_{f_y}\, g_f \qquad a_{nei}^{vp} = S_{f_y}\, (1 - g_f) \\
a_{own}^{wp} &= \sum_{f_{int}} S_{f_z}\, g_f \qquad a_{nei}^{wp} = S_{f_z}\, (1 - g_f)
\end{aligned} \tag{5.19}$$

Convective term:

It is well known in CFD that convection schemes of order greater than one in general do not fulfill the convection boundedness criterion (CBC). In order to obtain second order accuracy to some extent it is necessary to somehow bound second order schemes. Usually this is done by means of limiters, which amongst others depend on the far upwind cells. As OpenFOAM operates on unstructured meshes these far upwind cells may not be unique. Therefore some high order schemes (NVD/TVD) as they are implemented in OpenFOAM are outlined in appendix (A).

For the present work the author decided to go for a deferred corrected van Leer TVD scheme (vanLeerVDC) as his convection scheme of choice. It is not derived at this point for the sake of brevity. The reader is referred to appendix (A) for a complete derivation of the scheme. At this point, only the resulting coefficients shall be presented.

Recall that the convective term (5.21) of the governing equations in semi-discretised form is given by:

$$\sum_{faces} phi_f \mathbf{u}_f \tag{5.20}$$

And the specific convective force over a surface is:

$$\mathbf{F}_f^C = phi_f \mathbf{u}_f \tag{5.21}$$

Where the volume flux phi_f over the cell faces is considered to be known. The linearisation of the convection term using the known volume flux from the old time step is only first order accurate in time and convergence of outer iterations could be improved using a full Newton-Raphson linearisation for the face volume flux phi_f [24]. Since a full Newton-Raphson linearisation can however bring about numerical problems, the easier first order linearisation has been implemented in the frame of this work.

In order to be second order accurate the variable vectors $\mathbf{u}_f$ at faces f for limited schemes (NVD or TVD) can be written as:

$$\begin{aligned} &case\ \ phi_f \geq 0: \\ &\mathbf{u}_f = \mathbf{u}_{own} + (1 - g_f)\,\Upsilon\,(r_f)\,(\mathbf{u}_{nei} - \mathbf{u}_{own}) \\ &case\ \ phi_f < 0: \\ &\mathbf{u}_f = \mathbf{u}_{nei} + g_f \Upsilon\,(r_f)\,(\mathbf{u}_{own} - \mathbf{u}_{nei}) \end{aligned} \tag{5.22}$$

Subsequently implicit contributions will be outlined using above approach with van Leer's TVD limiter.

Note that, unlike upwind differencing, high resolution schemes do not deliver diagonally equal matrix coefficients for the implicit part of equation (5.21), unless a deferred correction approach is chosen. Deferred correction means that the upwind part of equation (5.22) is added implicitly to the matrix, which gives diagonal equality, and the higher order contribution is added explicitly to the source term. This approach can be important to preserve diagonal and block-diagonal dominance of a system of equations, as we will see in chapter (8).

Since the volume flux phi^{n+1} is not known beforehand, it is replaced by phi^*, which has to be updated through sub-looping until $phi^* \approx phi^{n+1}$.

For a deferred corrected van Leer TVD scheme, for the implicit part of equation (5.21) at time n+1, a contribution in form of matrix coefficients and a contribution to the source vector (arising from the deferred correction) is obtained for each cell.

$$
\begin{aligned}
a^{uu}_{own} &= \sum_{f_{int}} \|phi^*_f, 0\| \qquad a^{uu}_{nei} = -\| - phi^*_f, 0\| \\
a^{vv}_{own} &= \sum_{f_{int}} \|phi^*_f, 0\| \qquad a^{vv}_{nei} = -\| - phi^*_f, 0\| \\
a^{ww}_{own} &= \sum_{f_{int}} \|phi^*_f, 0\| \qquad a^{ww}_{nei} = -\| - phi^*_f, 0\|
\end{aligned}
\tag{5.23}
$$

$$
\begin{aligned}
b^{u}_{own} &= -\sum_{f_{int}} \|phi^*_f, 0\| \left(1 - g_f\right) \Upsilon \left(u^*_{nei} - u^*_{own}\right) - \| - phi^*_f, 0\| g_f \Upsilon \left(u^*_{own} - u^*_{nei}\right) \\
b^{v}_{own} &= -\sum_{f_{int}} \|phi^*_f, 0\| \left(1 - g_f\right) \Upsilon \left(v^*_{nei} - v^*_{own}\right) - \| - phi^*_f, 0\| g_f \Upsilon \left(v^*_{own} - v^*_{nei}\right) \\
b^{w}_{own} &= -\sum_{f_{int}} \|phi^*_f, 0\| \left(1 - g_f\right) \Upsilon \left(w^*_{nei} - w^*_{own}\right) - \| - phi^*_f, 0\| g_f \Upsilon \left(w^*_{own} - w^*_{nei}\right)
\end{aligned}
\tag{5.24}
$$

Viscous term:

The diffusion term is the effective dynamic viscosity times a Laplacian of the velocity vector. It is discretised using Gauss' divergence theorem and a linear corrected interpolation. This interpolation is second order accurate in space when two adjacent cells are orthogonally aligned. The order of accuracy decreases with increasing non-orthogonality of the grid. OpenFOAM implements an over-relaxed orthogonal correction approach for Laplacian operators (such as the viscous term), which shall be briefly outlined in the following section. The resulting matrix coefficients will be diagonally equal. The following derivations can be seen in more detail with a thorough error analysis in Jasak [1] and also in Ferzinger and Perić [21] and Muzaferija [25].

The semi-discretised diffusion term reads,

$$-\sum_f \mathbf{S}_f \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}\right)_f\right) \tag{5.25}$$

And the viscous force over a surface is:

$$F_f^D = \mathbf{S}_f \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}\right)_f\right) \tag{5.26}$$

Equation (5.26) can be written as:

$$F_f^D = \nu_{eff} \left|\mathbf{S}_f\right| \begin{pmatrix} \frac{\partial u}{\partial x} n_x + \frac{\partial u}{\partial y} n_y + \frac{\partial u}{\partial z} n_z \\ \frac{\partial v}{\partial x} n_x + \frac{\partial v}{\partial y} n_y + \frac{\partial v}{\partial z} n_z \\ \frac{\partial w}{\partial x} n_x + \frac{\partial w}{\partial y} n_y + \frac{\partial w}{\partial z} n_z \end{pmatrix}_f = \nu_{eff} \left|\mathbf{S}_f\right| \left(\frac{\partial \mathbf{u}}{\partial x} n_x + \frac{\partial \mathbf{u}}{\partial y} n_y + \frac{\partial \mathbf{u}}{\partial z} n_z\right)_f \tag{5.27}$$

The far right expression on the RHS can be written in local basis vectors [21].

$$\mathbf{n} \cdot \left(\nabla \mathbf{u}\right)_f = \frac{\partial \mathbf{u}}{\partial x} n_x + \frac{\partial \mathbf{u}}{\partial y} n_y + \frac{\partial \mathbf{u}}{\partial z} n_z = \frac{\partial \mathbf{u}}{\partial r} e_r + \frac{\partial \mathbf{u}}{\partial s} e_s + \frac{\partial \mathbf{u}}{\partial t} e_t \tag{5.28}$$

Keeping in mind, that OpenFOAM can only do implicit evaluation based on the information of two adjacent points, the basis vectors of equation(5.28) are chosen such that the first basis vector is aligning with the line connecting the owners' and the neighbours' centroids. ξ is the direction of the line connecting the owner and the neighbour centroids, and direction η is defined as:

$$\eta = \mathbf{n} - \mathbf{e}_\xi \tag{5.29}$$

Hence,

$$\mathbf{n} \cdot (\nabla \mathbf{u})_f = \frac{\partial \mathbf{u}}{\partial x} n_x + \frac{\partial \mathbf{u}}{\partial y} n_y + \frac{\partial \mathbf{u}}{\partial z} n_z = \frac{\partial \mathbf{u}}{\partial \xi} e_\xi + \frac{\partial \mathbf{u}}{\partial \eta} e_\eta \tag{5.30}$$

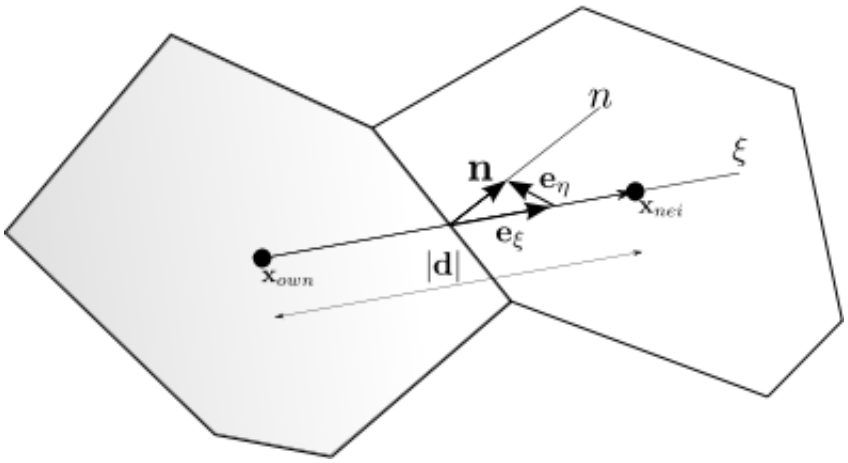

Figure 5.4: Control volumes with local coordinate directions

The first term on the RHS of equation (5.30) will be treated implicitly, whereas the second term can only be evaluated explicitly due to a limited implicit stencil. Discretisation contributions arising from second neighbours are not added implicitly. The practicle reason for this is that the addressing of second neighbours in an unstructured grid framework would be cumbersome, the numerical reason is that due to additional off-diagonal matrix elements the resulting sparse matrix could lose diagonal dominance. We shall rewrite equation (5.30) to see that the diffusive flux at time (n+1) is evaluated correctly as explicit variable values at time (n,*) approach

variable values at time (n+1).

$$\mathbf{n} \cdot \left(\nabla \mathbf{u}^{n+1}\right)_f = \frac{\partial \mathbf{u}^{n+1}}{\partial n} = \frac{\partial \mathbf{u}^{n+1}}{\partial \xi} e_\xi + \frac{\partial \mathbf{u}^{n+1}}{\partial \eta} e_\eta \approx \left(\frac{\partial \mathbf{u}^{n+1}}{\partial \xi}\right) e_\xi + \left(\frac{\partial \mathbf{u}^{*}}{\partial n} e_n - \frac{\partial \mathbf{u}^{*}}{\partial \xi} e_\xi\right) \tag{5.31}$$

Through sub-looping the explicit part with variable values at time (*) approaches values at time (n+1). Hence, when the variables for the explicit part approach values at time (n+1), equation (5.31) becomes implicit. Since the explicit part stems from the non-orthogonalities of the grid, this approach is also called non-orthogonal correction.

Now equation (5.27) can be rearranged.

$$F_f^D = \nu_{eff} \left|\mathbf{S}_f\right| \left(\frac{\partial \mathbf{u}}{\partial \xi} e_\xi + \frac{\partial \mathbf{u}}{\partial \eta} e_\eta\right) \tag{5.32}$$

$$F_f^D = \nu_{eff} \left|\mathbf{S}_f\right| \left(\frac{\partial \mathbf{u}}{\partial \xi} e_\xi + \frac{(\mathbf{n} - \mathbf{e}_\xi)}{|\mathbf{n} - \mathbf{e}_\xi|} \cdot (\nabla \mathbf{u})_f \, e_\eta\right) \tag{5.33}$$

$$F_f^D = \nu_{eff} \left|\mathbf{S}_f\right| \left(\frac{\partial \mathbf{u}}{\partial \xi} e_\xi + (\mathbf{n} - \mathbf{e}_\xi) \cdot (\nabla \mathbf{u})_f\right) \tag{5.34}$$

$$F_f^D = \nu_{eff} \left|\mathbf{S}_f\right| e_\xi \frac{\partial \mathbf{u}}{\partial \xi} + \nu_{eff} \left|\mathbf{S}_f\right| \left(\mathbf{n} - \frac{\mathbf{d}}{|\mathbf{d}|} e_\xi\right) \cdot (\nabla \mathbf{u})_f \tag{5.35}$$

Equation (5.35) shows how the diffusive fluxes can be split into two parts. The first part on the RHS can be treated implicitly, whereas the second term on the RHS is calculated explicitly such as in equation(5.43). It is also easy to see that the influence of the explicit and the implicit parts can be adjusted since the choice of the length of the basis vector $\mathbf{e}_\xi$ is arbitrary.

In the over-relaxed approach, as it is implemented in OpenFOAM, the magnitude of e_ξ is chosen in a way that causes the first term on the RHS of equation (5.35) to increase with increasing

non-orthogonality [1].

$$e_\xi = \frac{|\mathbf{d}|}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f| \tag{5.36}$$

Hence equation (5.35) can be written as:

$$F_f^D = \nu_{eff} |\mathbf{S}_f| \frac{|\mathbf{d}|}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f| \frac{\partial \mathbf{u}}{\partial \xi} + \nu_{eff} |\mathbf{S}_f| \left(\frac{\mathbf{S}_f}{|\mathbf{S}_f|} - \frac{\mathbf{d}}{|\mathbf{d}|} \frac{|\mathbf{d}|}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f| \right) \cdot (\nabla \mathbf{u})_f \tag{5.37}$$

and finally:

$$F_f^D = \nu_{eff} \frac{|\mathbf{d}|}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f|^2 \left(\frac{\mathbf{u}_{nei} - \mathbf{u}_{own}}{|\mathbf{d}|} \right) + \nu_{eff} \left(\mathbf{S}_f - \frac{\mathbf{d}}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f|^2 \right) \cdot (\nabla \mathbf{u})_f \tag{5.38}$$

Jasak [1] writes these expressions as follows:

$$F_f^D = \nu_{eff} |\mathbf{E}| \left(\frac{\mathbf{u}_{nei} - \mathbf{u}_{own}}{|\mathbf{d}|} \right) + \nu_{eff} \mathbf{T} \cdot (\nabla \mathbf{u})_f \tag{5.39}$$

where,

$$\mathbf{E} = \frac{\mathbf{d}}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f|^2 \tag{5.40}$$

and

$$\mathbf{T} = \mathbf{S}_f - \mathbf{E} \tag{5.41}$$

It can be easily seen that equation (5.40) gets singular when the angle between $\mathbf{d}$ and $\mathbf{S}_f$ is 90 degrees. Figure 5.5 shows the surface vectors $\mathbf{E}$ and $\mathbf{T}$ splitting the contribution for the implicit and explicit part of equation (5.39) for the over-relaxed, non-orthogonal correction approach.

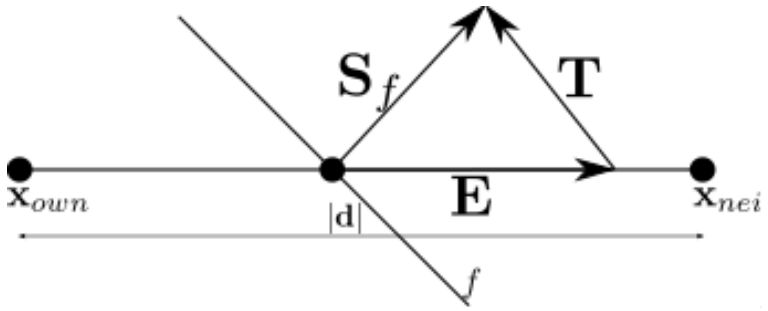

Figure 5.5: Surface normal vector split for orthogonal correction approach [1]

The first term on the RHS of equation(5.39) gives an implicit contribution and the second part is evaluated explicitly. Hereby the explicit part will be updated through sub-looping with the latest variable value (in this case $\mathbf{u}_f^*$). Calculating the explicit gradients for the second part of the RHS is done as outlined following equations (5.42,5.43, 5.44).

$$(\nabla \mathbf{u})_{cell} = \frac{1}{V_{cell}} \sum_f \mathbf{S}_f \mathbf{u}_f \tag{5.42}$$

and

$$\left(\overline{\nabla \mathbf{u}}\right)_f = g_f \left(\nabla \mathbf{u}\right)_{own} + \left(1 - g_f\right) \left(\nabla \mathbf{u}\right)_{nei} \tag{5.43}$$

The gradient of the second term on the RHS can be now evaluated explicitly. In order to be consistent the gradient calculation in direction of $\mathbf{d}$ has to be equal to the implicit term's. Hence we subtract the directional part of the interpolated gradient and add the direct surface gradient evaluation in direction of $\mathbf{d}$. This yields,

$$(\nabla \mathbf{u})_f = \left(\overline{\nabla \mathbf{u}}\right)_f - \left(\left(\overline{\nabla \mathbf{u}}\right)_f \cdot \frac{\mathbf{d}}{|\mathbf{d}|}\right) \frac{\mathbf{d}}{|\mathbf{d}|} + \frac{(\mathbf{u}_{nei} - \mathbf{u}_{own})}{|\mathbf{d}|} \frac{\mathbf{d}}{|\mathbf{d}|} \tag{5.44}$$

Jasak [1] states that non-orthogonal correction potentially causes unboundedness. To prevent oscillations and unboundedness the explicit term (non-orthogonal correction) can be limited, however to the detriment of accuracy.

$$
\begin{aligned}
a_{own}^{uu} &= \sum_{f_{int}} \nu_{eff}^{*} \cdot \frac{|E|}{|d|} \qquad a_{nei}^{uu} = -\nu_{eff}^{*} \frac{|E|}{|d|} \\
a_{own}^{vv} &= \sum_{f_{int}} \nu_{eff}^{*} \cdot \frac{|E|}{|d|} \qquad a_{nei}^{vv} = -\nu_{eff}^{*} \frac{|E|}{|d|} \\
a_{own}^{ww} &= \sum_{f_{int}} \nu_{eff}^{*} \cdot \frac{|E|}{|d|} \qquad a_{nei}^{ww} = -\nu_{eff}^{*} \frac{|E|}{|d|}
\end{aligned}
\tag{5.45}
$$

$$
\begin{aligned}
b_{own}^{u} &= \sum_{f_{int}} \nu_{eff}^{*} \left(T_x \frac{\partial u}{\partial x}^{*} + T_y \frac{\partial u}{\partial y}^{*} + T_z \frac{\partial u}{\partial y}^{*} \right) \\
b_{own}^{u} &= \sum_{f_{int}} \nu_{eff}^{*} \left(T_x \frac{\partial v}{\partial x}^{*} + T_y \frac{\partial v}{\partial y}^{*} + T_z \frac{\partial v}{\partial y}^{*} \right) \\
b_{own}^{u} &= \sum_{f_{int}} \nu_{eff}^{*} \left(T_x \frac{\partial w}{\partial x}^{*} + T_y \frac{\partial w}{\partial y}^{*} + T_z \frac{\partial w}{\partial y}^{*} \right)
\end{aligned}
\tag{5.46}
$$

5.2.2 Discretisation of Continuity Equation

The continuity equation in integral form can be written as:

$$
\int_V \nabla \cdot \mathbf{u} \, \mathrm{dV} = 0 \tag{5.47}
$$

Gauss' Theorem leads to,

$$
\oint_S (\mathbf{n} \cdot \mathbf{u}_f) \, dS = 0 \tag{5.48}
$$

Semi-discretised form,

$$
\sum_{faces} \mathbf{S}_f \cdot \mathbf{u}_f = 0 \tag{5.49}
$$

The equation can be written in terms of volume fluxes,

$$\sum_{faces} F_V = 0 \tag{5.50}$$

The non-occurrence of the pressure in the continuity equation for incompressible flows, leads to an ill conditioned matrix, since zero entries remain in its diagonal. Rhie and Chow [23] suggest an interpolation technique for the face velocities $\mathbf{u}_f$ that includes pressure gradients. The interpolation technique stems from the momentum equation, and will subsequently be laid out.

The discretised momentum equations for each cell then read:

$$\begin{aligned}
&\begin{pmatrix} a^u_{own_T} & 0 & 0 \\ 0 & a^v_{own_T} & 0 \\ 0 & 0 & a^w_{own_T} \end{pmatrix} \cdot \mathbf{u}_{own} + \begin{pmatrix} a^u_{own_{CD}} & 0 & 0 \\ 0 & a^v_{own_{CD}} & 0 \\ 0 & 0 & a^w_{own_{CD}} \end{pmatrix} \cdot \mathbf{u}_{own} \\
&+ \sum_{int} \begin{pmatrix} a^u_{nei_{CD}} & 0 & 0 \\ 0 & a^v_{nei_{CD}} & 0 \\ 0 & 0 & a^w_{nei_{CD}} \end{pmatrix} \cdot \mathbf{u}_{nei} + \sum_{bou} \begin{pmatrix} a^u_{bou_{CD}} & 0 & 0 \\ 0 & a^v_{bou_{CD}} & 0 \\ 0 & 0 & a^w_{bou_{CD}} \end{pmatrix} \cdot \mathbf{u}_{own} \\
&= -\nabla p \, \Delta V_{own} + \begin{pmatrix} b^u_{own_T} + b^u_{own_{CD}} + b^u_{bou} \\ b^v_{own_T} + b^v_{own_{CD}} + b^v_{bou} \\ b^w_{own_T} + b^w_{own_{CD}} + b^w_{bou} \end{pmatrix}
\end{aligned} \tag{5.51}$$

$$\frac{\mathbf{a}_{own}}{\Delta V_{own}} \mathbf{u}_{own} + \nabla p = \mathbf{u}_{own} - \sum_{int} \frac{\mathbf{a}_{nei}}{\Delta V_{own}} \mathbf{u}_{nei} - \sum_{bou} \frac{1}{\Delta V_{own}} \mathbf{a}_{bou} \cdot \mathbf{u}_{own} + \frac{1}{\Delta V_{own}} \mathbf{b} \tag{5.52}$$

These equations are valid for each cell of the domain, Rhie and Chow [23] propose to interpolate these equations to the surface (denoted by an overline $\overline{\cdot}$) to obtain an expression for $\mathbf{u}_f$. This is done in the following way:

$$\left(\frac{\mathbf{A}_{TCD}}{\Delta V} \right)_f \mathbf{u}_f + \nabla p_f \approx \overline{\left(\frac{\mathbf{A}_{TCD}}{\Delta V} \right)_f} \, \overline{\mathbf{u}_f} + \overline{\nabla p_f} \tag{5.53}$$

$\mathbf{A}_{TCD,f}$ being a 3x3 submatrix that contains the implicit velocity coefficients of the previously descritised momentum equations. Assuming that $\left(\frac{\mathbf{A}_{TCD}}{\Delta V}\right)_f \approx \overline{\left(\frac{\mathbf{A}_{TCD}}{\Delta V}\right)_f}$, the Rhie-Chow interpolation gives:

$$\mathbf{u}_f = \overline{\mathbf{u}_f} + \overline{\left(\frac{\mathbf{A}_{TCD}}{\Delta V}\right)_f^{-1}} \cdot \left(\overline{\nabla p_f} - \nabla p_f\right) \tag{5.54}$$

Hence the continuity equation (5.49) adapted by the Rhie-Chow interpolation reads:

$$\sum_{faces} \mathbf{S}_f \cdot \left(\overline{\mathbf{u}}_f + \overline{\mathbf{D}_f} \cdot \left(\overline{\nabla p_f} - \nabla p_f\right)\right) = 0 \tag{5.55}$$

The pressure gradient terms on the RHS introduce an artificial diffusion. Equation (5.55) remains still conservative. It can be seen that the diffusion error decreases with decreasing grid spacing. Subsequently we shall denote:

$$\overline{\mathbf{D}_f} = \overline{\left(\frac{\mathbf{A}_{TCD}}{\Delta V}\right)_f^{-1}} \tag{5.56}$$

Time discretisation

We want equation (5.55) to be valid for time (n+1). We obtain:

$$\sum_{faces} \mathbf{S}_f \cdot \left(\overline{\mathbf{u}}_f^{n+1} + \mathbf{D}_f^{n+1} \cdot \left(\overline{\nabla p_f}^{*} - \nabla p_f^{n+1,*}\right)\right) = 0 \tag{5.57}$$

The terms $\mathbf{D}_f^{n+1} \cdot \overline{\nabla p_f}^{*}$ and $\mathbf{D}_f^{n+1} \cdot \nabla p_f^{n+1,*}$ cannot or can only partially be discretised implicitly. Therefore they will be updated to approach time state (n+1) by means of sublooping. It can be seen, that theoretically the pressure gradients on the RHS of equation (5.57) should vanish as time state (*) approaches time state(n+1), hence equation(5.57) will fall back to equation(5.49). Practically this is not the case since $\mathbf{D}_f^{n+1} \cdot \overline{\nabla p_f}^{*}$ and $\mathbf{D}_f^{n+1} \cdot \nabla p_f^{n+1,*}$ are discretised differently. It follows that a small amount of diffusion will always effect the continuity equation. For the

limit of vanishing diffusion the continuity equations is second order accurate if $\mathbf{u}_f$ is obtained by central differencing.

Spatial discretisation

Again, due to OpenFOAM's mesh addressing method implicit surface interpolation will only depend on two adjacent cells.

Time (*) denotes a state of the variable that is somewhere between time state (n) and time state(n+1), starting at time (n) and approaching time state (n+1) by means of sublooping. Spatial discretisation for time (*) will be purely explicit. For time (n+1) both, implicit and explicit contributions may be obtained.

Velocity term (Divergence):

The velocity term in equation (5.49) is the remainder of the actual continuity equation for incompressible fluids, and it represents the divergence of the velocity over a closed control volume.

Term $\mathbf{S}_f \cdot \mathbf{u}_f$ is obtained by central differencing and we obtain:

$$\mathbf{S}_f \cdot \mathbf{u}_f = \mathbf{S}_f \cdot (\mathbf{u}_{own}\, g_f + \mathbf{u}_{nei}\, (1 - g_f)) \tag{5.58}$$

This yields the following matrix coefficients:

$$\begin{aligned} a_{own}^{pu} &= \sum_{f_{int}} S_{f_x}\, g_f \qquad a_{nei}^{pu} = S_{f_x}\, (1 - g_f) \\ a_{own}^{pv} &= \sum_{f_{int}} S_{f_y}\, g_f \qquad a_{nei}^{pv} = S_{f_y}\, (1 - g_f) \\ a_{own}^{pw} &= \sum_{f_{int}} S_{f_z}\, g_f \qquad a_{nei}^{pw} = S_{f_z}\, (1 - g_f) \end{aligned} \tag{5.59}$$

Numerical diffusion terms:

Numerical diffusion arises from the Rhie-Chow interpolation. The numerical diffusion tensor in equation (5.55) is anisotropic. The pressure terms in equation (5.55) correspond to two Laplacian derivatives. The first Laplacian term shall be expressed implicitly to avoid having zeros in the matrix diagonal. Otherwise the matrix would be ill-conditioned. The second Laplacian term will first be evaluated explicitly at the cell centres, before being linearly interpolated to the cell faces. The implicit Laplacian discretisation will be similar to that of the viscous term in the momentum equations (5.39) where an over-relaxed orthogonal correction approach for Laplacian operators and scalar viscosity was introduced. The non-orthogonal correction approach shall now be derived, in which the anisotropy of the diffusion tensor shall be taken into account. The derivation follows the methodology of Ferzinger and Perić [21] and will yield the coefficients outlined by Darwish [18].

Implicit numerical diffusion term (implicit Rhie-Chow term):

$$F_{RC,im} = -\mathbf{S}_f \cdot \overline{\mathbf{D}_f} \cdot \nabla p_f \tag{5.60}$$

$$\mathbf{e}_\gamma = \frac{\mathbf{S}_f \cdot \overline{\mathbf{D}_f}}{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}|} \tag{5.61}$$

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \mathbf{e}_\gamma \cdot \nabla p_f \tag{5.62}$$

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \mathbf{e}_\gamma \cdot \nabla p_f \tag{5.63}$$

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \left(\frac{\partial p}{\partial x} e_{\gamma,x} + \frac{\partial p}{\partial y} e_{\gamma,y} + \frac{\partial p}{\partial z} e_{\gamma,z} \right)_f \tag{5.64}$$

The far right expression on the RHS can be written in local basis vectors [21].

$$\mathbf{e}_\gamma \cdot \nabla p_f = \frac{\partial p}{\partial x} e_{\gamma,x} + \frac{\partial p}{\partial y} e_{\gamma,y} + \frac{\partial p}{\partial z} e_{\gamma,z} = \tag{5.65}$$

Keeping in mind, that OpenFOAM can only do implicit evaluation based on the information of two adjacent points, the basis vectors of equation(5.28) are chosen such that the first basis vector is aligning with line connecting the owners' and the neighbours' centroids. ξ is the direction of the line connecting the owner and the neighbour centroids, and direction η is defined as:

$$\mathbf{i}_\eta = \mathbf{e}_\gamma - \mathbf{i}_\xi \tag{5.66}$$

Hence,

$$\mathbf{e}_\gamma \cdot \nabla p_f = \frac{\partial p}{\partial x} e_{\gamma,x} + \frac{\partial p}{\partial y} e_{\gamma,y} + \frac{\partial p}{\partial z} e_{\gamma,z} = \frac{\partial p}{\partial \xi} i_\xi + \frac{\partial p}{\partial \eta} i_\eta \tag{5.67}$$

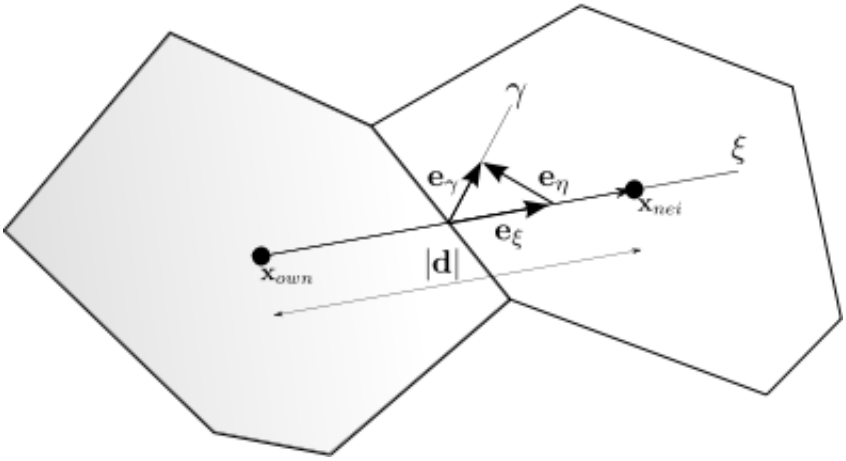

Figure 5.6: Control volumes with local coordinate directions

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \left(\frac{\partial p}{\partial \xi} i_\xi + \frac{\partial p}{\partial \eta} i_\eta \right) \tag{5.68}$$

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \left(\frac{\partial p}{\partial \xi} i_\xi + \frac{(\mathbf{e}_\gamma - \mathbf{i}_\xi)}{|\mathbf{e}_\gamma - \mathbf{i}_\xi|} \cdot \nabla p_f \, i_\eta \right) \tag{5.69}$$

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \left(\frac{\partial p}{\partial \xi} i_\xi + (\mathbf{e}_\gamma - \mathbf{i}_\xi) \cdot \nabla p_f \right) \tag{5.70}$$

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| i_\xi \frac{\partial p}{\partial \xi} - |\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \left(\mathbf{e}_\gamma - \frac{\mathbf{d}}{|\mathbf{d}|} i_\xi \right) \cdot \nabla p_f \tag{5.71}$$

Equation (5.71) shows how the diffusive flux can be split into two parts. The first part on the RHS can be treated implicitly, whereas the second term on the RHS is calculated explicitly such as in equation(5.39). It is also easy to see that the influence of the explicit and the implicit parts can be adjusted since the choice of the length of the basis vector $\mathbf{i}_\xi$ is arbitrary. As for the momentum equation's diffusion term the over-relaxed approach has been chosen.

$$i_\xi = \frac{|\mathbf{d}|}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f| \tag{5.72}$$

Hence equation(5.71) can be written as:

$$F_{RC,im} = -|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \frac{|\mathbf{d}|}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f| \frac{\partial p}{\partial \xi} - |\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \left(\frac{\mathbf{S}_f \cdot \overline{\mathbf{D}_f}}{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}|} - \frac{\mathbf{d}}{|\mathbf{d}|} \frac{|\mathbf{d}|}{\mathbf{d} \cdot \mathbf{S}_f} |\mathbf{S}_f| \right) \cdot (\nabla p)_f \tag{5.73}$$

Equation(5.73) can be further simplified to yield.

$$F_{RC,im} = -\frac{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \, |\mathbf{S}_f|}{\mathbf{d} \cdot \mathbf{S}_f} (p_{nei} - p_{own}) - \underbrace{\left(\mathbf{S}_f \cdot \overline{\mathbf{D}_f} - \frac{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \, |\mathbf{S}_f|}{\mathbf{d} \cdot \mathbf{S}_f} \mathbf{d} \right)}_{\mathbf{N}_{p,f}} \cdot (\nabla p)_f \tag{5.74}$$

The first term of equation(5.74) gives an implicit contribution, whereas the second term has to be evaluated explicitly. Hereby the explicit part will be updated through sub-looping with the latest variable value (in this case p_f^*). Calculating the explicit gradients for the second term is done as follows.

$$(\nabla p)_{cell} = \frac{1}{V_{cell}} \sum_f \mathbf{S}_f p_f \tag{5.75}$$

and

$$(\nabla p)_f = g_f \, (\nabla p)_{own} + (1 - g_f) \, (\nabla p)_{nei} \tag{5.76}$$

As already shown for the momentum equations, for the sake of consistency we have to subtract the directional part of the interpolated gradient and add the direct surface gradient evaluation in direction of $\mathbf{d}$. This yields,

$$(\nabla p)_f = \left(\overline{\nabla p}\right)_f - \left(\left(\overline{\nabla p}\right)_f \cdot \frac{\mathbf{d}}{|\mathbf{d}|} \right) \frac{\mathbf{d}}{|\mathbf{d}|} + \frac{(p_{nei} - p_{own})}{|\mathbf{d}|} \frac{\mathbf{d}}{|\mathbf{d}|} \tag{5.77}$$

$$a^{pp}_{own} = \sum_{f_{int}} \frac{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \, |\mathbf{S}_f|}{\mathbf{d} \cdot \mathbf{S}_f} \qquad a^{pp}_{nei} = -\frac{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}| \, |\mathbf{S}_f|}{\mathbf{d} \cdot \mathbf{S}_f} \tag{5.78}$$

$$b^p_{own} = \sum_{f_{int}} \mathbf{N}_{p,f} \cdot (\nabla p)_f \tag{5.79}$$

Explicit numerical diffusion term (explicit Rhie-Chow term):

$$F_{RC,ex} = \mathbf{S}_f \cdot \overline{\mathbf{D}_f} \cdot \overline{\nabla p_f} \tag{5.80}$$

$$b^p_{own} = \sum_{f_{int}} \mathbf{S}_f \cdot \overline{\mathbf{D}_f} \cdot \overline{\nabla p_f} \tag{5.81}$$

5.3 Discretisation of Governing Equations for Moving Control Volumes

Regarding the discretisation of the momentum and continuity equation for moving control volumes, mesh deformations shall not be subject of this section. Only a look at the remaining terms, as given by equation (3.82) shall be taken. In the case of non-deforming meshes performing a rigid body motion, the governing equations for moving control volumes are identical to those of non-moving control volumes, except for the momentum equations' convection term.

5.3.1 Discretisation of Momentum Equation

This term can easily be evaluated by explicitly calculating an ALE (Arbitrary Lagrange Euler) flux relative to the mesh motion. The upwind cell is determined by the relative ALE flux and not by the absolute flux.

Recalling equation(3.82) the momentum equation reads,

$$\int\limits_{V_x(t)} \frac{\partial \rho \mathbf{u}}{\partial t} \, \mathrm{dV_x} + \oint\limits_{S_x(t)} \mathbf{n} \cdot \left(\rho \left(\mathbf{u} - \mathbf{u}_g\right) \mathbf{u}\right) \, \mathrm{dS_x} = \oint\limits_{S_x(t)} \mathbf{n} \cdot \underline{\underline{\sigma}} \mathrm{dS_x} + \int\limits_{V_x(t)} \rho \mathbf{g} \mathrm{dV_x} \tag{5.82}$$

The semi-discretised form then reads,

$$\frac{\delta \mathbf{u}}{\delta t} \delta V + \sum_{faces} \mathbf{S}_f \cdot \left((\mathbf{u} - \mathbf{u}_g)\mathbf{u}\right)_f = - \sum_{faces} \mathbf{S}_f p_f + \sum_{faces} \mathbf{S}_f \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}\right)_f\right) \tag{5.83}$$

Convective Term:

$$\sum_{faces} \mathbf{S}_f \cdot \left((\mathbf{u} - \mathbf{u}_g)\mathbf{u}\right)_f \tag{5.84}$$

Introducing the ALE flux phi_{ALE},

$$\sum_{faces} phi_{ALE}\mathbf{u}_f \tag{5.85}$$

The ALE flux therein being,

$$phi_{ALE} = \mathbf{S}_f \cdot \mathbf{u} - \mathbf{S}_f \cdot \mathbf{u}_g \tag{5.86}$$

Hence the ALE flux consist of the absolute volume flux subtracted by the grid movement flux.

$$phi_{ALE} = phi_f - phi_g \tag{5.87}$$

For a deferred corrected van Leer TVD scheme for the implicit part of equation (5.84) we obtain a contribution in form of matrix coefficients and a contribution to the source (arising from the deferred correction) for each cell,

$$\begin{aligned}
a_{own}^{uu} &= \sum_{f_{int}} \|phi_{ALE}^*, 0\| \qquad a_{nei}^{uu} = -\| - phi_{ALE}^*, 0\| \\
a_{own}^{vv} &= \sum_{f_{int}} \|phi_{ALE}^*, 0\| \qquad a_{nei}^{vv} = -\| - phi_{ALE}^*, 0\| \\
a_{own}^{ww} &= \sum_{f_{int}} \|phi_{ALE}^*, 0\| \qquad a_{nei}^{ww} = -\| - phi_{ALE}^*, 0\|
\end{aligned} \tag{5.88}$$

$$\begin{aligned}
b_{own}^{u} &= -\sum_{f_{int}} \|phi_{ALE}^*, 0\| (1 - g_f) \Upsilon (u_{nei}^* - u_{own}^*) - \| - phi_{ALE}^*, 0\| g_f \Upsilon (u_{own}^* - u_{nei}^*) \\
b_{own}^{v} &= -\sum_{f_{int}} \|phi_{ALE}^*, 0\| (1 - g_f) \Upsilon (v_{nei}^* - v_{own}^*) - \| - phi_{ALE}^*, 0\| g_f \Upsilon (v_{own}^* - v_{nei}^*) \\
b_{own}^{w} &= -\sum_{f_{int}} \|phi_{ALE}^*, 0\| (1 - g_f) \Upsilon (w_{nei}^* - w_{own}^*) - \| - phi_{ALE}^*, 0\| g_f \Upsilon (w_{own}^* - w_{nei}^*)
\end{aligned} \tag{5.89}$$

5.3.2 Discretisation of Continuity Equation

The continuity equation for moving control volumes given by equation (3.76) simplified for non-deforming meshes, rigid body motion and constant density reduces to:

$$\sum_{faces} \mathbf{S}_f \cdot \mathbf{u}_f = 0 \tag{5.90}$$

This is the same continuity equation as found for inertial frames. The only difference that occurs during the discretisation is that instead of the volume flux *phi* the relative ALE volume flux phi_{ALE} now influences the numerical diffusion tensor $\mathbf{D}$ for the Rhie-Chow interpolation.

5.4 Discretisation of Governing Equations for Rotating Frames

In respect to the governing equations in inertial reference frame, there are only three things that are different. Namely the convection term of the momentum equations, the additional rotational term of the momentum equations, and the diffusion tensor obtained by Rhie-Chow interpolation, which is derived differently.

5.4.1 Discretisation of Momentum Equation

Following equation (3.121) the momentum equations in integral form can be written as:

$$\int_V \frac{\partial \mathbf{u}_a}{\partial t} \mathrm{d}V + \int_V \nabla \cdot (\mathbf{u}_r \mathbf{u}_a)\, \mathrm{d}V + \int_V \Omega \times \mathbf{u}_a \, \mathrm{d}V = -\int_V \nabla p \mathrm{d}V + \int_V \nabla \cdot (\nu_{eff} \nabla \mathbf{u})\, \mathrm{d}V \quad (5.91)$$

Note that in OpenFOAM the pressure variable actually corresponds to a pressure divided by a constant density.

Gauss' Theorem leads to:

$$\int_V \frac{\partial \mathbf{u}_a}{\partial t} \mathrm{d}V + \oint_S \mathbf{n} \cdot (\mathbf{u}_r \mathbf{u}_a)_f \, \mathrm{dS} + \int_V \Omega \times \mathbf{u}_a \, \mathrm{d}V = -\oint_S \mathbf{n} p_f \, \mathrm{dS} + \oint_S \mathbf{n} \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}_a \right)_f \right) \mathrm{dS} \quad (5.92)$$

Semi-discretised form:

$$\frac{\delta \mathbf{u}_a}{\delta t} \delta V + \sum_{faces} \mathbf{S}_f \cdot (\mathbf{u}_r \mathbf{u}_a)_f + (\Omega \times \mathbf{u}_a)\, \delta V = -\sum_{faces} \mathbf{S}_f p_f + \sum_{faces} \mathbf{S}_f \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}_a \right)_f \right) \quad (5.93)$$

The convection term can be developed as follows:

$$\sum_{faces} \mathbf{S}_f \cdot (\mathbf{u}_r \mathbf{u}_a)_f \tag{5.94}$$

$$\sum_{faces} (\mathbf{S}_f \cdot \mathbf{u}_{r,f})\, \mathbf{u}_a \tag{5.95}$$

Introducing the relative flux $phi_{r,f}$ gives:

$$\sum_{faces} phi_{r,f} \mathbf{u}_a \tag{5.96}$$

The relative flux $phi_{r,f}$ can be obtained by extracting the absolute flux $phi_{a,f}$ from the matrix coefficients of the continuity equation and using the following relation.

$$phi_{r,f} = \mathbf{S}_f \cdot \mathbf{u}_{r,f} = \mathbf{S}_f \cdot \mathbf{u}_{a,f} - \mathbf{S}_f \cdot (\Omega \times \mathbf{r}_{0P}) \tag{5.97}$$

$$phi_{r,f} = phi_{a,f} - \mathbf{S}_f \cdot (\Omega \times \mathbf{r}_{0P}) \tag{5.98}$$

This equation can be transformed to:

$$\frac{\delta \mathbf{u}_a}{\delta t}\delta V + \sum_{faces} phi_{r,f} \mathbf{u}_{a,f} + (\Omega \times \mathbf{u}_a)\, \delta V + \sum_{faces} \mathbf{S}_f p_f - \sum_{faces} \mathbf{S}_f \cdot \left(\nu_{eff} \left(\nabla \mathbf{u}_a\right)_f\right) = 0 \tag{5.99}$$

And it can be written in terms of forces,

$$\frac{\delta \mathbf{u}_a}{\delta t}\delta V + (\Omega \times \mathbf{u}_a)\, \delta V + \sum_{faces} \mathbf{F}_f^C + \sum_{faces} \mathbf{F}_f^P - \sum_{faces} \mathbf{F}_f^D = 0 \tag{5.100}$$

Convective Term:

In this work the convective term has been linearising using the relative volume fluxes of the previous time step (or iteration). The upwind part of the convection term reads,

$$\begin{aligned}
\sum_{f_{int}} a_{own}^{uu} &= |phi_{r,f}^{*}, 0| \qquad a_{nei}^{uu} = -|-phi_{r,f}^{*}, 0| \\
\sum_{f_{int}} a_{own}^{vv} &= |phi_{r,f}^{*}, 0| \qquad a_{nei}^{vv} = -|-phi_{r,f}^{*}, 0| \\
\sum_{f_{int}} a_{own}^{ww} &= |phi_{r,f}^{*}, 0| \qquad a_{nei}^{ww} = -|-phi_{r,f}^{*}, 0|
\end{aligned} \tag{5.101}$$

The deferred correction part yields following explicit contribution,

$$\begin{aligned}
b_{own}^{u} &= -\sum_{f_{int}} ||phi_{r,f}^{*}, 0||\,(1-g_f)\,\Upsilon\,(u_{nei}^{*}-u_{own}^{*}) - ||-phi_{r,f}^{*}, 0||g_f\Upsilon\,(u_{own}^{*}-u_{nei}^{*}) \\
b_{own}^{v} &= -\sum_{f_{int}} ||phi_{r,f}^{*}, 0||\,(1-g_f)\,\Upsilon\,(v_{nei}^{*}-v_{own}^{*}) - ||-phi_{r,f}^{*}, 0||g_f\Upsilon\,(v_{own}^{*}-v_{nei}^{*}) \\
b_{own}^{w} &= -\sum_{f_{int}} ||phi_{r,f}^{*}, 0||\,(1-g_f)\,\Upsilon\,(w_{nei}^{*}-w_{own}^{*}) - ||-phi_{r,f}^{*}, 0||g_f\Upsilon\,(w_{own}^{*}-w_{nei}^{*})
\end{aligned} \tag{5.102}$$

Rotational Term:

The rotational force term that arises for the momentum equations in rotational frame formulation leads to the following implicit coefficients.

$$\begin{aligned}
a_{own}^{uv} &= -\Omega_z\,\Delta V \qquad a_{own}^{uw} = \Omega_y\,\Delta V \\
a_{own}^{vu} &= \Omega_z\,\Delta V \qquad a_{own}^{vw} = -\Omega_x\,\Delta V \\
a_{own}^{wu} &= -\Omega_y\,\Delta V \qquad a_{own}^{wv} = \Omega_x\,\Delta V
\end{aligned} \tag{5.103}$$

5.4.2 Discretisation of Continuity Equation

The continuity equation for rotating reference frames was derived in section (3.3), were it has been shown that it is equal to the continuity equation (5.49) for the inertial reference frame. Recall the semi-discretised form:

$$\sum_{faces} \mathbf{S}_f \cdot \mathbf{u}_f = 0 \tag{5.104}$$

Numerical diffusion tensor:

As already laid out for the inertial reference frame, the non-occurrence of the pressure in the continuity equation for incompressible flows, leads to an ill conditioned matrix. The previously suggested Rhie and Chow [23] interpolation technique (equation (5.53)) for the face velocities $\mathbf{u}_f$, which is obtained by contrasting juxtaposition of the pressure term with the (false) transient (T), the convection (C) and the diffusion terms (D), can be extended to momentum equations in rotating reference frames. Thereby the contrasting juxtaposition also has to include the additional rotational term (R) that occurs in the momentum equations.

Following Rhie and Chow[23] the surface interpolation to obtain $\mathbf{u}_f$ shall take the physics of the momentum equations into account. Hereby it is assumed that the momentum equations evaluated at a face is approximately equal to the linear interpolation to the face of the evaluated momentum equations at the cells adjacent to that face. For steady incompressible flows in rotational reference frame without body forces this yields,

$$\left(\frac{\mathbf{A}_{TCDR}}{\Delta V}\right)_f \mathbf{u}_f + \nabla p_f \approx \overline{\left(\frac{\mathbf{A}_{TCDR}}{\Delta V}\right)_f} \, \overline{\mathbf{u}_f} + \overline{\nabla p_f} \tag{5.105}$$

$\mathbf{A}_{TCDR,f}$ being a 3x3 submatrix that contains the convection coefficients, diffusion coefficients and the rotational term's coefficients of the previously descritised momentum equations. Assuming that $\left(\frac{\mathbf{A}_{TCDR}}{\Delta V}\right)_f \approx \overline{\left(\frac{\mathbf{A}_{TCDR}}{\Delta V}\right)_f}$, we obtain,

$$\mathbf{u}_f = \overline{\mathbf{u}_f} + \overline{\left(\frac{\mathbf{A}_{TCDR}}{\Delta V}\right)^{-1}}_f \cdot \left(\overline{\nabla p_f} - \nabla p_f\right) \tag{5.106}$$

Hence the continuity equation adapted by the Rhie-Chow interpolation reads:

$$\sum_{faces} \mathbf{S}_f \cdot \left(\overline{\mathbf{u}}_f + \mathbf{D}_f \cdot \left(\overline{\nabla p}_f - \nabla p_f\right)\right) = 0 \tag{5.107}$$

The resulting discretised terms for the implicit and explicit Laplacian of the pressure are identical to the ones of equations (5.80)(5.60) with the diffusion tensor $\mathbf{D}_f$ being the one obtained by the above formulation.

Chapter 6

Discretisation At Boundary Faces

Contents

As a continuation of the previous chapter (5) the discretisation at boundaries shall be introduced in what follows, according to the concept of block coupled algorithms (chapter 4).

The discretisation of the governing equations at boundary faces can be a trivial matter. However, some boundary conditions may need special attention, especially when dealing with block coupled systems of equations. In this case off-diagonal coefficients may have to be inserted into the block coefficients of the cell adjacent to the boundary face. Examples for boundary conditions, for which coupling terms arise are i.e. the total pressure boundary condition, or the no-slip wall boundary condition (shear stress). The proper implementation of such boundary conditions is of utmost importance, since it highly effects the numerical properties of the system, and therefore its robustness and convergence speed. In the following sections boundary conditions with and without coupling terms shall be outlined. Simpler boundary conditions

that only inject diagonal (say segregated) elements into the solution matrix have already been implemented in the OpenFOAM framework and can be used without change for the outlined coupled algorithms. The most important segregated boundary conditions shall be outlined quickly. Subsequently boundary conditions with coupling terms shall be outlined following Darwish et al. [26]. The author attaches importance to mention the contribution of Luca Mangani and Marwan Darwish to the implementation of the coupled boundary conditions in the coupled framework of OpenFOAM.

6.1 Dirichlet Boundary Condition

The Dirichlet (or first type) boundary condition applied to the block coupled pressure velocity system is very simple to implement. For the continuity and momentum equations it does not introduce any coupling terms.

Mathematically it can be described as:

$$y(\mathbf{x}, t) = f(\mathbf{x}, t) \quad \forall x \in S \tag{6.1}$$

$y(\mathbf{x}, t)$ is the primitive variable that is to be solved for. Hence a field of the primitive variable has to be prescribed at the surface. The field value in conjunction with the derived discretised coefficients gives a contribution of the boundary face to its adjacent cell. This contribution can be directly added to the source term of the respective linearised equation.

6.2 Neumann Boundary Condition

The Neumann (or second type) boundary condition applied to the block coupled pressure velocity system is also very simple to implement. Like the Dirichlet boundary condition it does not introduce coupling terms.

Mathematically it can be described as:

$$\frac{\partial y}{\partial \mathbf{n}}(\mathbf{x}, \mathbf{t}) = f(\mathbf{x}, \mathbf{t}) \quad \forall \mathbf{x} \in S. \tag{6.2}$$

$y(\mathbf{x}, t)$ is the primitive variable that is to be solved for. Hence the surface normal derivative has to be prescribed at the boundary face. A special type of the Neumann boundary condition is the zero gradient boundary condition. This means that a thought cell next to the boundary cell has the same primitive variable value as the boundary cell itself. The resulting coefficients can be added implicitly.

6.3 Wall Boundary Condition

At a wall the volume flux *phi* over the boundary face is zero. Hence, there is no contribution of the wall to the continuity equation. Furthermore the contribution of the convection term in the momentum equations will be zero.

The pressure and shear forces will obviously give contributions to the momentum equations.

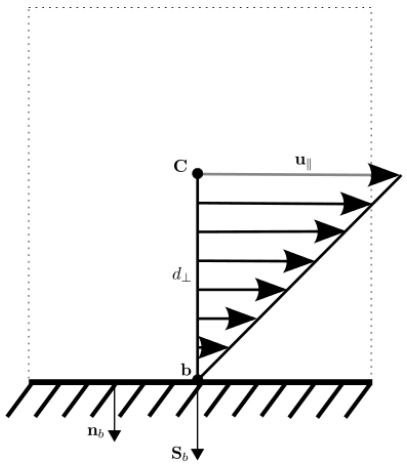

Figure 6.1: Velocity profile in the near wall region of a non-moving wall

In the current approach a zero order extrapolation of the pressure at the boundary cell to its boundary face has been chosen to evaluate the pressure forces.

$$\mathbf{F}_{pressure} = p_b \mathbf{S}_b = p_c \mathbf{S}_b \tag{6.3}$$

The resulting coefficients can be added implicitly to the linearised equations and read:

$$a_{own}^{up} = S_{b_x} \qquad a_{own}^{vp} = S_{b_y} \qquad a_{own}^{wp} = S_{b_z} \tag{6.4}$$

Darwish [26] suggests a more accurate extrapolation to the wall. This first order accurate approach could lead to a better prediction of the pressure at the wall, and finally to a better prediction of detachments.

Following [26] the shear stress can be computed as:

$$\mathbf{F}_{shear} = \tau_b \, ||\mathbf{S}_b|| \tag{6.5}$$

with the shear stress at the boundary being:

$$\frac{\tau_b}{\rho} = -\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \tag{6.6}$$

For non-moving walls discretisation yields:

$$\frac{\tau_b}{\rho} = -\nu \frac{\mathbf{u}_{\parallel}}{d_{\perp}} \tag{6.7}$$

Using the surface normal unit vector, the parallel velocity can be obtained as:

$$\mathbf{u}_{\parallel} = \mathbf{u} - (\mathbf{u} \cdot \mathbf{n}_b) \, \mathbf{n}_b \tag{6.8}$$

Hence,

$$\frac{\mathbf{F}_{shear}}{\rho} = -\frac{\nu \, ||\mathbf{S}_f||}{d_{\perp}} \left\{ \begin{array}{l} u - (u \cdot n_{b,x} + v \cdot n_{b,y} + w \cdot n_{b,z}) \, n_{b,x} \\ v - (u \cdot n_{b,x} + v \cdot n_{b,y} + w \cdot n_{b,z}) \, n_{b,y} \\ w - (u \cdot n_{b,x} + v \cdot n_{b,y} + w \cdot n_{b,z}) \, n_{b,z} \end{array} \right\} \tag{6.9}$$

The shear force term yields both diagonal (segregated) and off-diagonal contributions to the coefficient matrix. The coefficients for a boundary cell, arising from the shear force at the boundary, hence read:

$$
\begin{aligned}
a_{own}^{uu} &= \frac{\nu\,||\mathbf{S}_b||}{d_\perp}\left(1-n_{b,x}^2\right) & a_{own}^{uv} &= -\frac{\nu\,||\mathbf{S}_b||}{d_\perp}\,n_{b,x}n_{b,y} & a_{own}^{uw} &= -\frac{\nu\,||\mathbf{S}_b||}{d_\perp}\,n_{b,x}n_{b,z} \\
a_{own}^{vu} &= -\frac{\nu\,||\mathbf{S}_b||}{d_\perp}\,n_{b,y}n_{b,x} & a_{own}^{vv} &= \frac{\nu\,||\mathbf{S}_b||}{d_\perp}\left(1-n_{b,y}^2\right) & a_{own}^{vw} &= -\frac{\nu\,||\mathbf{S}_b||}{d_\perp}\,n_{b,y}n_{b,z} \\
a_{own}^{wu} &= -\frac{\nu\,||\mathbf{S}_b||}{d_\perp}\,n_{b,z}n_{b,x} & a_{own}^{wv} &= -\frac{\nu\,||\mathbf{S}_b||}{d_\perp}\,n_{b,z}n_{b,y} & a_{own}^{ww} &= \frac{\nu\,||\mathbf{S}_b||}{d_\perp}\left(1-n_{b,z}^2\right)
\end{aligned}
\tag{6.10}
$$

6.4 Moving Wall Boundary Condition

The moving wall boundary condition is very similar to the wall boundary condition. The difference being that the shear stress is now defined by the relative parallel velocity $(\mathbf{u}_{\parallel} - \mathbf{u}_{\parallel,wall})$ between the boundary cell and the wall.

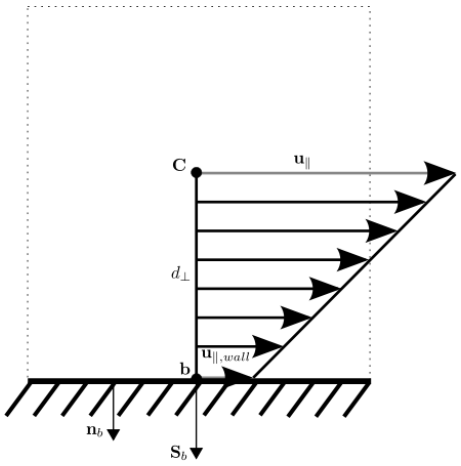

Figure 6.2: Velocity profile in the near wall region of a moving wall

At the wall the volume flux *phi* over the boundary face is again zero. Hence, there is no contribution of the wall to the continuity equation. Furthermore the contribution of the convection term in the momentum equations are zero.

The pressure and shear forces again give contributions to the momentum equations.

As for the pure wall boundary condition a zero order extrapolation of the pressure is chosen to evaluate the pressure forces.

$$\mathbf{F}_{pressure} = p_b \, \mathbf{S}_b = p_c \, \mathbf{S}_b \tag{6.11}$$

The resulting implicit coefficients are:

$$a_{own}^{up} = S_{b_x} \qquad a_{own}^{vp} = S_{b_y} \qquad a_{own}^{wp} = S_{b_z} \tag{6.12}$$

Following [26] the shear stress reads:

$$\mathbf{F}_{shear} = \tau_b \, ||\mathbf{S}_f|| \tag{6.13}$$

with the shear stress at the boundary being:

$$\frac{\tau_b}{\rho} = -\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \tag{6.14}$$

For moving walls discretisation yields:

$$\frac{\tau_b}{\rho} = -\nu \frac{\mathbf{u}_{\parallel} - \mathbf{u}_{\parallel,wall}}{d_{\perp}} \tag{6.15}$$

The parallel velocity vector at the boundary cell is obtained as in equation (6.8). The parallel wall velocity is obtained the same way. The implicit contribution remains unchanged in respect to the pure wall boundary condition. An explicit contribution is however added, which accounts for the parallel wall velocity. The implicit and explicit coefficients read:

$$\begin{aligned}
a_{own}^{uu} &= \frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \left(1 - n_{b,x}^2\right) & a_{own}^{uv} &= -\frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \, n_{b,x} n_{b,y} & a_{own}^{uw} &= -\frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \, n_{b,x} n_{b,z} \\
a_{own}^{vu} &= -\frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \, n_{b,y} n_{b,x} & a_{own}^{vv} &= \frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \left(1 - n_{b,y}^2\right) & a_{own}^{vw} &= -\frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \, n_{b,y} n_{b,z} \\
a_{own}^{wu} &= -\frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \, n_{b,z} n_{b,x} & a_{own}^{wv} &= -\frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \, n_{b,z} n_{b,y} & a_{own}^{ww} &= \frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \left(1 - n_{b,z}^2\right)
\end{aligned} \tag{6.16}$$

$$\begin{aligned}
b_{own}^{u} &= \frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \cdot \left(u_{wall} - \left(u_{wall} \cdot n_{b,x} + v_{wall} \cdot n_{b,y} + w_{wall} \cdot n_{b,z}\right) n_{b,x}\right) \\
b_{own}^{v} &= \frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \cdot \left(v_{wall} - \left(u_{wall} \cdot n_{b,x} + v_{wall} \cdot n_{b,y} + w_{wall} \cdot n_{b,z}\right) n_{b,y}\right) \\
b_{own}^{w} &= \frac{\nu \, ||\mathbf{S}_b||}{d_{\perp}} \cdot \left(w_{wall} - \left(u_{wall} \cdot n_{b,x} + v_{wall} \cdot n_{b,y} + w_{wall} \cdot n_{b,z}\right) n_{b,z}\right)
\end{aligned} \tag{6.17}$$

6.5 Slip Wall Boundary Condition

At a wall the volume flux *phi* over the boundary face is again zero. Hence, there is again no contribution to the continuity equation. Furthermore the contribution of the convection term in the momentum equations is zero. By definition of the slip wall boundary condition, there is no shear stress at the wall, thus there is no contribution coming from the shear stress tensor.

The pressure forces are the only terms that give contributions to the momentum equations.

Zero order extrapolation of the pressure yields:

$$\mathbf{F}_{pressure} = p_b \mathbf{S}_f = p_c \mathbf{S}_b \tag{6.18}$$

The resulting coefficients can be added implicitly to the linearised equations and read:

$$a_{own}^{up} = S_{b_x} \qquad a_{own}^{vp} = S_{b_y} \qquad a_{own}^{wp} = S_{b_z} \tag{6.19}$$

After solving the resulting set of equations a velocity field is obtained. The velocity vectors adjacent to the slip boundary are not necessarily parallel to the wall, therefore they are corrected by subtracting the part normal to the boundary.

Chapter 7

Implementation Of Block Coupled Interfaces

Contents

In what follows the implicit discretisation of block interfaces according to the concept of block coupled algorithms (chapter 4) shall be outlined.

In the present work two block interfaces that are of utmost importance in CFD have been implemented, namely, an arbitrary mesh interface (AMI) with non-conforming faces (which can also be periodic) and a processor interface for block coefficients. Both shall be outlined in the following sections.

7.1 Arbitrary Mesh Interface (AMI)

Arbitrary mesh interfaces are patches that connect adjacent blocks of a mesh. Blocks can have patches that match conformally adjacent counterparts on neighbouring blocks. The faces of a patch, however, do not necessarily have to match the faces of its counter-patch conformally. In the case of non-conforming faces the surface discretisation has to account for this overlapping using face fraction weights w_f. Figure 7.1 shows such a patch pair with non-conforming faces. The surface discretisation over these faces is carried out over each weighted sub-surface of the respective master surface, all of which have been obtained using a special face cutting algorithm.

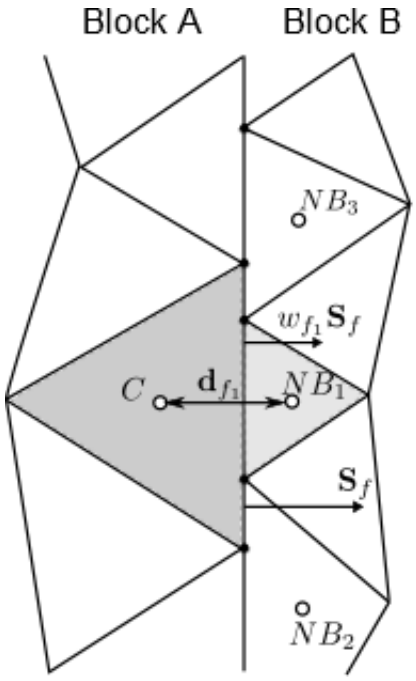

Figure 7.1: Non-conforming AMI interface

The AMI discretisation of the pressure Laplacian given by equation (5.74), shall provide an example of how a conservative discretisation over a weighted sub-surface is carried out. Considering one such sub-surface with a face fraction weight w_{f_1} we have:

$$
\begin{aligned}
-w_{f_1}\mathbf{S}_f \cdot \overline{\mathbf{D}_f} \cdot \nabla p_f = &- \frac{|w_{f_1}\mathbf{S}_f \cdot \overline{\mathbf{D}_f}||w_{f_1}\mathbf{S}_f|}{\mathbf{d}_{f_1} \cdot w_{f_1}\mathbf{S}_f} \left(p_{NB_1} - p_C\right) \\
&- \underbrace{\left(w_{f_1}\mathbf{S}_f \cdot \overline{\mathbf{D}_f} - \frac{|w_{f_1}\mathbf{S}_f \cdot \overline{\mathbf{D}_f}|\,|w_{f_1}\mathbf{S}_f|}{\mathbf{d}_{f_1} \cdot w_{f_1}\mathbf{S}_f}\mathbf{d}_{f_1}\right)}_{\mathbf{N}_{p,f}} \cdot \overline{\nabla p_f}
\end{aligned}
\tag{7.1}
$$

The implicit part (the first term on the RHS of equation(7.1)) can be linearised as:

$$
a_C^{pp} = w_{f_1}\frac{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}|\,|\mathbf{S}_f|}{\mathbf{d}_{f_1} \cdot \mathbf{S}_f} \qquad a_{NB_1}^{pp} = -w_{f_1}\frac{|\mathbf{S}_f \cdot \overline{\mathbf{D}_f}|\,|\mathbf{S}_f|}{\mathbf{d}_{f_1} \cdot \mathbf{S}_f} \tag{7.2}
$$

Whereas the explicit part is moved to the RHS.

The discretisation at periodic interfaces is equivalent to connecting two adjacent blocks (even though effectively only a patch of a block is connected to another patch of the same block). These periodic interfaces can also have non-conforming faces and can even be cyclic. The discretisation procedure of these so-called cyclic AMI interfaces follows the same routine as that of an ordinary AMI. However, rotational transformations are necessary to account for the cyclic periodicity.

Therefore, before discretising over a master patch the cell centroid of the slave patches' cells as well as the slaves' face points have to be transformed (rotated).

Based on these newly obtained geometric positions a face cutting algorithm is carried out to evaluate the surface fraction weights w_f. Then other geometric quantities such as the centroid distance vectors $\mathbf{d}$ have to be calculated. Once all the geometric entities are readily processed, the usual discretisation can be carried out, followed by a transformation to account for the velocity's rotation.

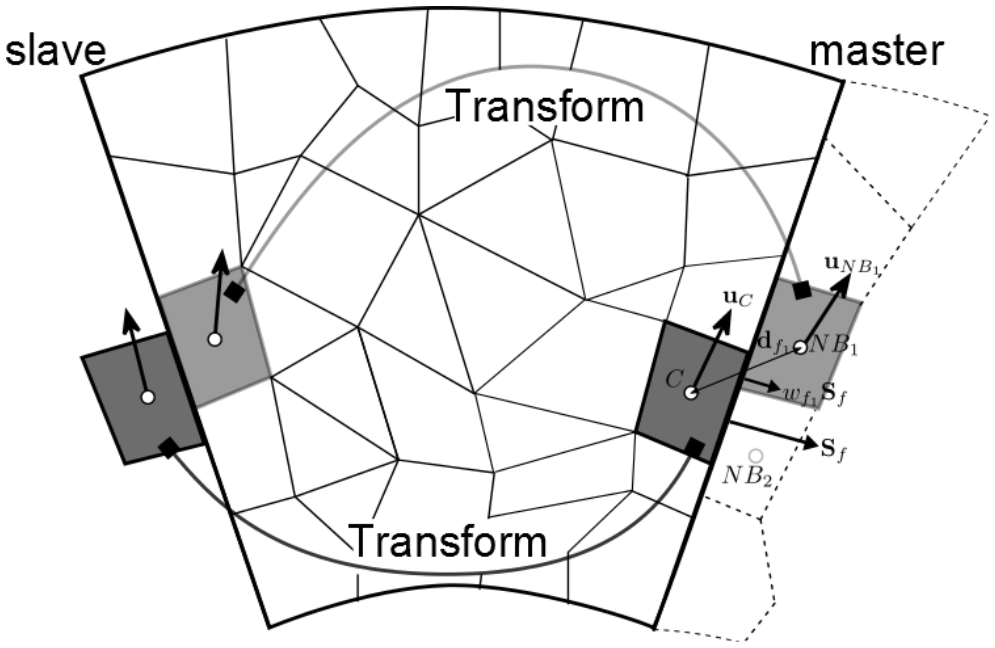

Figure 7.2: Non-conforming cyclic AMI interface

A sketch of a cyclic AMI interface with rotational periodicity and non-conforming faces is shown in Figure 7.2. To obtain a conservative implicit discretisation for a patch pair the following steps have to be carried out. First the adjacent cells' face points and cell centres of each patch (master and slave side) have to be transformed (rotated). Based on these newly obtained geometric positions a face cutting algorithm is carried out to evaluate the surface fraction weights w_f. Once all the geometric entities are readily processed, the respective cell centre distance vectors, distance weights and surface normal vectors for each neighbouring cell pair can be calculated. Implicit block coefficients can now be calculated for each term in the governing equations. These coefficients are conservative since the surface integration is carried out for each face fraction separately. This yields preliminary coefficients $\mathbf{A}_{NB,geom}$, which have to be dotted with a transformation tensor in order to account for the primitive variables' rotation (if the primitive variables are vector components).

$$\mathbf{A}_{NB} = \mathbf{A}_{NB,geom} \cdot \mathbf{T}_{NB} \tag{7.3}$$

For given pressure-velocity matrix system the following 4x4 transformation tensor for the neigh-

bouring primitive variables ($\mathbf{u}$ and p) is valid for a rotation about a unit rotation vector ω and a rotation angle α.

$$\mathbf{T}_{NB} = \begin{bmatrix} c(\alpha) + \omega_x^2(1 - c(\alpha)) & \omega_x\omega_y(1 - c(\alpha)) - \omega_z s(\alpha) & \omega_y s(\alpha) + \omega_x\omega_z(1 - c(\alpha)) & 0 \\ \omega_z s(\alpha) + \omega_x\omega_y(1 - c(\alpha)) & c(\alpha) + \omega_y^2(1 - c(\alpha)) & -\omega_x s(\alpha) + \omega_y\omega_z(1 - c(\alpha)) & 0 \\ -\omega_y s(\alpha) + \omega_x\omega_z(1 - c(\alpha)) & \omega_x s(\alpha) + \omega_y\omega_z(1 - c(\alpha)) & c(\alpha) + \omega_z^2(1 - c(\alpha)) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

We can say that the coefficient and addressing arrays are "extended" since the coefficients of the cyclic AMI interfaces are added to the coefficient pattern arising from the discretisation at internal faces.

Injecting the implicit coefficients arising from the interfaces directly into the block matrix structure instead of treating them separately yields some advantages. First of all, there is no need to create interpolated cyclic AMI interfaces for coarse grid AMG levels, which would have been necessary if interfaces had been treated explicitly using a ghost cell approach. In other words, the algebraic agglomeration doesn't notice any interface, which has a very positive effect on the convergence of the linear solver. Additionally the assembly for the direct solver at the coarsest AMG level is also straight forward, since all coefficients are injected into the block matrix and the sparse matrix addressing pattern (including that of the injected interface coefficients) is known. Finally the update of the fluxes can be carried out by multiplying the variable vector directly with the continuity equation's matrix coefficients, avoiding possible inconsistencies in their reconstruction.

7.2 Processor Interfaces

Parallel processing in OpenFOAM is done via domain decomposition and the MPI (Message Passing Interface) library. The domain is divided into subsets and each subset is treated by a separate processor. The boundary conditions that have been applied to the complete domain are carried over to the subsets, and at the intersection between two subsets new boundaries, so-called processor interfaces are added (see Figure 7.3). The basic Gaussian face discretisation at these interfaces is done for all terms as if the interfaces' faces were internal faces. Explicit coefficients are directly added to the source term of the respective cells adjacent to these faces. Implicit diagonal coefficients can also be added to the matrix directly. However, since the solution vector for a specific processor does not contain the solution of a neighbouring processor's cells (i.e. the solution of cells adjacent to the processor's interface) the extra-diagonal coefficients arising from these neighbours can not directly be injected into the matrix of a subset, hence they are stored in a separate coefficient structure $\mathbf{A}_{proc}$

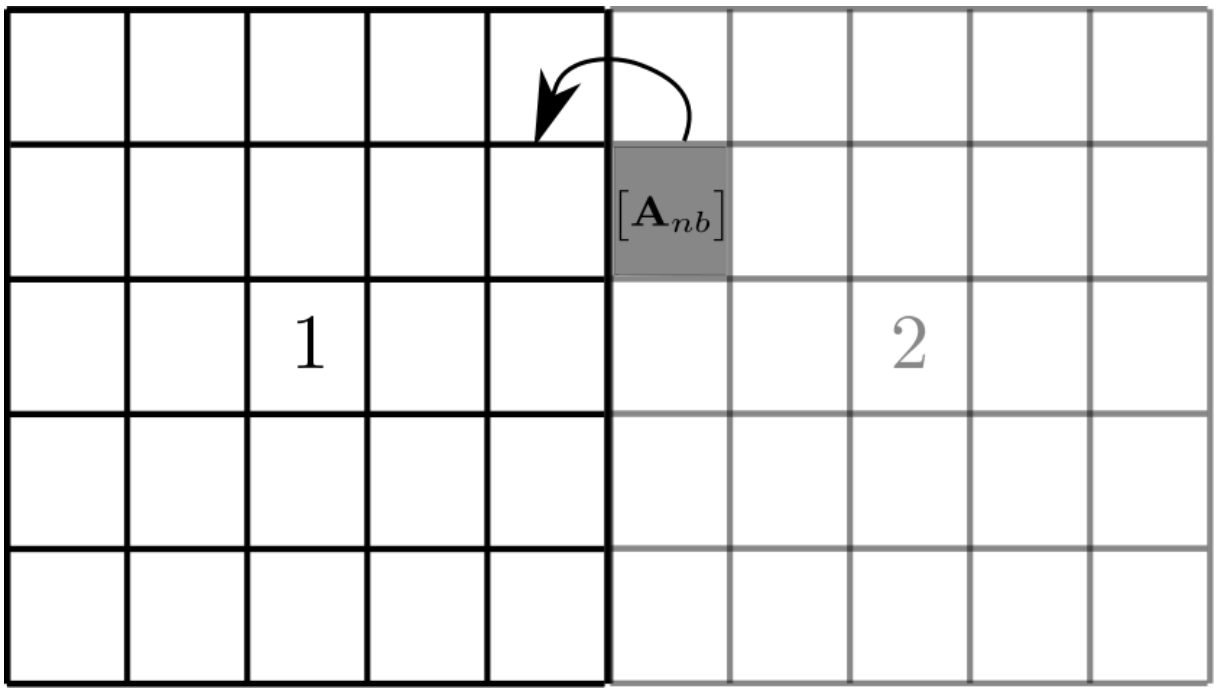

Figure 7.3: Processor interface

If we want to solve the system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ we can use iterative solvers. These iterative solvers employ successive calls of $\mathbf{A} \cdot \mathbf{x}$ multiplications. These multiplications can now be divided into

a so-called core multiplication and a subsequent processor multiplication update.

$$\mathbf{A}_{core} \cdot \mathbf{x}_{core} + \mathbf{A}_{proc} \cdot \mathbf{x}_{proc} = \mathbf{b} \tag{7.5}$$

In the case of the block AMG solver outlined in chapter (8) the solution step is carried out using block ILU(0) smoothers (section 8.3) on various grid levels. There, two problems are to be addressed. First, how are the interface coefficients added in the block ILU(0)'s algorithm (see 8.3)? Second, how are interface coefficients calculated and accounted for at coarse grid levels?

The answer to the first question goes as follows:

The to be solved system (equation (7.5)) is brought to the following correction form.

$$\mathbf{A}_{core} \cdot \mathbf{x}'_{core} = \mathbf{b} - \mathbf{A}_{core} \cdot \mathbf{x}^{old}_{core} - \mathbf{A}_{proc} \cdot \mathbf{x}^{old}_{proc} \tag{7.6}$$

Hence the influence of the processor coefficients and their adjacent cell values acts explicitly to the RHS.

Then the newly found system is evaluated by means of an approximate block ILU(0) smoother (for details see section(8.3)).

The newly found ILU system reads:

$$\mathbf{L}_{core} \cdot \mathbf{U}_{core} \cdot \mathbf{x}'_{core} = \mathbf{b} - \mathbf{A}_{core} \cdot \mathbf{x}^{old}_{core} - \mathbf{A}_{proc} \cdot \mathbf{x}^{old}_{proc} \tag{7.7}$$

The correction $\mathbf{x}'_{core}$ of the processors' solution vectors are subsequently added to the previous preliminary solution $\mathbf{x}^{old}_{core}$. Several such sweeps are carried out and after each iteration the RHS is adapted with the newly found solution vector at the interface $\mathbf{x}^{old}_{proc}$. Like this the information coming from an adjacent processor keeps being added in a semi-implicit fashion.

The answer to the second question goes as follows:

During the agglomeration process of the block AMG solver, also the block coefficients of cells adjacent to processor interfaces are additively merged. Figure 7.4 shows agglomerated cells at both sides of a processor interface. At processor 1 the agglomerated cell A now matches two cells (agglom. cell B and agglom. cell C). However due to the additive agglomeration process only a single neighbour block coefficient is stored for the agglomerated cell A. Therefore a "ghost" cell solution vector (denoted neighbour ghost cell) is calculated matching the boundary face of the agglomerated cell A. This is done through simple weighted averaging using face weights of the opposite cells.

Figure 7.4: AMG processor interface

The obtained field of "ghost" cell solution vectors shall be called patch neighbour field (short pnf). The resulting ILU system corresponding to equation (7.7) hence reads:

$$\mathbf{L}_{core,AMG} \cdot \mathbf{U}_{core,AMG} \cdot \mathbf{x}'_{core,AMG} = \mathbf{b}_{AMG} - \mathbf{A}_{core,AMG} \cdot \mathbf{x}^{old}_{core,AMG} - \mathbf{A}_{proc,AMG} \cdot \mathbf{x}^{old}_{pnf} \quad (7.8)$$

Chapter 8

Block Algebraic Multigrid Solver

Contents

The solution procedure for a block coupled linear system of equations, as obtained employing the block coupled algorithm presented in chapter 4, is of utmost importance. Therefore a block algebraic multi-grid linear solver was implemented in this work. In what follows the concept of this solver is presented in detail.

The motivation for using a multi-grid solver is that it is probably the fastest method for solving large sets of linear equations. Thereby this type of linear equation solver exhibits a very good scalability in respect to the number of equations that are to be solved. This property is in general very important for solving large scale problems, and it becomes even more

important when dealing with block matrix systems. For single variable systems N equations are obtained after discretising a computational domain, where N is the number of cells. For block algorithms however, n x N equations are obtained, where n is the number of variables. Hence, for a block algorithm the scaling property of a linear solver is even more crucial to its overall efficiency. Bad scaling would completely annihilate the gain that is obtained from the strong inter-equation coupling. Therefore an additive-correction algebraic multi-grid method for block coupled systems of equations has been implemented in the scope of this work for the proposed block coupled algorithms. This shall be outlined next.

8.1 Additive Correction Block Algebraic Multi-Grid Method

8.1.1 Additive Correction Concept

The concept of the algebraic multi-grid method is very simple to understand, and it can be easily applied to block equation systems. The particular multi-grid technique that has been implemented in the scope of this thesis is called additive correction method (or ACM). It was proposed by Hutchinson and Raithby[27] and it is applied to block equation systems in this work.

The basic idea of each multi-grid method is to solve fields of a specific computational domain on various grids with different grid sizes. The reason for this is that classical iterative methods, such as Jacobi or Gauss Seidel, exhibit a bad overall convergence behaviour, but they actually reduce frequency errors of the size of the numerical grid spacing very rapidly. Hence the idea is solving the same equations on various grids with different spacing, and passing on the information (or solution) from coarser to finer grids [28].

For the ACM method the discretisation is only carried out on the finest grid. Thereof coarse grid equations are generated, summing up coefficients of the fine grid equations, to coarse grid "blocks". The coarse grid cluster addressing for the coarse levels have to be agglomerated beforehand. The adding up of the fine grid's coefficients results in new coefficients, and a new addressing at the next coarser grid level. Since the coarse grid coefficients have been obtained in essence by summing up equations of cells within a cluster, which are integrally conservative, the coarse grid coefficients and therewith the equations for a coarse grid cluster are conservative themselves. The equation on the coarse grids are in fact derived from a correction equation of the respective finer grids. After obtaining a coarse grid solution, this solution is added as a correction to the next finer grid solution[28]. The overall concept is illustrated in Figure 8.1.

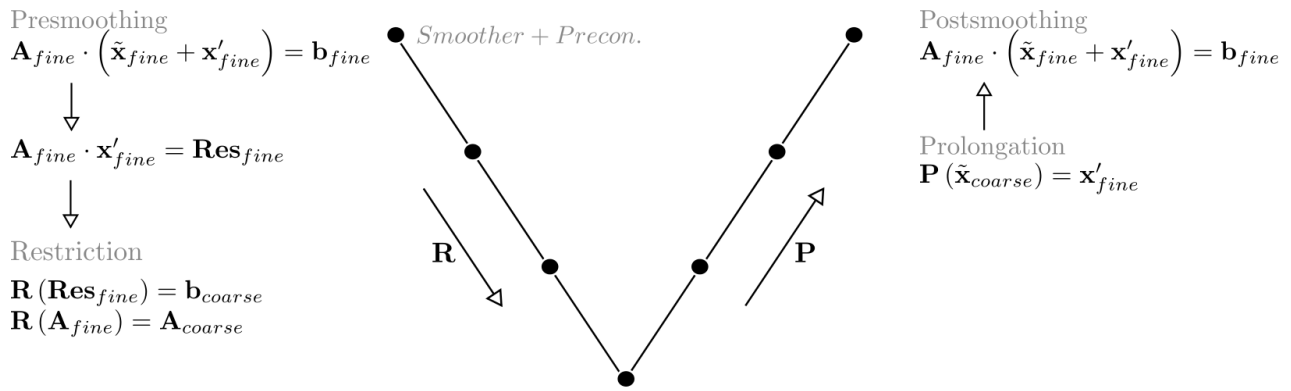

Figure 8.1: Algebraic or Geometric Multi-Grid Agglomeration

Let us now go through the concept of the ACM step by step.

After discretising the PDE(s) on the finest grid, a coefficient matrix $\mathbf{A}_{fine}$, a source vector $\mathbf{b}_{fine}$, as well as an initial guess $\tilde{\mathbf{x}}_{fine}$ for the exact solution of the system $\mathbf{A}_{fine} \cdot \mathbf{x}_{fine} = \mathbf{b}_{fine}$ are obtained.

$$\mathbf{A}_{fine} \cdot \left(\tilde{\mathbf{x}}_{fine} + \mathbf{x}'_{fine}\right) = \mathbf{b}_{fine} \tag{8.1}$$

Where $\mathbf{x}'_{fine}$ is the correction added to the preliminary solution $\tilde{\mathbf{x}}_{fine}$ to obtain the exact solution $\mathbf{x}_{fine}$.

Starting with the initially guessed solution a pre-smoothing step is carried out using an approximative Block ILU(0) smoother (see section 8.3). This will yield a preliminary solution of the fine system which is already closer to its exact solution. The system can then be rewritten in residual (or prime form),

$$\mathbf{A}_{fine} \cdot \mathbf{x}'_{fine} = \mathbf{b}_{fine} - \mathbf{A}_{fine} \cdot \tilde{\mathbf{x}}_{fine} \tag{8.2}$$

$$\mathbf{A}_{fine} \cdot \mathbf{x}'_{fine} = \mathbf{Res}_{fine} \tag{8.3}$$

Now an estimation of $\mathbf{x}'_{fine}$ on a coarser grid system can be calculated. Therefore equation (8.3) is projected to the next coarser grid. The projection is done through the so-called restriction $\mathbf{R}$, which maps the coefficients of $\mathbf{A}_{fine}$ and the residual vector $\mathbf{Res}_{fine}$ to the coarser grid.

$$\begin{aligned} \mathbf{R}\left(\mathbf{A}_{fine}\right) &= \mathbf{A}_{coarse} \\ \mathbf{R}\left(\mathbf{Res}_{fine}\right) &= \mathbf{b}_{coarse} \end{aligned} \tag{8.4}$$

This yields the new system:

$$\mathbf{A}_{coarse} \cdot \left(\tilde{\mathbf{x}}_{coarse} + \mathbf{x}'_{coarse}\right) = \mathbf{b}_{coarse} \tag{8.5}$$

Again smoothing loops are carried out (initializing $\tilde{\mathbf{x}}_{coarse} = 0$) to obtain an approximate solution for $\tilde{\mathbf{x}}_{coarse}$.

Now equation (8.5) can again be written in residual form like in (8.3) and projected to the next coarser grid.

This process can be carried out until the coarsest grid level has been reached, in a cascade-like manner.

Finally the solution for the coarsest grid correction is obtained using a direct solver and LU decomposition. Thereby all coefficients have to be gathered and put into a full linear matrix. This can be difficult when dealing with parallel interfaces. The gathering algorithm shall not be laid out here for the sake of brevity. Once the coarsest level solution has been obtained, this solution can be projected to the next finer level. This so-called prolongation maps the obtained correction of the coarser levels to the correction of the finer levels, which is subsequently added to the preliminary solution of the finer levels to yield a better approximation of it. This is also done from the top to the bottom in a cascade-like fashion. The prolongation is done using a direct zero-order injection of the before calculated corrections.

First the coarse grid solution is prolonged (mapped) to the fine grid correction.

$$\mathbf{P}\left(\tilde{\mathbf{x}}_{coarse}\right) = \mathbf{x}'_{fine} \tag{8.6}$$

Then it is added to the preliminary approximation of $\tilde{\mathbf{x}}_{fine}$.

$$\tilde{\mathbf{x}}_{fine} = \tilde{\mathbf{x}}_{fine} + \mathbf{x}'_{fine} \tag{8.7}$$

Finally post-smoothing loops are carried out to drive $\tilde{\mathbf{x}}_{fine}$ even closer to the accurate solution.

In what follows the restriction and prolongation procedure for the ACM will be outlined. Separate subsections are dedicated to the agglomeration (section 8.2) and smoothing (section 8.3) process.

The outlined restriction and prolongation processes are not limited to so-called V-Cycles as illustrated in Figure 8.1. They can also be easily applied to more sophisticated types of cycles (see section 8.4).

8.1.2 Restriction

The ACM method is based on creating a correction equation, the correction vector of which is supposed to be evaluated on a reduced set of equations which shall represent a "coarser grid". In reality, the set correction equations on a fine grid is only reduced by algebraically adding up some of these equations which should represent a confined set of cells. Assuming that the corrections of all fine grid cells within the same coarse grid cell have approximately the same value, a piecewise constant interpolation ($x'_i = x'_I$).

This assumption along with the process of adding up the matrix coefficients and the source term is commonly called restriction.

Splitting equation (8.1) into a cell's diagonal and neighbour contribution yield a more detailed view of the correction equation. For a particular cell following formulation of equation (8.1) can be written:

$$\mathbf{a}_i \cdot (\tilde{\mathbf{x}}_i + \mathbf{x}'_i) + \sum_{nei(i)} \mathbf{a}_{nei(i)} \cdot (\tilde{\mathbf{x}}_{nei} + \mathbf{x}'_{nei}) = \mathbf{b}_i \tag{8.8}$$

Brought to residual form equation (8.8) reads:

$$\mathbf{a}_i \cdot \mathbf{x}'_i + \sum_{nei(i)} \mathbf{a}_{nei(i)} \cdot \mathbf{x}'_{nei} = \underbrace{\mathbf{b}_i - \mathbf{a}_i \cdot \tilde{\mathbf{x}}_i + \sum_{nei(i)} \mathbf{a}_{nei(i)} \cdot \tilde{\mathbf{x}}_{nei}}_{\mathbf{Res}_i} \tag{8.9}$$

Depending on the agglomeration, various equations will be added up to build a new "clustered" equation. Cluster I contains i equations (or cells). Therefore the coefficients of all of these equations are summed up to obtain corrections for the newly clustered coarse grid equations. While summing up the neighbours nei are divided into inner neighbours nb of a cell i (neighbours that are part of the same cluster) and outer neighbours NB of a cell i, which are not part of the same cluster.

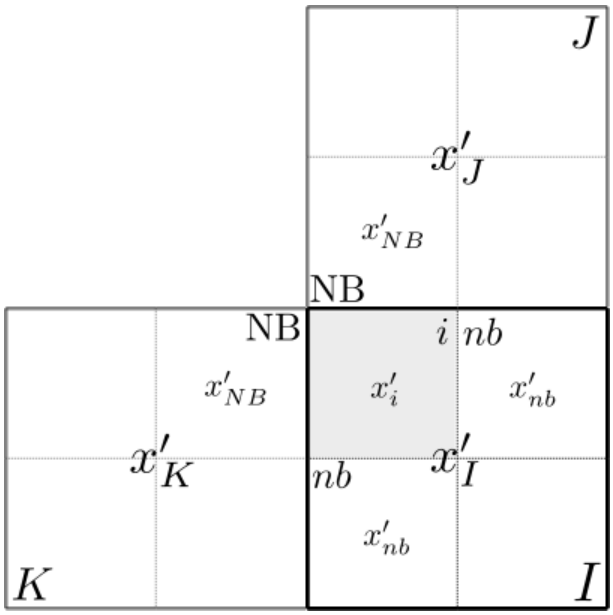

Figure 8.2: ACM correction indexing for restriction

$$\sum_{i(I)} \mathbf{a}_i \cdot \mathbf{x}'_i + \sum_{i(I)} \sum_{nb(i)} \mathbf{a}_{nb} \cdot \mathbf{x}'_{nb} + \sum_{i(I)} \sum_{NB(i)} \mathbf{a}_{NB} \cdot \mathbf{x}'_{NB} = \sum_{i(I)} \mathbf{Res}_i \tag{8.10}$$

Assuming a piecewise constant interpolation, the corrections of all equations within a cluster I are constant. This signifies that $\mathbf{x}'_i = \mathbf{x}'_{nb} = \mathbf{x}'_I$, which leads to the following expression.

$$\sum_{i(I)} \mathbf{a}_i \cdot \mathbf{x}'_I + \sum_{i(I)} \sum_{nb(i)} \mathbf{a}_{nb} \cdot \mathbf{x}'_I + \sum_{i(I)} \sum_{NB(i)} \mathbf{a}_{NB} \cdot \mathbf{x}'_{NB} = \sum_{i(I)} \mathbf{Res}_i \tag{8.11}$$

The first two terms on the LHS contribute to the diagonal block coefficient $\mathbf{a}_I$, whereas the third term constitutes the off-diagonal block coefficients of the new coarse cell neighbours.

$$\mathbf{a}_I = \sum_{i(I)} \mathbf{a}_i + \sum_{i(I)} \sum_{nb(i)} \mathbf{a}_{nb} \tag{8.12}$$

Now the fine grid corrections $x'_{I,fine}$ constitute the new coarse grid variables $\tilde{x}_{I,coarse}$ that shall be solved for in the coarse grid system. After performing some pre-smoothing loops on this system, a new preliminary solution for the coarse grid corrections $\tilde{x}_{I,coarse}$ is found, which can itself be improved by restricting the system to the next coarser grid and solving for the coarse grid corrections $x'_{I,coarse}$. The whole procedure can be carried out until the coarsest level equation system has been reached. Thereafter corrections have to be added (prolonged) successively from coarser to finer levels.

8.1.3 Prolongation

After some post smoothing loops have been carried out on the coarse grid system, the coarse grid solution has to be mapped to the fine grid correction. The interpolation to the fine grid is again done using a piecewise constant interpolation.

$$x'_{i,fine} = x'_{I,fine} = \mathbf{P}(\tilde{x}_{I,coarse}) = \tilde{x}_{I,coarse} \tag{8.13}$$

8.2 Agglomeration

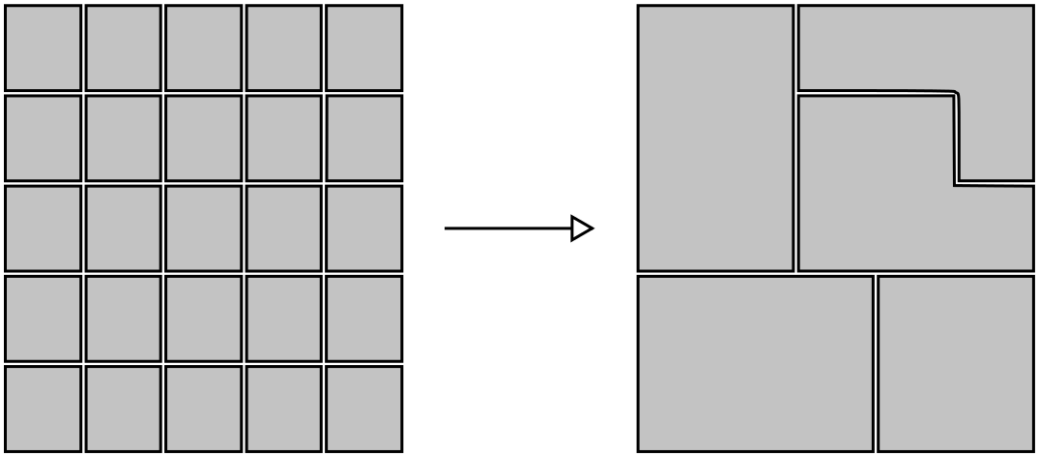

Figure 8.3: Algebraic or Geometric Multigrid Agglomeration

8.2.1 Agglomeration Methods

Agglomeration methods for algebraic multi grid solvers can be divided into two categories:

- Algebraic Agglomeration Methods
- Geometric Agglomeration Methods

While algorithms for geometric agglomeration methods make use of geometric entities such as cell volumes, cell faces, or cluster directionality, algebraic agglomeration focuses on the actual physics of the flow, taking discretised coefficients as agglomeration weights.

Within the scope of this work approximately a dozen agglomeration algorithms, of both geometric and algebraic nature, have been implemented. In general the development of a well functioning agglomeration algorithm is a mix of physically senseful ideas, testing and luck. The level of complexity of such algorithms can rapidly become very high. The most efficient agglomeration method that has been implemented within the scope of this work is based on the algebraic approach of Hutchinson and Raithby [27] [29]. This method has been used for all test cases carried out in section 10. The main concept of agglomeration is based on two rules (following [30]), which are:

RULE 1 "A neighbour of a parent control volume is 'eligible' to be a child if the coefficient for the face between the neighbour and parent is of the same order or larger than the coefficient for the face between the parent and the grandparent."

Where a parent control volume can be seen as the seeing cell for child control volumes, which shall be attached to the parent's control volume cluster. A grandparent control volume is hence the seeding cell of the parent control volume and is also part of the same cluster.

This translates to:

Algorithm 1 Rule 1

1: **for** $neighbourCell\, j = 1 \rightarrow nNeighboursOf(i)$ **do**
2: **if** $max(a_{ij}, a_{ji}) \geq \frac{1}{2} max(a_{ih}, a_{hi})$ **then**
3: cell j is child of i
4: **end if**
5: **end for**

Where h is the grandparent index.

RULE 2 "An eligible child can become a child (be agglomerated with the parent) if the coefficient between the parent and the eligible child is of the same order or larger than the maximum coefficient between the eligible child and its other neighbours"

This translates to:

Algorithm 2 Rule 2

1: **for** $neighbourCell\, j = 1 \rightarrow nNeighboursOf(k)$ **do**
2: **if** $j \neq i$ AND $max(a_{ij}, a_{ji}) \geq \frac{1}{2} max(a_{ik}, a_{ki})$ **then**
3: cell j is child of i
4: **end if**
5: **end for**

Where the parent index is i

Elias [30] gives even additional guidelines for the proper implementation of such an algorithm.

8.3 Block ILU(0) Smoother

The pre- and post-smoothing loops within the multi-grid solver could be carried out with any iterative solver. However it is important that the smoothing operation acts as efficiently as possible since it accounts for the by far biggest computational part within the proposed method. It is essential that the smoothing operation itself can be carried out fast, and that the residual is dropped rapidly and in a stable manner. Often standard iterative solvers, such as Jacobi or Gauss Seidel are used, but also Krylov subspace solvers such as GMRES. Many papers show however, that the most efficient smoothers for many multi-grid algorithms are ILU (incomplete lower upper) smoothers. Since ILU smoothers are incomplete solvers, it is not possible to find an exact solution with them. However, if the linear system is reformulated, and written in correction form $\mathbf{A} \cdot \mathbf{x}' = \mathbf{Res}$, it is only important to decrease the correction $\mathbf{x}'$ gradually, which ILU Smoothers manage to do for a wide range of numerical problems.

LU Solver

Factorization

It is possible to decompose a matrix $\mathbf{A}$ into the dot product a lower triangular matrix $\mathbf{L}$ and a upper triangular matrix $\mathbf{U}$ of the form,

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \tag{8.14}$$

where the diagonal coefficients of the upper triangular matrix are equal to 1.

$$\mathbf{L} = \begin{bmatrix} l_{00} & 0 & 0 \\ l_{10} & l_{11} & 0 \\ l_{20} & l_{21} & l_{22} \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} 1 & u_{01} & u_{02} \\ 0 & 1 & u_{12} \\ 0 & 0 & 1 \end{bmatrix} \tag{8.15}$$

Since it is known beforehand that the diagonal coefficients of the upper triangular matrix are

equal to 1, the coefficients of the both upper and lower matrix can be stored in one single coefficient matrix.

The factorization (decomposition) algorithm can be derived from the Gauss elimination procedure and reads:

Algorithm 3 LU Factorization

```
1: for i = 1 → n do
2:     for k = 0 → i − 1 do
3:         if a_kk ≠ 0 then
4:             a_ik ← a_ik/a_kk
5:             for j = k + 1 → n do
6:                 if a_ij ≠ 0 then
7:                     a_ij ← a_ij − a_ik ∗ a_kj
8:                 end if
9:             end for
10:        end if
11:    end for
12: end for
```

Forward and backward substitution

After decomposing the matrix $\mathbf{A}$ into an upper matrix $\mathbf{U}$ and a lower matrix $\mathbf{L}$, the solution of the equation system can be obtained by subsequent forward and backward substitution.

$$\mathbf{L} \cdot \mathbf{U} \cdot \mathbf{x} = \mathbf{b} \tag{8.16}$$

We can write.

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{Y} \tag{8.17}$$

Inserting equation (8.17) in equation(8.16) we obtain.

$$\mathbf{L} \cdot \mathbf{Y} = \mathbf{b} \tag{8.18}$$

Now we can easily evaluate $\mathbf{Y}$ through forward substitution in equation(8.18), and subsequently

we obtain the solution vector $\mathbf{x}$ through backward substitution in equation (8.17).

Algorithm 4 LU forward substitution

```
1: for i = 0 → n do
2:     Y(i) ← b(i)
3:     for j = 0 → i − 1 do
4:         Y(i) ← Y(i) − a_ij · Y(j)
5:     end for
6: end for
```

Algorithm 5 LU backward substitution

```
1: for i = n → 0 do
2:     x(i) ← 1/a_ii · Y(i)
3:     for j = i + 1 → n do
4:         x(i) ← x(i) − a_ij/a_ii · x(j)
5:     end for
6: end for
```

ILU(0) Solver

For very large systems of equations the LU decomposition turns out to be too expensive, since its factorization takes $O(n^3)$ computational time. Therefore for sparse matrices it can make sense to calculate an approximate solution, for which the computational time is far less (e.g. $O(n)$). This can be achieved by modifying the factorization of the LU algorithm. The dot product of the obtained lower and upper matrix will then be equal to the original matrix plus an error term $\mathbf{R}$ that will be neglected.

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} + \mathbf{R} \tag{8.19}$$

Factorization

There are various possible types of factorizations to obtain an incomplete LU decomposition. A very powerful one is the ILU(0) factorization. This factorization adapts the original LU decomposition algorithm in a way that preserves the sparsity pattern of the original matrix. This results in a very fast decomposition process, that it very favourable in terms of memory requirements. Algorithm 6 shows a pseudo code of such a decomposition.

Algorithm 6 ILU(0) Factorisation

```
1: for i = 1 → n do
2:     for k = 0 → i − 1 do
3:         if a_ik ≠ 0 and a_kk ≠ 0 then
4:             a_ik ← a_ik/a_kk
5:             for j = k + 1 → n do
6:                 if a_ij ≠ 0 then
7:                     a_ij ← a_ij − a_ik ∗ a_kj
8:                 end if
9:             end for
10:        end if
11:    end for
12: end for
```

Forward and backward substitution

The forward and backward substitution steps are the same as in algorithms (4) and (5).

Modified Block ILU(0) Solver

As for the scalar ILU(0) decomposition the existing sparse matrix structure shall be conserved. Since the OpenFOAM library uses an lower/upper addressing based on faces shared by two adjacent cells, the matrix structure is symmetric. This means that for each face the matrix' row and column position for the lower, upper and diagonal coefficients are known. However due to addressing structure it is very expensive to gather information about e.g. the existence of further coefficients that might be eligible for the ILU decomposition process. An example for this is given in Figure 8.4, where a matrix structure constructed from four cells is illustrated. There the interaction of block coefficients (3,2) and (0,2) should be taken into account in the original ILU(0) factorisation process. However, the OpenFOAM structure doesn't provide the information that these two cells are in the same row. Therefore the ILU(0) algorithm has been modified, and coefficient couples like the one described are not taken into account. In any case, these couples rarely occur in large matrix systems arising from our discretisation process, and therefore the resulting modified ILU(0) factorisation will lead to an almost identical decomposition as the original ILU(0) decomposition.

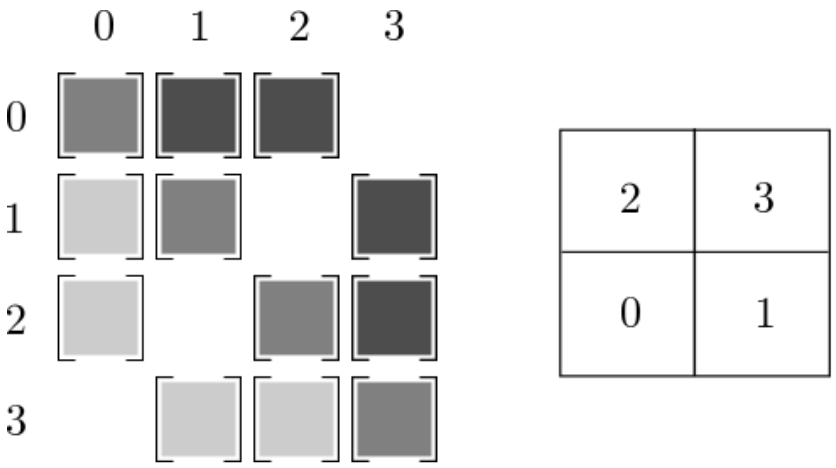

Figure 8.4: Block Matrix Structure - light grey, lower block coeffs - grey, diagonal block coeffs - dark grey, upper block coeffs

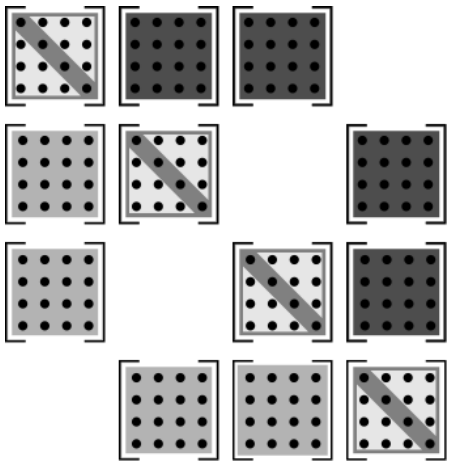

Figure 8.5: Block Matrix Structure - light grey, lower block coeffs - grey, diagonal block coeffs - dark grey, upper block coeffs

Other than the symmetry of the block coefficients, we can also use the quadratic structure of the block coefficients themselves to quickly address scalar coefficients for the modified ILU(0) factorisation. In algorithm (6) it can be seen that in line 7 coefficient a_{kj} is requested for the factorisation. This coefficient can either be zero (or not filled) or it can be filled. However, as it would be costly in our framework to gather the information if this coefficient is actually filled, we will not perform an information-gathering search algorithm. Instead we shall limit our ILU facorisation on coefficients a_{kj} of which we know that they are filled. There are three cases for which we know the existence and the addressing of coefficient a_{kj}.

The existence of a_{kj} is always known if:

- a_{ik} and a_{ij} are in the same lower block coefficient.
- a_{ij} is part of a diagonal block coefficient.
- a_{ik} is part of the lower triangle of a diagonal block coefficient.

Thereof results the following modified ILU(0) factorisation algorithm:

Algorithm 7 Modified Block ILU(0) Factorisation

```
1: for I = 1 → N do
2:     for i = 0 → n do
3:         for K = 0 → I − 1 do
4:             if A_IK is known and A_II is known and A_KK is known then
5:                 for k = 0 → k < n do
6:                     A_IK(i,k) ← A_IK(i,k)/A_KK(k,k)
7:                     for j = k + 1 → n do
8:                         A_IK(i,j) ← A_IK(i,j) − A_IK(i,k) ∗ A_KK(k,j)
9:                     end for
10:                    for j = 0 → n do
11:                        A_II(i,j) ← A_II(i,j) − A_IK(i,k) ∗ A_KI(k,j)
12:                    end for
13:                end for
14:            end if
15:        end for
16:        for k = 0 → k < i do
17:            A_II(i,k) ← A_II(i,k)/A_II(k,k)
18:            for j = k + 1 → n do
19:                A_II(i,j) ← A_II(i,j) − A_II(i,k)/A_II(k,j)
20:            end for
21:            for J = I + 1 → N do
22:                if A_IJ is known and A_II is known then
23:                    for j = 0 → n do
24:                        A_II(i,k) ← A_II(i,k)/A_II(k,k)
25:                        for j = k + 1 → n do
26:                            A_IJ(i,j) ← A_II(i,k)/A_IJ(k,j)
27:                        end for
28:                    end for
29:                end if
30:            end for
31:        end for
32:    end for
33: end for
```

Forward and backward substitution

The algorithms for forward and backward substitution in block equation systems are written below.

Algorithm 8 Modified Block ILU(0) forward substitution

for $I = 0 \to N$ **do**
  for $i = 0 \to n$ **do**
    $Y_I(i) \leftarrow b_I(i)$
    for $j = 0 \to i - 1$ **do**
      $Y_I(i) \leftarrow Y_I(i) - A_{II}(i,j) \cdot Y_I(j)$
    end for
    for $J = 0 \to J < I$ **do**
      for $j = 0 \to n$ **do**
        $Y_I(i) \leftarrow Y_I(i) - A_{IJ}(i,j) \cdot Y_J(j)$
      end for
    end for
  end for
end for

Algorithm 9 Modified Block ILU(0) backward substitution

for $I = N \to 0$ **do**
  for $i = n \to 0$ **do**
    $X_I(i) \leftarrow \frac{1}{A_{II}(i,i)} Y_I(i)$
    for $j = i + 1 \to n$ **do**
      $X_I(i) \leftarrow X_I(i) - \frac{A_{II}(i,j)}{A_{II}(i,i)} \cdot X_I(j)$
    end for
    for $J = I + 1 \to N$ **do**
      for $j = 0 \to n$ **do**
        $X_I(i) \leftarrow X_I(i) - \frac{A_{IJ}(i,j)}{A_{II}(i,i)} \cdot X_J(j)$
      end for
    end for
  end for
end for

8.4 Block AMG cycles

The outlined restriction and prolongation processes are not limited to so-called V-Cycles as illustrated in Figure 8.1. They can also be easily applied to F- or W-Cycles.

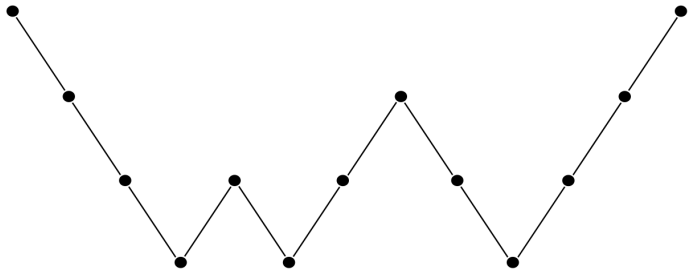

Figure 8.6: F-Cycle

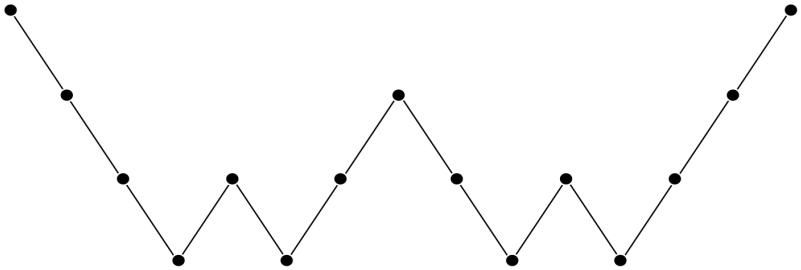

Figure 8.7: W-Cycle

Chapter 9

Turbulence Models for Unsteady Computations

Contents

The most important aim of current thesis was to model flow physics as accurate as possible to be able to correctly predict instabilities of flows in hydraulic turbomachines. To this end the author has done extensive literature studies to find out which state-of-the art turbulence models would give the best compromise between computational cost and accuracy, keeping in mind that ultimately hydraulic designers should use the resulting code to develop hydraulic machines.

The design process usually imposes restrictions in terms of time but also in terms of computational capacity, limiting the choice of turbulence models to those that could do the job accurately enough within a reasonable amount of time. Taking these considerations into account, the author opted for Menter's k-Omega SST model [31] as turbulence models of choice. However, the author believes that it would be worthwhile trying a lot of other turbulence models, such as the k-Omega SST/SAS model [32], Reynolds stress models([33]), the v2f model([34]), or even DES models([35]). The reasons why this hasn't been done in this work was that there wasn't enough

time left to do so, but also that some of the mentioned models require very small time steps and thus computational time to capture the flow phenomena accurately. For very small time steps segregated algorithms may however be more suitable than block coupled ones, because they will probably outperform them in terms of computation time and memory requirement.

For the outlined pressure-based coupled algorithm the turbulence models are solved after the updated solution of the pressure-velocity system is obtained. A dynamic turbulent viscosity field ν_t is obtained by solving the turbulence equations. Subsequently, this turbulent viscosity will be added to the laminar viscosity ν to yield an effective viscosity ν_{eff} for the diffusion coefficients in the momentum equations. For steady state simulations this loose coupling will finally result in a converged solution. For transient simulations the non-linearity introduced by the treatment of the turbulence equations should also be accounted for by sub-looping (loose coupling) for each time step.

9.1 The k-Omega SST (U)RANS Model

As closure model for the turbulent fluctuations an eddy viscosity model, the well known $k-\omega$ Shear Stress Transport - SST model, has been considered. This model has been implemented by Luca Mangani and used for all test cases in section(10). The model, originally proposed by Menter [31], is based on a modification to the standard $k-\omega$ model, which overcomes its sensitivity to quite arbitrary free-stream values of ω.

In the sub- and logarithmic boundary layer region the model adopted is the standard $k-\omega$ model, because of its robustness and the non-existence of a damping function and Dirichlet type boundary conditions. For the rest of the domain the standard $k-\epsilon$ model, written in terms of ω, has been preferred due to its good ability in predicting different kinds of flows. The steady-state transport equations for k and ω are outlined below.

The k equation reads,

$$\frac{\partial k}{\partial t} + \nabla \cdot (\mathbf{u}k) - \nabla \cdot [(\nu + \nu_t \alpha_K)\nabla k] = P_k - \beta^* \omega k \tag{9.1}$$

The omega equation reads,

$$\frac{\partial \omega}{\partial t} + \nabla \cdot (\mathbf{u}\omega) - \nabla \cdot [(\nu + \nu_t \alpha_V)\nabla \omega] = \frac{C_1 P_k}{\nu_t} - C_2 \omega^2 + \frac{2\alpha_\varepsilon (1 - F_1)}{\omega} \nabla k \cdot \nabla \omega \tag{9.2}$$

The resulting turbulent dynamic viscosity ν_t which subsequently has to be added to the laminar kinematic viscosity ν to yield the effective kinematic viscosity ν_{eff} is given by,

$$\nu_t = \frac{c_\mu k}{max(c_\mu \omega, SF_2)} \tag{9.3}$$

The original model is based on a Low Reynolds formulation and therefore it requires a fine mesh resolution at walls in order to satisfy the necessary condition $y^+ \approx 1$. In flow simulations of a certain complexity in terms of geometry and fluid structures this requirement leads to a

strict constraint. Therefore, in order to increase grid independence a mixed approach between wall-function and Low-Reynolds approach was added to the $k-\omega$ SST model. The idea (see [31]) is to blend the two approaches via a blending function Γ, which is calculated algebraically from the non-dimensional wall distance. Both turbulent production and turbulent specific dissipation are imposed on the first node mixing the effects of the Low Reynolds and High Reynolds contributions:

$$P_{awt,p} = P_{LR,p}e^{-\Gamma} + P_{HR,p}e^{-\frac{1}{\Gamma}}, \tag{9.4}$$

$$\omega_{awt,p} = \omega_{LR,p}e^{-\Gamma} + \omega_{HR,p}e^{-\frac{1}{\Gamma}}, \tag{9.5}$$

where Γ is calculated with an algebraic expression for y^+ defined with $u_\tau = max(u_{\tau,LR}, u_{\tau,HR})$. A blending function is applied following Kader's universal law [36].

Chapter 10

Case Studies

Contents

The case studies in this chapter shall demonstrate the performance of the coupled approach outlined in chapter 4 in terms of convergence speed, mesh size scalability and robustness by comparing results to those obtained with a state-of-the-art segregated SIMPLE-C benchmark solver by Mangani [37]. The first three test cases, namely the *Backward facing step*, *Draft tube* and *Pelton distributor* represent industrial-size steady simulations in stationary reference frame, whose results have already been published in Mangani et al. [38], and are taken thereof.

The next three test cases, namely the *Pump runner*, *Francis turbine runner* and *Kaplan turbine runner* represent industrial-size steady simulations in rotational reference frames. The results have been taken from Mangani et al. [39]. As closure model for turbulent fluctuations in all simulations Menter's $k-\omega$ SST model was used as outlined in section 9.1.

The performance of all steady calculations has been evaluated at a defined RMS (equation 10.1) convergence threshold of $1e-5$.

$$RMS(\phi) = \frac{\sqrt{\frac{1}{N}\sum_{i=0}^{N}\left(res(\phi(i))/a_C^{\phi\phi}\right)^2}}{max(\phi,0) - min(\phi,0)} \tag{10.1}$$

Mesh size scaling property was evaluated on some test cases using a normalized scale factor (equation (10.2)).

$$Scale\,Factor = \frac{(time/c.v.)_{nCells}}{(time/c.v.)_{nCells,ref}} \tag{10.2}$$

The scale factor thus describes ratio between the calculation time per control volume given a certain mesh refinement (nCells) and the calculation time per control volume given a reference mesh refinement (nCells,ref).

All steady simulations were carried out using a first order upwind discretisation for the convection term.

In addition to the steady simulations in stationary and rotational frame a transient simulation of the flow in a centrifugal pump has been carried out in section 10.3, using the $k-omega$ SST model and a second order vanLeer scheme (section A) for the convection term. Obtained results have been compared to PIV measurements to prove the accuracy of proposed method.

10.1 Stationary frame of reference

10.1.1 Backward facing step

The *Backward facing step* test case was chosen to demonstrate the good scalability of the outlined solver in respect to the number of grid cells. The geometry of the test case can be seen in Figure 10.2. The test case has been carried out with 3 different grid sizes. For all grid sizes the same flow field has been obtained for both the block coupled and the segregated algorithm. The difference in performance and scalability, evaluated at the defined RMS convergence threshold, is outlined in Table 10.1.

Table 10.1: *Backward facing step* - Performance comparison of the coupled (C) and segregated (S) algorithm

# CV	(C)time[s]	(C)time/CV[s]	(S)time[s]	(S)time/CV[s]	S/C
12k	4.3	0.000351	32.5	0.002653	7.6
48k	24.1	0.000493	393.7	0.008051	16.3
195k	139.9	0.000715	5888.0	0.030102	42.1

From Table 10.1 it can be clearly seen that the block coupled algorithm outperforms the segregated algorithm in terms of convergence speed. More importantly the *Backward facing step* test case proves the superiority of the coupled solver over the segregated solver in respect to scalability with increasing grid size. In Figure 10.1 a scaling factor is plotted as a function of the grid size. It can be seen that the coupled solver scales almost linearly, whereas the segregated solver's convergence behaviour deteriorates a lot with increasing mesh sizes.

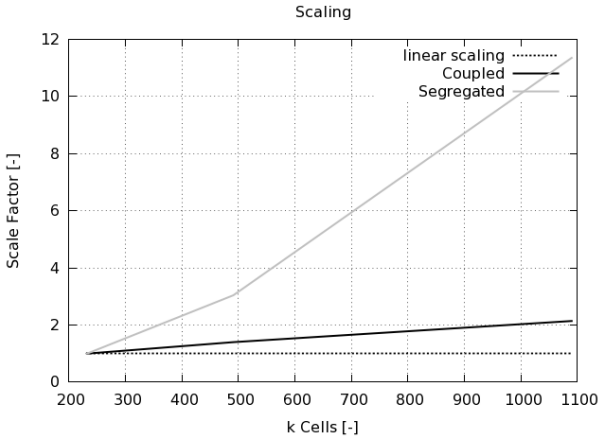

Figure 10.1: Backward facing step - mesh size scaling

Figure 10.2 compares the velocity profiles at the indicated position of the coupled and the segregated approach. The small difference of the flow field is related to a slightly different boundary treatment.

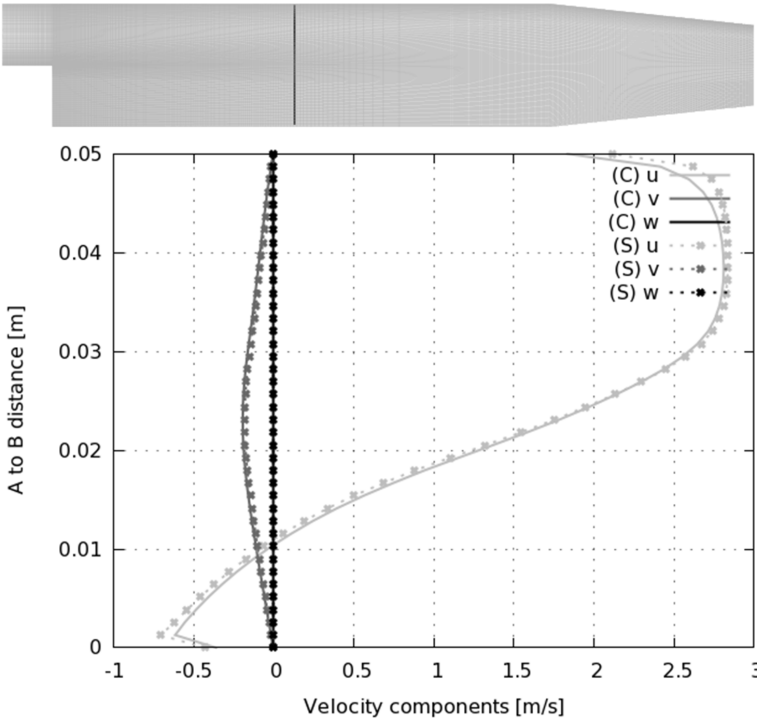

Figure 10.2: Velocity profiles. Segregated(S) - dotted lines, coupled(C) - full lines

In respect to convergence behaviour the coupled solver shows a smooth and steady convergence with a very good convergence rate. In Figure 10.3 it can be seen that the segregated solver

on the other hand converges slowly and its convergence shows fluctuating behaviour. These fluctuations of the outer iterations, which arise from the weak variable coupling, are considered to be a sign of bad robustness.

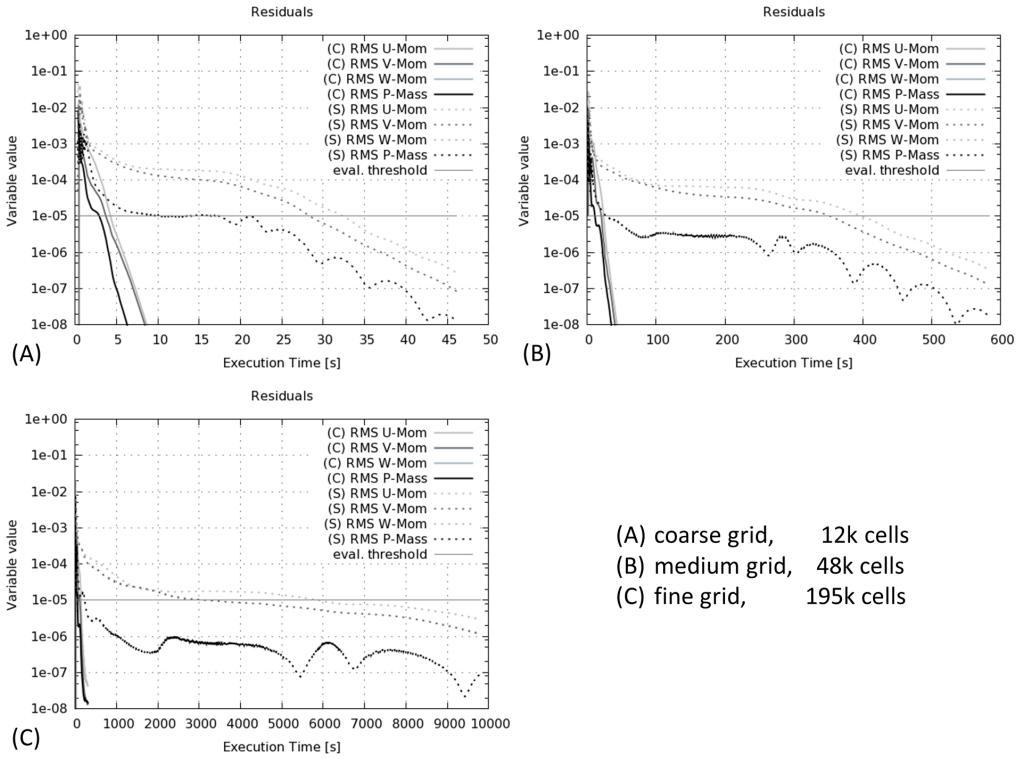

Figure 10.3: Convergence histories for *Backward facing step* test case. Segregated(S) - dotted lines, coupled(C) - full lines

10.1.2 Draft tube

Draft tubes are very huge constructional elements that are placed behind hydraulic turbines in order to minimize efficiency losses at the turbine runner outlet by decreasing there the static pressure using a diffuser. Hence, the *Draft tube* test case is a particularly difficult test case, because of its diffuser characteristic, which leads to flow detachment at its separation pier. The geometry, shown in Figure 10.4, has sharp edges and the mesh contains highly skewed cells at the butt of the pier. At the inlet a swirling flow is prescribed, meaning that not only a non-uniform axial velocity field is prescribed, but also a circumferential velocity field that accounts for the pre-swirl generated by a thought turbine runner. For the pressure a zero gradient boundary condition is set at the inlet. The turbulence quantities at the inlet are chosen to be constant for the sake of simplicity. At the outlet a zero gradient condition is used for the velocity and turbulence quantities and a constant field is applied for the pressure. At the walls blended wall functions are used to evaluate the shear stress accordingly.

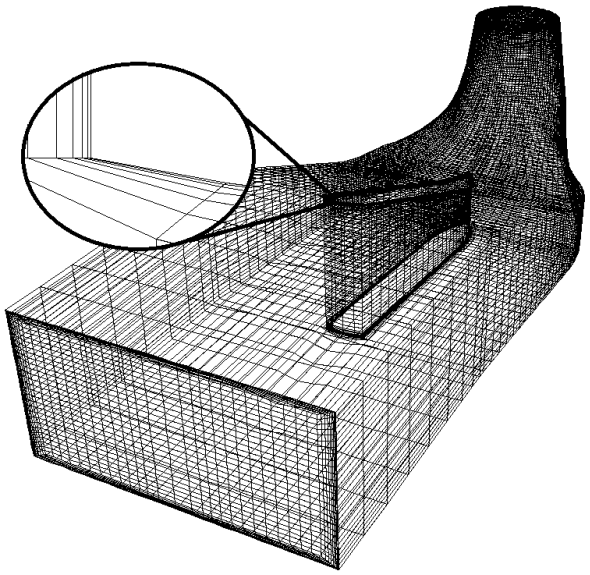

Figure 10.4: Computational grid of *Draft tube* test case

The difference in performance and scalability is outlined in Table 10.2, evaluated at the defined RMS convergence threshold. The segregated method seems not to attain this convergence level for the finest grid, or only very slowly. Table 10.2 indicates that the coupled solver

converges much faster than its segregated counterpart and that the segregated solver is being outperformed in terms of scaling.

Table 10.2: *Draft tube* - Performance comparison of the coupled (C) and segregated (S) algorithm

# CV	(C)time[s]	(C)time/CV[s]	(S)time[s]	(S)time/CV[s]	S/C
232k	535.9	0.002304	1342.2	0.005771	2.50
491k	988.6	0.002013	3656.7	0.007447	3.70
1090k	1997.0	0.001831	x	x	x

Astonishingly the mesh size scalability is even sub-linear for the *Draft tube* test case (see Figure 10.5). The reason for this over-performance is due to extremely skewed cells in the butt region of the pier (see Figure 10.4). Since the quality of the mesh is increased with an increasing number of cells, the convergence rate also seems to increase for finer meshes.

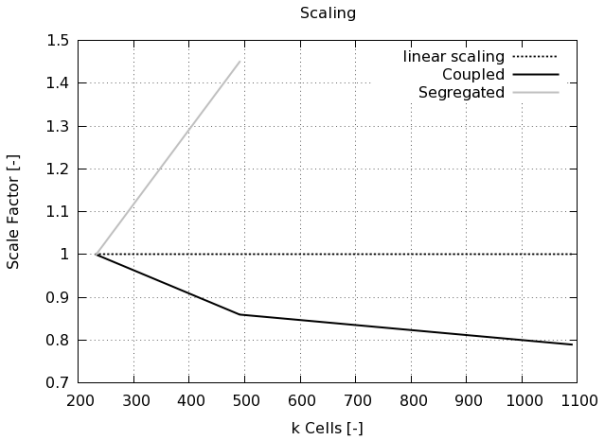

Figure 10.5: *Draft tube* - mesh size scaling

Figure 10.6 illustrates the good performance of the coupled approach in respect to the segregated approach. For the fine mesh configuration the segregated solver's convergence rate almost stalls.

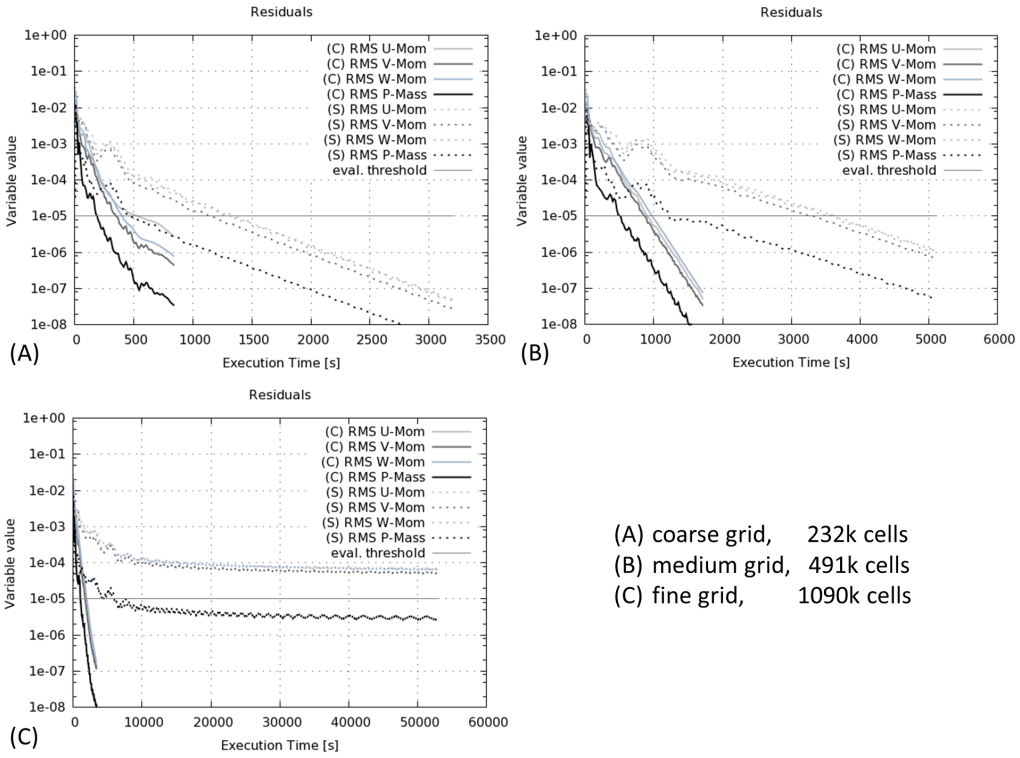

Figure 10.6: Convergence histories for *Draft tube* test case. Segregated(S) - dotted lines, coupled(C) - full lines

The velocity contour plot of a slice through the draft tube shows very similar flow patterns. The differences again arise from the alternate boundary treatment which leads to slightly different detachment positions.

Figure 10.7: Velocity contour plot of *Draft tube* test case. (S) left, (C) right

10.1.3 Pelton distributor

The function of a Pelton turbine distributor is to distribute water coming from a high altitude basin to a couple of injector nozzles that shall continuously apply high-speed jets of water to a Pelton turbine runner. The flow in Pelton turbine distributors is very demanding for CFD applications, because plenty of detachment zones exist, that can lead to difficulties in convergence for steady state flow. I.e. there are flow detachment areas at the beginning of the distributor's injectors and furthermore the grid contains highly skewed cells at the bifurcations. The *Pelton Distributor* test case is evaluated with two different mesh sizes in order to investigate the mesh size scaling properties of the coupled solver. The mesh sizes are computationally quite demanding (with 2.7 and 6.3 mio. cells respectively).

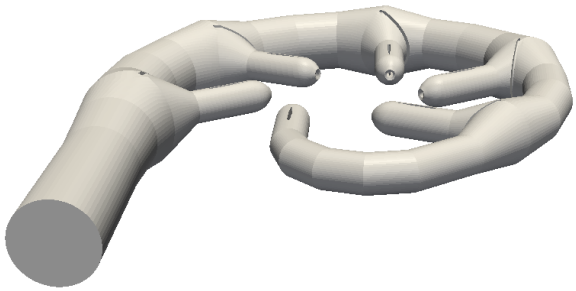

Figure 10.8: *Pelton Distributor*

The difference in performance in respect to the benchmark solver by Mangani [37] is outlined in Table 10.3, evaluated at the defined RMS convergence threshold.

Table 10.3: *Pelton Distributor* - Mesh size scaling of the coupled (C) algorithm

# CV	(C)time[s]	(C)time/CV[s]
2726k	2932	0.001075
6253k	6068	0.000970

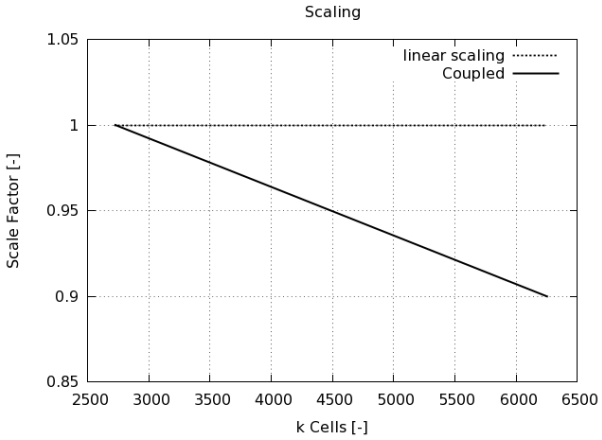

Figure 10.9: Pelton Distributor - mesh size scaling

Table 10.3 and Figure 10.9 demonstrate again that the coupled solver scales excellently with increasing numbers of cells, even for very big meshes. As it has been the case for the *Draft tube* (section 10.1.2), the coupled solver even over-performs since the time per control volume for the bigger mesh case is even smaller than for the smaller mesh case. However for the fine grid case, the coupled algorithm experiences a slowdown in convergence rate after passing the 1e-5 threshold of the RMS residuals.

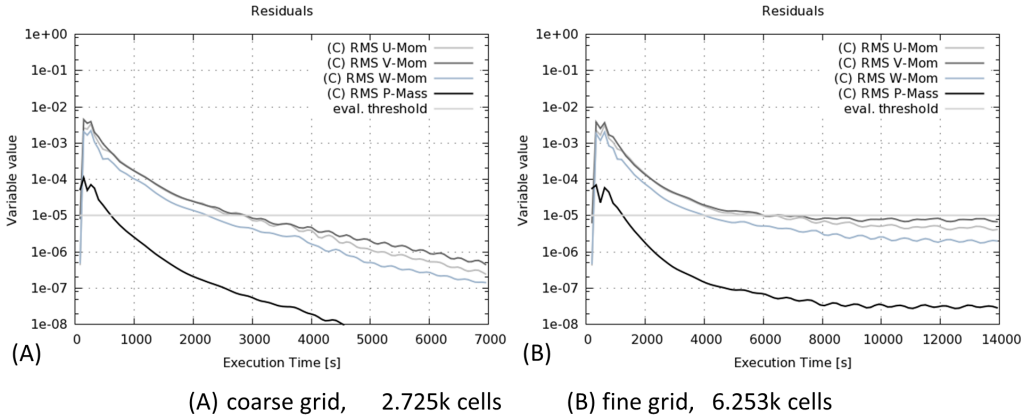

Figure 10.10: Convergence history for *Pelton distributor* test case (2.726 k cells)

10.2 Rotational frame of reference

10.2.1 Pump runner

The set-up for the pump runner test case consists of a single blade channel with cyclic AMI interfaces (section 7.1). Hub, shroud and blade are moving with a prescribed absolute velocity, so the shear stress at these patches is treated as outlined in section (6.4). For these walls a zero gradient boundary condition is prescribed for the pressure while the blended wall functions are used for turbulence quantities. At inlet velocity and turbulence quantities are specified, while a zero gradient boundary condition is applied to the pressure. For the outlet a Neumann boundary condition is applied to the velocity and turbulence quantities, while a specified value condition is used for the pressure field.

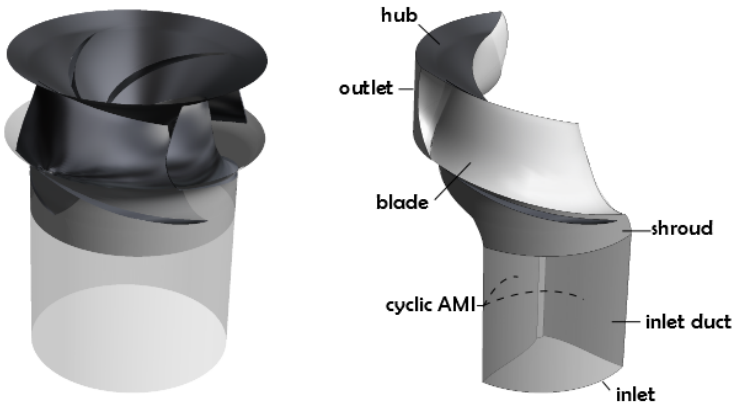

Figure 10.11: *Pump* configuration

The investigated operating point is close to the optimal operating point, and almost no secondary flow occurs inside the runner channel. The pump has a medium specific speed characteristic, and its relatively high deceleration rate makes it an aggressive diffuser.

For validation purpose the velocity profiles evaluated from hub to shroud at the trailing edge computed with the coupled solver were compared to those computed with the segregated benchmark solver. The profiles in Figure 10.12 show good conformance.

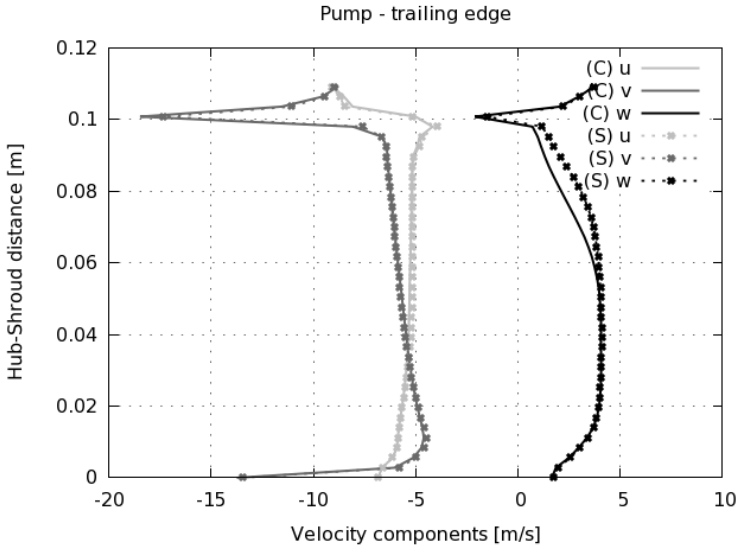

Figure 10.12: *Pump* - velocity profiles close to trailing edge. Segregated(S) - dotted lines, coupled(C) - full lines

To evaluate the performance and scalability three different mesh sizes with increasing numbers of cells are used. Results in Table 10.4 clearly show that the coupled solver scales near linearly with mesh size while the segregated solver suffers a bit as the number of elements increases. The convergence characteristics can be seen in Table 10.4 and in Figure 10.13.

Table 10.4: *Pump* - Performance comparison of the coupled (C) and segregated (S) algorithm

# CV	(C)time[s]	(C)time/CV[s]	(S)time[s]	(S)time/CV[s]	S/C
309k	652.4	0.002111	x	x	x
802k	2399.2	0.002991	20351.7	0.025250	8.48
1260k	3144.5	0.002495	36360.9	0.028858	11.56

It is worth noting that the segregated solver did not converge to the desired level on the coarsest grid. In all runs the coupled solver substantially outperformed the segregated solver.

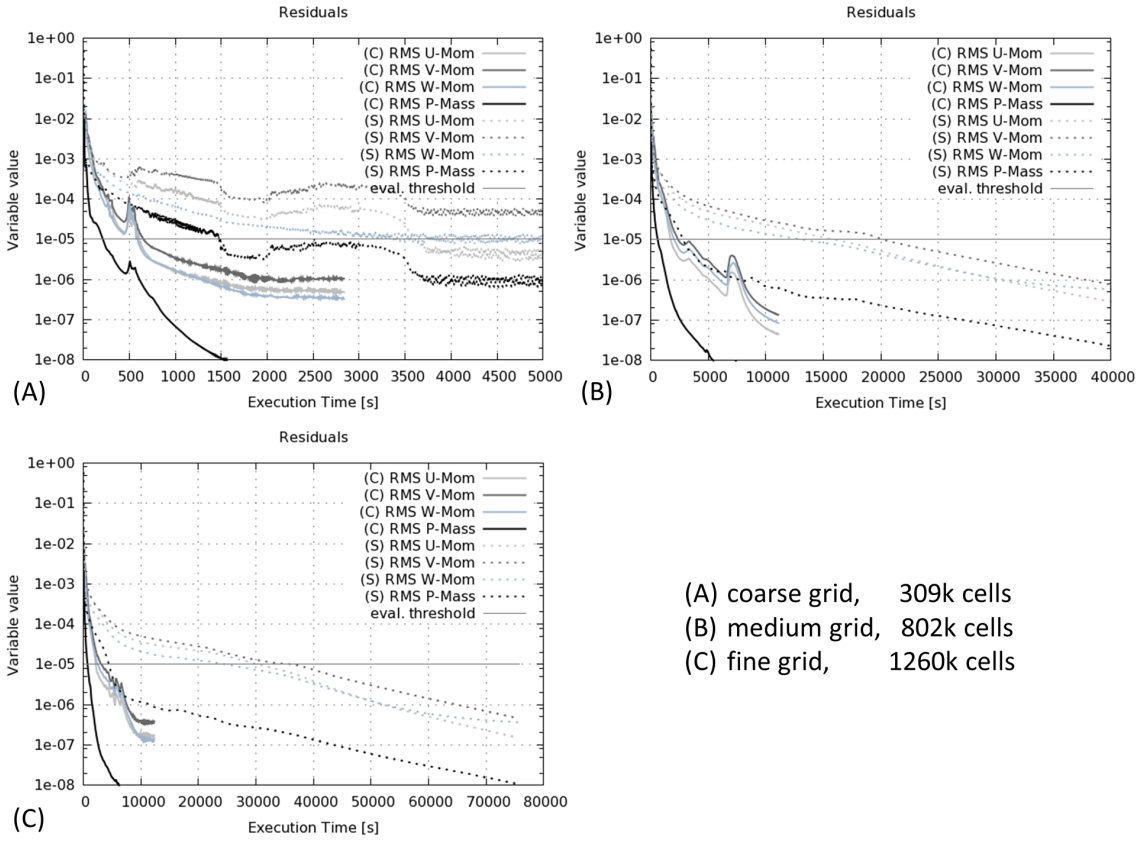

Figure 10.13: Convergence histories for *Pump* test case. Segregated(S) - dotted lines, coupled(C) - full lines

10.2.2 Francis turbine runner

The Francis turbine test case's set-up is somewhat similar to the pump runner's setup, since all boundary conditions are applied in the same way, the only difference being the direction of the flow. Again only a single blade channel is modelled using a cyclic AMI interface. The turbine has a relatively high-head low-debit characteristic, and the flow is being accelerated rather than decelerated (as it was the case for the pump runner). This fact should a priori improve convergence behaviour in respect to the pump.

Figure 10.14: *Pump* configuration

Velocity profiles evaluated from hub to shroud at the trailing edge again show good conformance in respect to the segregated approach (see Figure 10.15). Small discrepancies for the u component of the velocity are very probable due to a small difference in the turbulence treatment especially at the boundary.

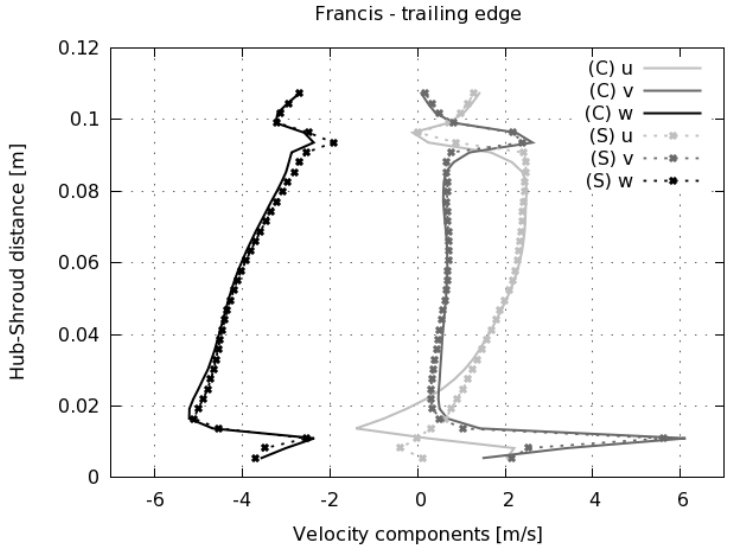

Figure 10.15: *Fancis* - velocity profiles close to trailing edge

In terms of mesh size scalability Table 10.5 indicates again that the coupled solver scales almost linearly, whereas the segregated solver does not. The runtime until a specified evaluation threshold was reached can be seen in Table 10.5 as well as in Figure 10.17.

Table 10.5: *Francis* - Performance comparison of the coupled (C) and segregated (S) algorithm

# CV	(C)time[s]	(C)time/CV[s]	(S)time[s]	(S)time/CV[s]	S/C
205k	136.2	0.000665	2568.9	0.012543	18.86
566k	411.4	0.000726	14442.5	0.025499	35.11
1026k	1109.8	0.001082	34936.7	0.034051	31.48

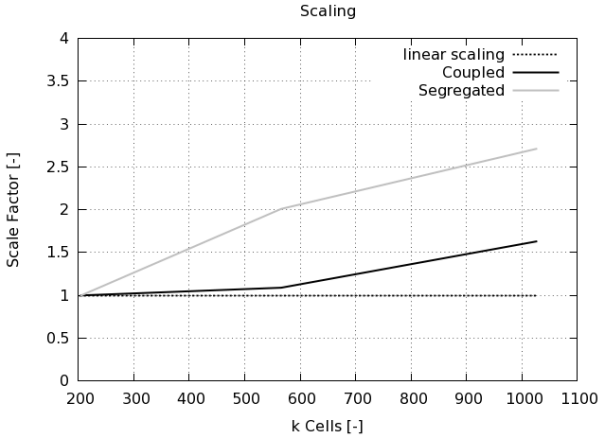

Figure 10.16: *Francis* - mesh size scaling

Figure 10.17 shows that the coupled solver outperforms the segregated one in terms of convergence speed and scalability. However its convergence stalls quite early hinting at potential for improvement. The reason for the plateauing of the convergence is due to oscillations of the algorithm's outer iterations, which can be caused by bad mesh quality, increasing the weight of the explicit non-orthogonal correction of Laplacian terms in momentum, pressure and turbulence equations. Although it is difficult at this stage to point down the actual origin of this problem, it is thought to be related to the implementation of the turbulence model, which could still be improved.

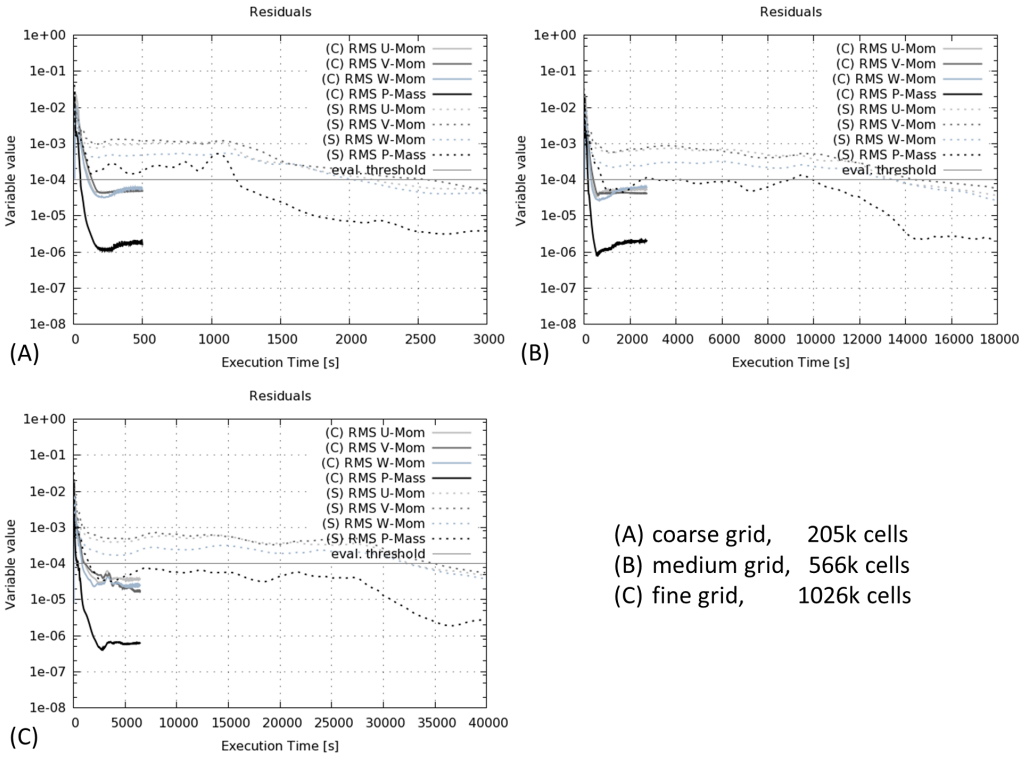

Figure 10.17: Convergence histories for *Francis* test case. Segregated(S) - dotted lines, coupled(C) - full lines

10.2.3 Kaplan turbine runner

A last turbomachinery test case shall be shown to demonstrate the good functioning of proposed method. It is a Kaplan turbine whose mesh consists of several blocks. This is because for this type of turbine gaps exist between the hub and the inner part of the blades, as well as between the shroud and the outer part of the blades. These gaps are meshed separately and the meshes are subsequently connected to the main part of the computational domain which comprises a circumferential horizontal intake, the blade region and the outlet region. The gaps and the main part of the mesh are connected with non-conformal AMI interfaces (as outlined in section 7.1). It is essential that the discretisation at these non-conformal interfaces is done in a conservative fashion in order to obtain stability of convergence. Therefore this test case is very demanding. Since the test case is also rotationally symmetric only one blade channel is being modelled using cyclic AMI interfaces. The remainder of the boundaries is treated identically to the boundaries in section (10.2.2).

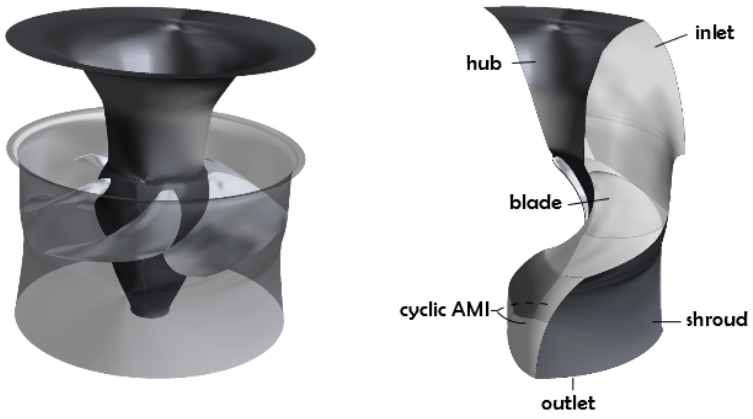

Figure 10.18: *Kaplan* turbine configuration

Velocity profiles evaluated from hub to shroud at the trailing edge again show good conformance in respect to the segregated approach (see Figure 10.19).

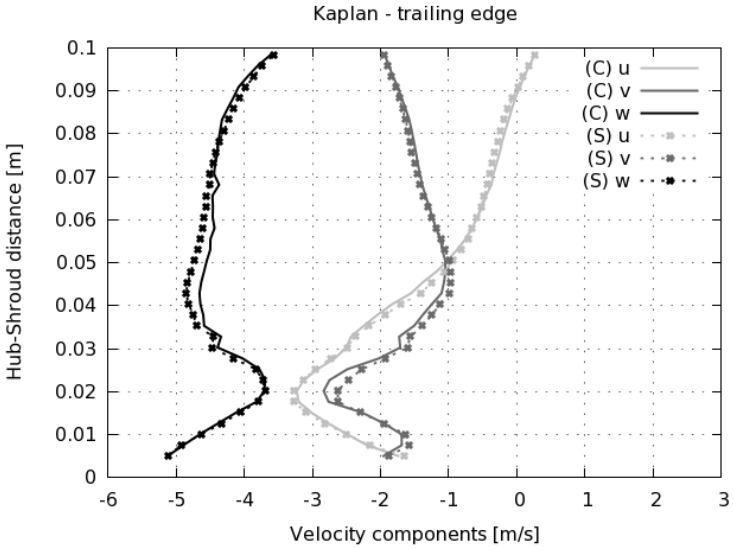

Figure 10.19: *Kaplan* - velocity profiles close to trailing edge

For this test case the coupled and segregated methods are only compared for one grid size. Table 10.6 and Figure 10.20 again prove the superiority of the coupled solver over the segregated solver.

Table 10.6: *Kaplan* - Performance comparison of the coupled (C) and segregated (S) algorithm

# CV	(C)time[s]	(C)time/CV[s]	(S)time[s]	(S)time/CV[s]	S/C
275k	960.7	0.003489	5840.6	0.021215	6.08

The coupled approach's fast convergence rate of the *Kaplan* test case and its relative smoothness in convergence indicate that not only the method is faster than the segregated method, but also that the solver is very robust due to the strong inter-equation and inter-component coupling, as well as due to the fully implicit implementation of the coupled interfaces (cyclic AMIs).

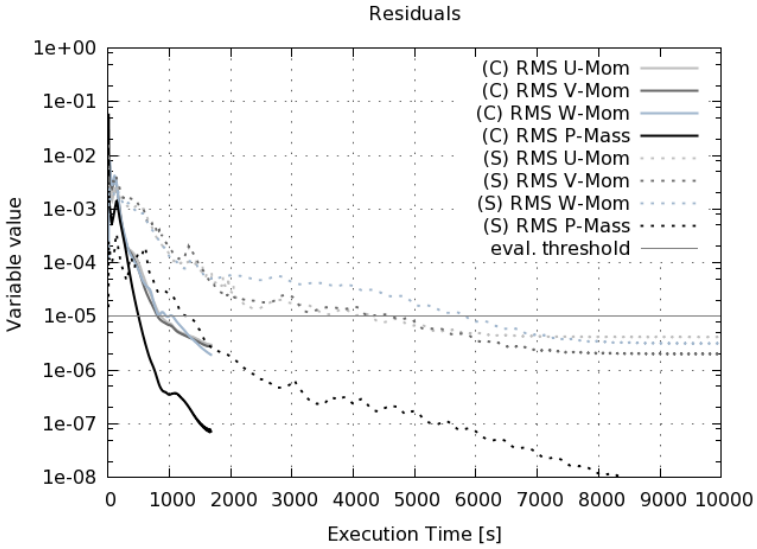

Figure 10.20: Convergence histories for *Kaplan* test case. Segregated(S) - dotted lines, coupled(C) - full lines

10.3 Transient dynamic mesh

10.3.1 Centrifugal Pump

The physical accuracy of the transient block-coupled solver for rotating meshes was tested by comparing results of a centrifugal pump with experimental data from PIV (particle image velocimetry) measurements carried out by Stefan Hödl [2]. Therefore a special pump model was built with a spiral casing that consisted of two halves, one of which was entirely made of plexiglass as can be seen in Figure 10.21 (d). A laser-light-section was introduced at a plane perpendicular to the rotation axis through the plexiglass section of the spiral casing. The model set-up with the corresponding laser-light section is seen in Figures 10.21 (a) and (b).

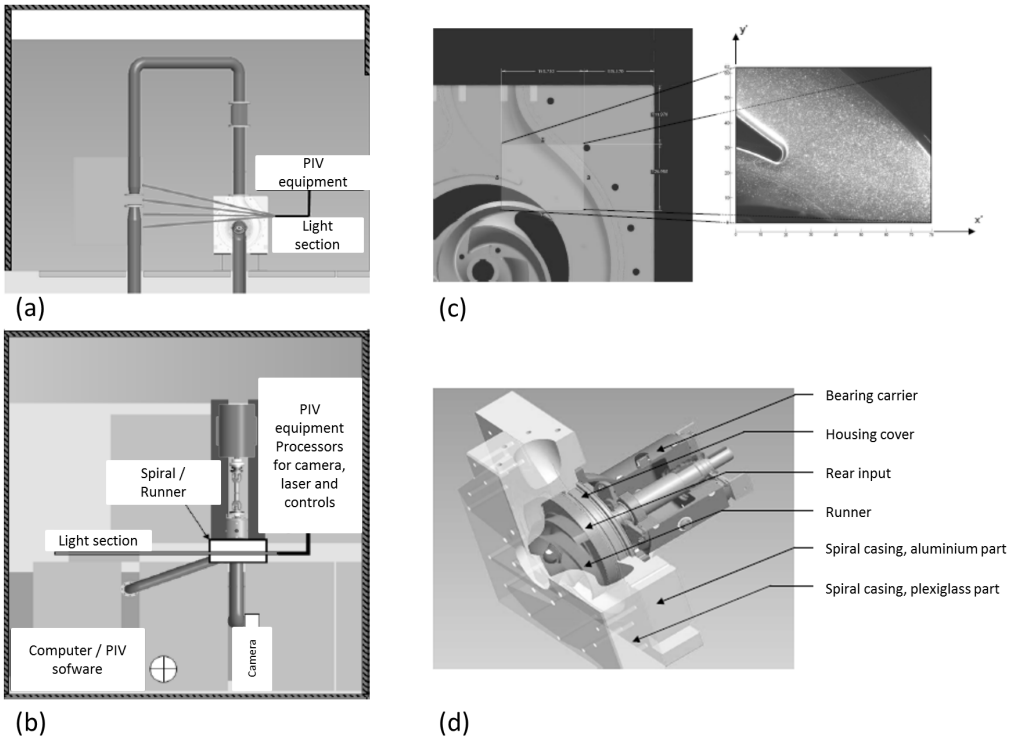

Figure 10.21: *Centrifugal pump* - velocity profiles close to trailing edge [2]

At a section of this plane (Figure 10.21 (c)) time-averaged velocity profiles were generated at various runner positions.

Corresponding to this set-up and a transient simulation with a dynamically rotating mesh

(runner) was carried out. The simulation's set-up consists of a static intake part, a rotating runner part, and a static spiral casing part. Velocity profiles were taken at the same plane as for the experimental model. The computational set-up is seen in Figure 10.22

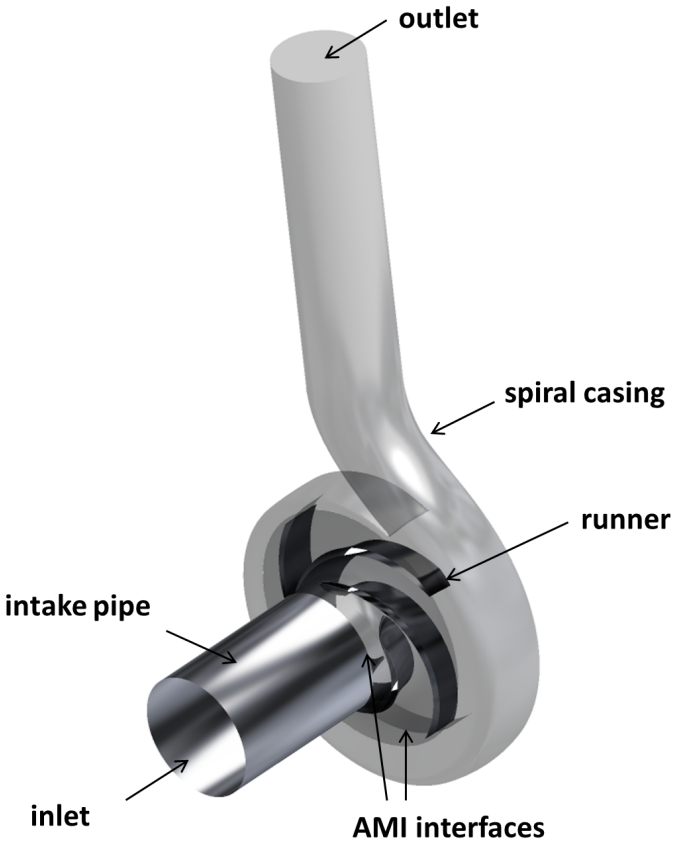

Figure 10.22: *Centrifugal pump* configuration

The computational model consists of an intake pipe, the walls of which are not rotating. A rotating runner, whose inlet is connected to the intake pipe via non-conformal AMI interfaces (as outlined in section 7.1). The runner's blades as well as its hub are rotating with a prescribed rotational velocity. The shroud is non-rotating. The runner's outlet is connected to the spiral casing via non-conformal AMI interfaces. The spiral casing's wall are static. At the inlet a constant velocity profile is prescribe that complies with the operating point's volume flux rate, and a zero gradient boundary condition is applied for the pressure. At the outlet a constant pressure field and a zero gradient condition for the velocity field are prescribed.

In what follows experimental and numerical velocity profiles at the section seen in Figure 10.21

(c) are compared for the optimum load operation point.

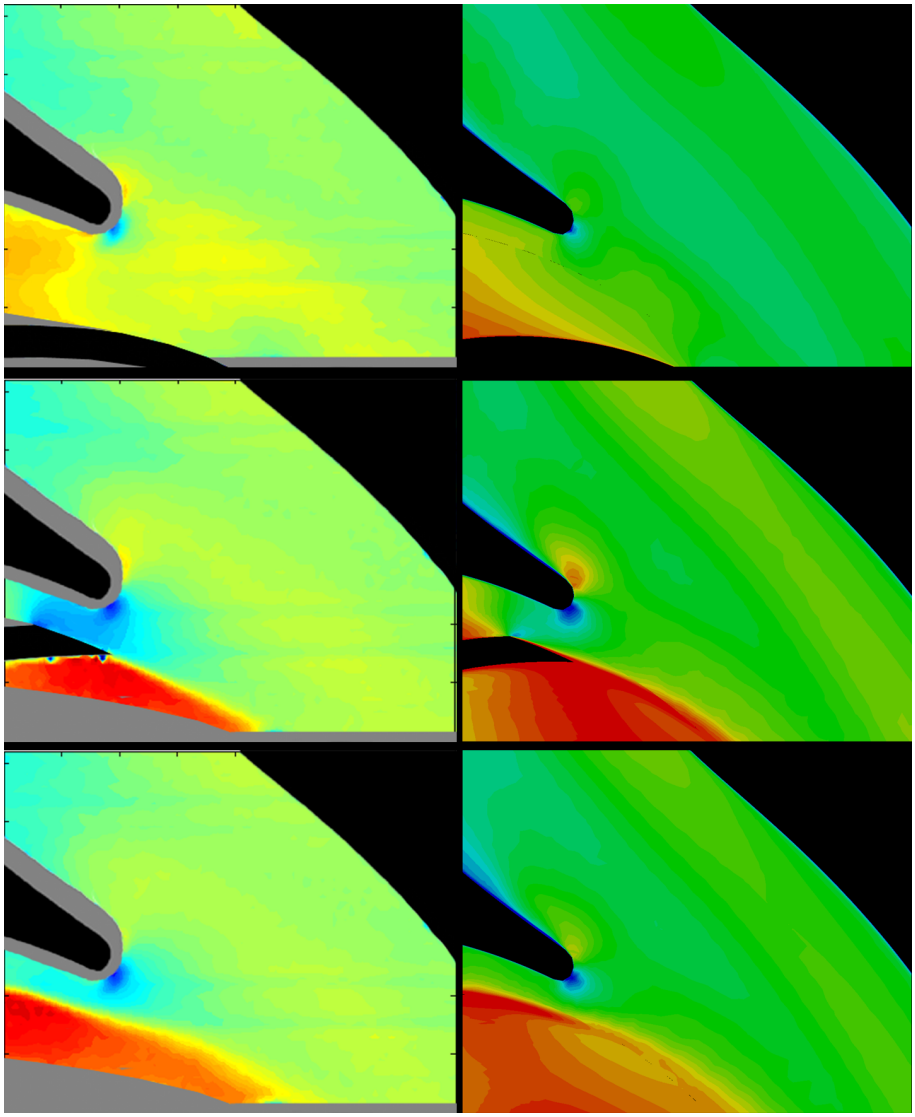

Figure 10.23: left - PIV measurements, right - Block coupled transient solver results

Experimental and numerical results shown in Figure 10.23 exhibit overall a relatively good qualitative concordance. Stagnation points are relatively well captured and the main flow features around the spur coincide quite well. The scaling in may have been slightly different between PIV measurements and numerical results. A more thorough investigation is in preparation, including comparisons at off-design operation conditions.

Chapter 11

Conclusion

Contents

11.1 Summary of Thesis Achievements

The work presented in this thesis comprises the development of a set of fully coupled incompressible pressure-based solvers, using the OpenFOAM CFD library. A steady state solver for inertial reference frames, a solver for rotational reference frames and transient solvers, including the Arbitrary Lagrange Euler formulation for moving meshes have been presented. Efficient solution strategies for solving block coupled linear equation systems have been outlined, including a block multi-grid solver based on an additive correction approach. The governing equations for various configurations of reference frames have been derived. The discretisation of all arising terms has been presented for polygonal meshes. In this context the discretisation at domain interfaces has been outlined in sufficient detail. Caution has been given to assure that also at these interfaces implicit coefficients were derived to preserve the benefit of strong variable coupling. Finally all code features have been implemented to allow parallelization by

means of domain decomposition and parallel interfaces. Furthermore a variety of test cases has been carried out. Thereby a $k - \Omega$ SST turbulence model has been tested and comparisons to results of a segregated solver have been made. Finally simulations of fully transient pump flows have been carried out and compared to particle image velocimetry (PIV) measurements.

The meticulous implementation of mentioned features resulted in a set of solvers that are able to handle even very big industrial applications. The developed set of solvers was found to outperform state-of-the-art segregated solvers in terms of convergence speed, parallel scalability, mesh size scalability and robustness.

11.2 Applications

The block coupled solvers that have been presented in this work can be used in a wide range of applications. From a designer's point of view, the main advantage of these solvers in respect to commercial codes codes is their flexibility to implement new physical models and features. A further big advantage is the absence of license costs.

The most prominent applications for which the solvers are used at Andritz Hydro are outlined next.

Standard Static Part Calculations

These calculations include parts such as spiral casings, Pelton turbine distributors, draft tubes or bulb turbine distributors.

Multiple Reference Frame Calculations

The developed solver for steady solutions in rotational reference frames is employed to calculate fluid flows in pumps, pump-turbines as well as in Kaplan and Francis turbines.

Coupled solver as Navier Stokes Engine for Genetic Optimizer

The robustness of outlined solvers makes them particularly attractive for use as a Navier-Stokes engine for genetic optimization.

Transient Calculations

Transient calculations with dynamic meshes are well suited for detailed studies of flow phenomena inside rotating turbomachinery. Robustness is often an issue for this kind of application. There the coupled solver solver shows its strength arising from tight inter-equation coupling.

11.3 Future Work

The author believes that the implementation of the block coupled solvers has reached a certain state of maturity. All solvers are very stable and fast. Hence the achieved state of the code should give a very good basis for future developments. In present work only a 2-equation (U)RANS turbulence model has been considered. The author believes that it makes sense to implement alternative turbulence models within the coupled framework. One may run into numerical difficulties when trying to couple turbulence equations more implicitly because of their very non-linear nature. However, some terms of these equations may be suitable for extra-diagonal implicit coupling.

A further point of interest may be the implementation of averaging interfaces (mixing plane interfaces) between the single rotor channel and the stator of turbomachines to alleviate the computational cost of such calculations.

Although the author believes that the object-oriented code structure of the developed block coupled features and solvers is quite well organized there is still space for improvement. Since the OpenFOAM library has originally been designed for segregated algorithms the developers of the basic block coupling objects have been trying to stick as much as possible to existing features. Additional features that have been lined out in this work do also stick to this, say, segregated base structure. At some point, however, programming within segregated solver driven framework becomes tedious and parts of the main code structure and some object structures pose serious obstacles for the programming of coupled features. This results in hard-coded implementations that make the code less readable and more difficult to maintain. For that reason, the author and his colleagues, Luca Mangani and Marwan Darwish, have i.e. struggled a lot with the implementation of coupled boundary conditions and coupled interfaces, such as periodic interfaces, arbitrary mesh interfaces, or parallel interfaces. The implementation of off-diagonal coupling parts has also been done in a hard coded fashion. It is a difficult task to improve the existing block coupling framework that is available within the OpenFOAM library and to make it fully templated and object based. Restructuring the code is still considered worthwhile in order to have a neat object-oriented structure providing ease of maintenance.

This would further increase the potential of this block coupled OpenFOAM framework.

Additional improvements in terms of computational performance could be made by applying even better and innovative solution strategies for the solution of coupled linear systems of equations. This also includes the development of tailor-made preconditioners for a given system.

Finally the author believes that the coupled framework may be suitable for a wide range of other applications such as multiphase solvers. Any system of equations that contains strongly coupled components may benefit of a tighter variable coupling.

Appendix A

NVD/TVD

The convection term is discretised using Gauss' Theorem and a limited surface interpolation scheme. Two distinct discretisation methods, namely the normalised variable diagram schemes (NVD) and the total variation diminishing schemes (TVD) are implemented in OpenFOAM. Both methods are based on limiters, which guarantee the schemes' boundedness. In both cases the limiters intrinsically carry the information of two consecutive upwind gradients. In this section the discretisation of the NVD and TVD schemes, as they are employed in OpenFOAM will be outlined, following Jasak et al.[3] and Darwish et al. [40]. For a more detailed description of the schemes the reader is refered to the mentioned papers. As an example for the NVD schemes, the Gamma scheme by Jasak et al. [3] will be outlined. Van Leer's limiter will be used as an example for TVD schemes. This scheme will subsequently be used to further derive the discretisation of the governing equations. It shall also be stated at this point that a deferred correction approach has to be taken for both schemes in order to assure that the sum of implicit off-diagonal coefficient values, arising from the discretisation, is not bigger then the diagonal coefficient value. For time (n+1) the scheme will produce upwind coefficients, fulfilling this plus a limited surface interpolation contribution to the source term. This limited contribution has to be updated using subloops. The deferred corrected van Leer TVD scheme is between first and second order accurate.

Usually NVD and TVD schemes are derived for structured grids, and they use the far upwind

cells to calculate an unboundedness indicator at cell faces. Bearing in mind that OpenFOAM's code structure is based on unstructured grid, and that it can only address a cell's neighbours, a way has to be found to circumvent the missing far upwind cell's information.

At this point OpenFOAM's implementation of NVD schemes shall be laid out, following [3]. Additionally TVD schemes for unstructured grids, as they are implemented in OpenFOAM will be derived based on [40] [4] [41].

A.1 NVD Schemes

Let us consider a convected scalar ϕ around a cell surface as seen in figure (A.2). Following the notations of figure (A.2) the normalized variable can be defined as:

$$\tilde{\phi} = \frac{\phi - \phi_U}{\phi_D - \phi_U} \tag{A.1}$$

U and D are the far upwind and downwind indices respectively, C is the upwind cell index and f is the face index at the face under consideration (surface integration).

We state that,

$$\tilde{\phi}_f = f\left(\tilde{\phi}_C\right) \tag{A.2}$$

where

$$\tilde{\phi}_C = \frac{\phi_C - \phi_U}{\phi_D - \phi_U} \tag{A.3}$$

and

$$\tilde{\phi}_f = \frac{\phi_f - \phi_U}{\phi_D - \phi_U} \tag{A.4}$$

To avoid unphysical oscillations, ϕ_C and consequently ϕ_f have to be locally bounded, such that

$$\phi_U \leq \phi_C \leq \phi_D \tag{A.5}$$

or

$$\phi_U \geq \phi_C \geq \phi_D \tag{A.6}$$

The convection boundedness criterion (CBC) by Leonard [42] states that:

$$0 \leq \tilde{\phi}_C \leq 1 \tag{A.7}$$

Hence,

$$\begin{aligned} \tilde{\phi}_C = 0 \Rightarrow \phi_C = \phi_U \\ \tilde{\phi}_C = 1 \Rightarrow \phi_C = \phi_D \end{aligned} \tag{A.8}$$

Figure (A.1) a) shows $\tilde{\phi}_f$ as a function of $\tilde{\phi}_C$. Therein the shaded region is the region where the CBC is fulfilled [43].

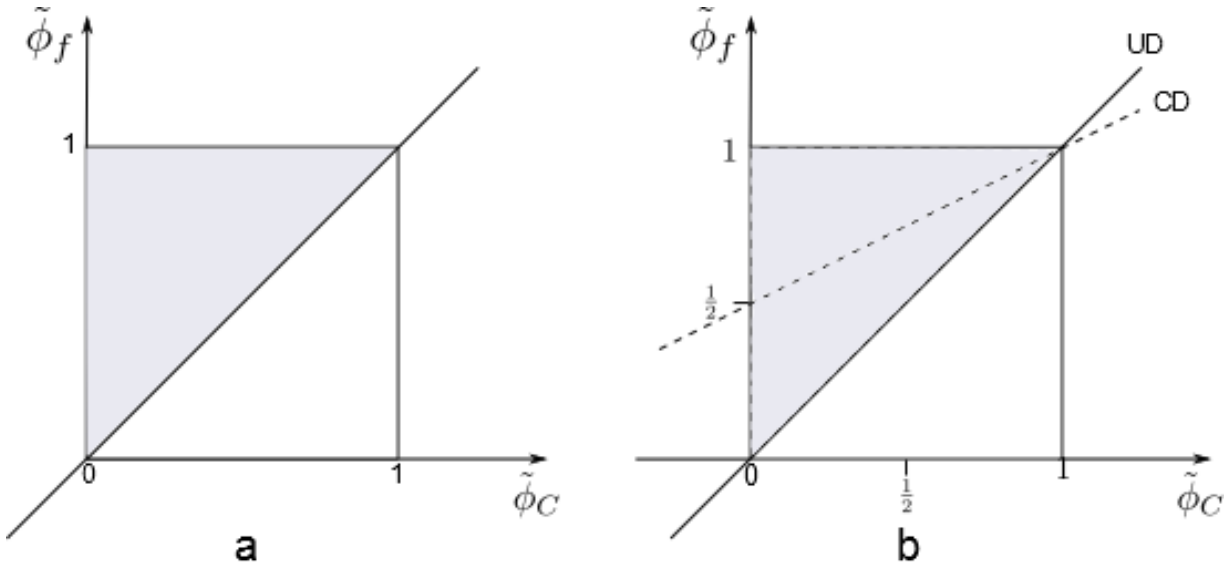

Figure A.1: NVD diagram by Jasak[3]

mod

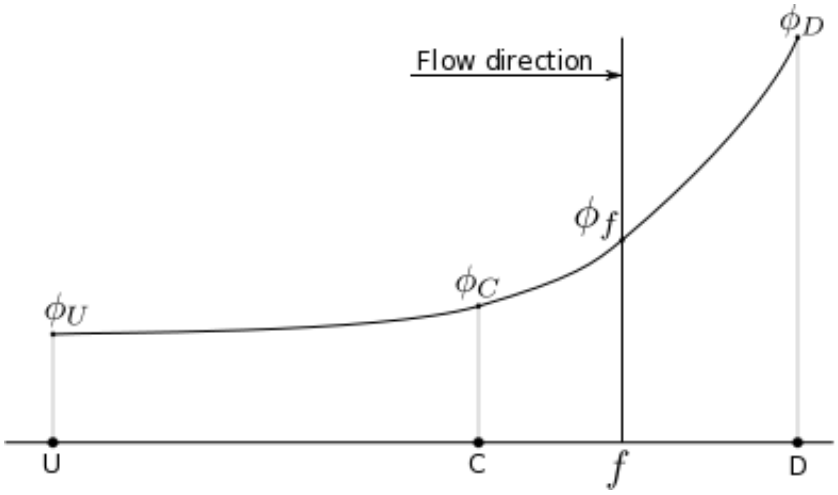

Figure A.2: Convection Boundedness Criterion

As the CBC uses variable values ϕ_U of the far upwind cell, the CBC has to be modified for arbitrary unstructured meshes, which do not know its far upwind node. Independent of the grid structure, the information available for each face comprises the gradients of ϕ on the face as well as the gradients at the centroids of its adjacent cells. This information will be used to replace the missing far upwind information. Figure (A.3) is similar to figure (A.2), but now values ϕ_f^+ and ϕ_f^- are introduced at the cell faces. If ϕ_C is bounded by ϕ_U and ϕ_D it will also be bounded by ϕ_f^- and ϕ_f^+.Hence,

$$\tilde{\phi}_C = \frac{\phi_C - \phi_f^-}{\phi_f^+ - \phi_f^-} = 1 - \frac{\phi_f^+ - \phi_C}{\phi_f^+ - \phi_f^-} \tag{A.9}$$

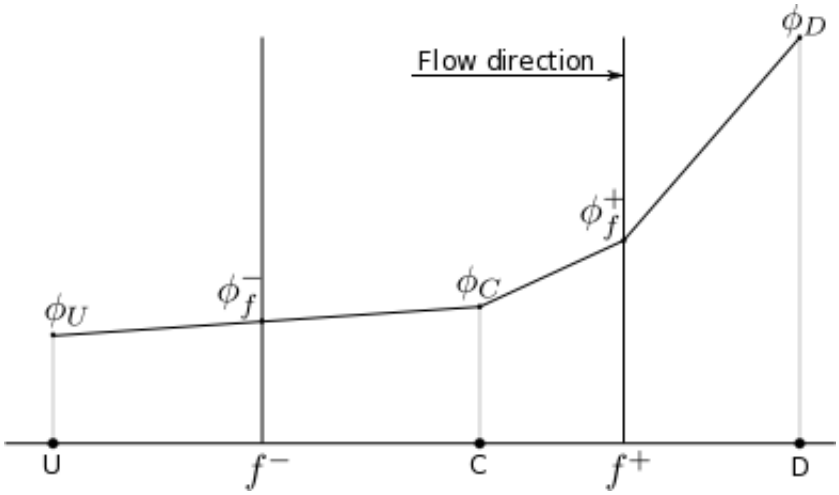

Figure A.3: Modified Convection Boundedness Criterion for unstructured meshes

The goal now is to reformulate (A.9), using face and cell gradients in a way that avoids the use of far upwind values ϕ_f^- and ϕ_U.

The linear interpolation factor at a face g_f is given by:

$$g_f = \frac{d_{face,D}}{d_{C,D}} \tag{A.10}$$

or more generally for non-orthogonal grids

$$g_f = \frac{mag\left[\mathbf{S}_{f^+} \cdot (\mathbf{x}_D - \mathbf{x}_{f^+})\right]}{mag\left[\mathbf{S}_{f^+} \cdot (\mathbf{x}_{f^+} - \mathbf{x}_C)\right] + mag\left[\mathbf{S}_{f^+} \cdot (\mathbf{x}_D - \mathbf{x}_{f^+})\right]} \tag{A.11}$$

Furthermore the numerator of the fraction at the right hand side of (A.9) can be written as

$$\phi_f^+ - \phi_C = g_f \frac{\phi_D - \phi_C}{x_D - x_C}\left(x_D - x_C\right) = g_f \left(\nabla\phi\right)_{f^+} \cdot \hat{\mathbf{d}}_{C,D}\left(x_D - x_C\right) \tag{A.12}$$

with

$$\begin{aligned} \frac{\phi_D - \phi_C}{x_D - x_C} &= (\nabla\phi)_{f^+} \cdot \hat{\mathbf{d}} \\ \hat{\mathbf{d}}_{C,D} &= \frac{\mathbf{d}_{C,D}}{|\mathbf{d}_{C,D}|} \end{aligned} \tag{A.13}$$

The denominator of the fraction at the right hand side of (A.9) can be written as

$$\phi_f^+ - \phi_f^- = \frac{\phi_f^+ - \phi_f^-}{x_f^+ - x_f-} = (\nabla\phi)_C \cdot \hat{\mathbf{d}}_{C,D} \left(x_f^+ - x_f^-\right) \tag{A.14}$$

Reinserting the numerator (A.12) and the denominator (A.14) into (A.9) we obtain.

$$\tilde{\phi}_C = 1 - \frac{g_f \left(\nabla\phi\right)_{f^+} \cdot \hat{\mathbf{d}}_{C,D} \left(x_D - x_C\right)}{\left(\nabla\phi\right)_C \cdot \hat{\mathbf{d}}_{C,D} \left(x_f^+ - x_f^-\right)} \tag{A.15}$$

Given that the upwind cell's value is placed at its centroid, it will be assumed that an imaginary far upwind cell's centroid is situated as far away of the upwind cell's centroid as the downwind cell's centroid. Hence,

$$\frac{x_f^+ - x_f^-}{x_D - x_C} = 2\frac{x_f^+ - x_C}{x_D - x_C} = 2g_f \tag{A.16}$$

Merging equations (A.15) and (A.16) gives

$$\tilde{\phi}_C = 1 - \frac{g_f \left(\nabla\phi\right)_{f^+} \cdot \hat{\mathbf{d}}_{C,D}}{2\left(\nabla\phi\right)_C \cdot \hat{\mathbf{d}}_{C,D}} \tag{A.17}$$

with

$$\left(\nabla\phi\right)_{f^+} \cdot \hat{\mathbf{d}}_{C,D} = \phi_D - \phi_C \tag{A.18}$$

we obtain

$$\tilde{\phi}_C = 1 - \frac{\phi_D - \phi_C}{2\left(\nabla\phi\right)_C \cdot \hat{\mathbf{d}}_{C,D}} \tag{A.19}$$

An example of a NVD scheme implemented in OpenFOAM is the Gamma scheme.

Following Jasak et al.[3] the correlation $\tilde{\phi}_f = f\left(\tilde{\phi}_C\right)$ is expressed in three parts. for $\tilde{\phi}_C \leq 0$ or $\tilde{\phi}_C \geq 0$ (UD):

$$\tilde{\phi}_f = \tilde{\phi}_C \tag{A.20}$$

for $0 \leq \beta_m \leq 1$ (blended UD-CD):

$$\tilde{\phi}_f = \frac{1}{2} + \frac{1}{2}\tilde{\phi}_C \tag{A.21}$$

for $\beta_m \leq 0 \leq 1$ (CD):

$$\tilde{\phi}_f = \frac{1}{2} + \frac{1}{2}\tilde{\phi}_C \tag{A.22}$$

β_m is a blending factor. If it is set to zero, the differencing scheme jumps apruptly from central differencing to upwind differencing when the scheme detects a unboundedness limit. This jump however, as can be seen in figure (A.4) a) can deteriorate convergence rates significantly. Hence a smoother transition from UD to CD is preferred (figure (A.4) b)). Jasak et al. [3] propose a value of 0.5 for β_m.

Figure (A.1) shows how the Gamma NVD scheme acts in the NVD diagram. Outside the unity rectangle the scheme limits itself to an upwind scheme. Inside the unity rectangle the scheme blends smoothly towards the central differencing scheme.

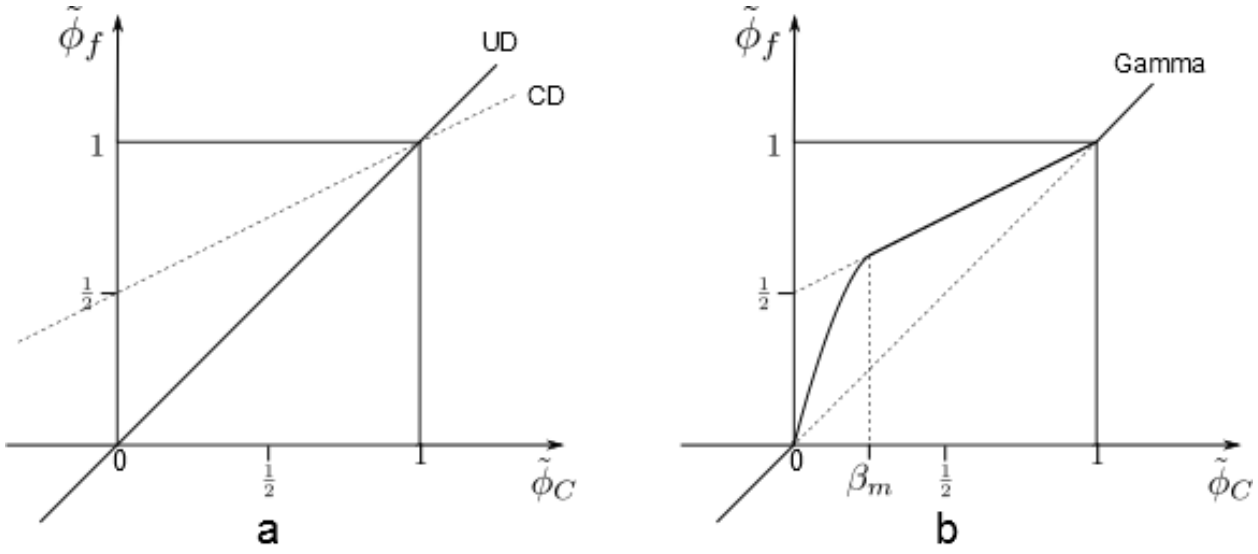

Figure A.4: NVD diagram for Upwind Differencing scheme (UD) and Central Differencing scheme (CD) at the left, Gamma NVD scheme on the right

A.2 TVD Schemes

Following Roe [41], TVD schemes' flux can be written as a linear combination of a diffusive but stable upwind part, and an antidiffusive (second order) part, which is multiplied by the flux limiter function $\Upsilon(r_f)$, which is a non-linear function of r_f [40], the upwind ratio of consecutive gradients. The factor g_f (equ: A.29) accounts for non-homogenious grid spacing.

$$\phi_f = \phi_C + (1 - g_f)\,\Upsilon(r_f)\,(\phi_D - \phi_C) \tag{A.23}$$

This upwind ratio of consecutive gradients $\Upsilon(r_f)$ is defined in a slightly different way as its equivalent $\tilde{\phi}_C$ for the NVD schemes (equ. (A.3))

$$r_f = \frac{\phi_C - \phi_U}{\phi_D - \phi_C} = \frac{-(\phi_D + \phi_C) + (\phi_D - \phi_C)}{\phi_D - \phi_C} \tag{A.24}$$

$$r_f = \frac{\phi_D - \phi_U}{\phi_D - \phi_C} - 1 \tag{A.25}$$

Where r_f is a scalar field adherent to the surfaces of the grid. The values would be computable if the far upwind node was replaced by a known term [40]. This can be done similarly to (A.19).

$$\phi_D - \phi_U = \nabla\phi_C \cdot \mathbf{d}_{U,D} = 2\nabla\phi_C \cdot \mathbf{d}_{C,D} \tag{A.26}$$

Vector $\mathbf{d}_{C,D}$ is the distance vector between the downwind and upwind node. Vector $\mathbf{d}_{U,D}$ is here assumed to be along vector $\mathbf{d}_{C,D}$ with the upwind cell's centroid C being in the middle of the downwind and an imaginary far upwind node. Hence we obtain,

$$r_f = \frac{2\nabla\phi_C \cdot \mathbf{d}_{C,D}}{\phi_D - \phi_C} - 1 \tag{A.27}$$

Note that this notation is also valid for cells neighboring boundary faces.

The van Leer TVD scheme shall be outlined as an example for TVD schemes in OpenFOAM. It will also be used in subsequent derivations as it is the convection scheme chosen for the testcases in chapter (10).

The van Leer limiter function is given by:

$$\Upsilon\left(r_f\right) = \frac{r_f + |r_f|}{1 + |r_f|} \tag{A.28}$$

Figure (A.5) illustrates van Leer's limiter function in Sweby's diagram [4]. The grey region is the region where boundedness is guaranteed, and is therefore the admissible limiter region for second order convection schemes [4]. Note that all stable second order limiters have to go through point (1,1) to guarantee the TVD criterion. In figure (A.5) it can be seen that van Leer's limiter is a smooth function, which effects convergence in a positive way. Furthermore, for all negative values of r_f the limiter function $\Upsilon\left(r_f\right)$ is null, hence the scheme contracts to an upwind scheme.

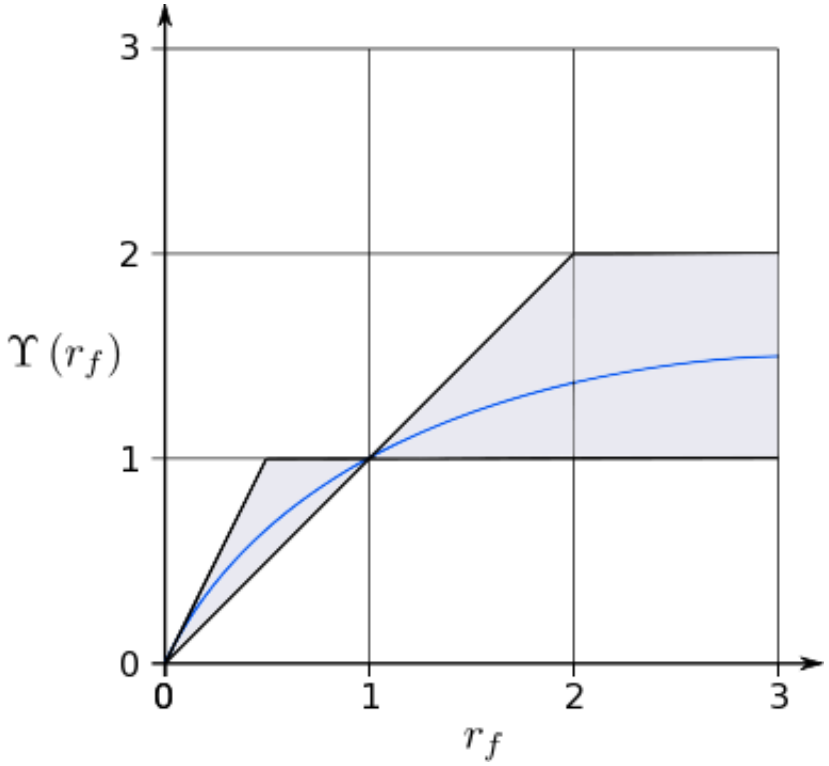

Figure A.5: van Leer's limiter function (blue line) in Sweby's diagram [4], with second order TVD region (grey)

OpenFOAM formulation:

All equations so far were written for cells U, C, D (upwind, central, downwind). We shall now generalize the equations to match OpenFOAM's code structure. Remember that OpenFOAM's code structure is based on owner and neighbour cells. Depending on the face flux direction, the upwind cell can be a neighbor cell and the downwind cell an owner cell, and vice versa.

The linear interpolation factor at a face g_f in OpenFOAM can be expressed in general terms as:

$$g_f = \frac{mag\left[\mathbf{S}_f \cdot (\mathbf{x}_{nei} - \mathbf{x}_f)\right]}{mag\left[\mathbf{S}_f \cdot (\mathbf{x}_f - \mathbf{x}_{own})\right] + mag\left[\mathbf{S}_f \cdot (\mathbf{x}_{nei} - \mathbf{x}_f)\right]} \tag{A.29}$$

And equation (A.23) in OpenFOAM language yields:

$$\begin{aligned} &case \;\; flow\, direction\, is\, positive: \\ &\phi_f = \phi_{own} + (1 - g_f)\,\Upsilon\,(r_f)\,(\phi_{nei} - \phi_{own}) \\ &case \;\; flow\, direction\, is\, negative: \\ &\phi_f = \phi_{nei} + g_f \Upsilon\,(r_f)\,(\phi_{own} - \phi_{nei}) \end{aligned} \tag{A.30}$$

Appendix B

Gauss Divergence Theorem

Gauss' theorem relates a volume to a surface integral [44]. For vector fields every CFD engineer knows the divergence theorem by heart. The author however believes that it is good to give a short roundup also for scalar and tensor fields.

Let ϕ, $\mathbf{u}$ and $\mathbf{T}$ be continuous scalar, vector and tensor fields, that are at least once differentiable within a volume Ω, which is bounded by a continuous surface S with outward-pointing unit normal vectors $\mathbf{n}$ [44].

Gauss' theorem for scalar fields ϕ yields:

$$\int_{\Omega} (\nabla \cdot \phi) \, \mathrm{d}\Omega = \oint_{S} (\mathbf{n}\phi_f) \, \mathrm{dS} = \oint_{S} \mathbf{S}_f \phi_f \tag{B.1}$$

Gauss' theorem for vector fields $\mathbf{u}$ yields:

$$\int_{\Omega} (\nabla \cdot \mathbf{u}) \, \mathrm{d}\Omega = \oint_{S} (\mathbf{n} \cdot \mathbf{u}) \, \mathrm{dS} = \oint_{S} \mathbf{S}_f \cdot \mathbf{u} \tag{B.2}$$

Gauss' theorem for tensor fields $\mathbf{T}$ yields:

$$\int_{\Omega} (\nabla \cdot \mathbf{T}) \, \mathrm{d}\Omega = \oint_{S} (\mathbf{n} \cdot \mathbf{T}) \, \mathrm{dS} = \oint_{S} \mathbf{S}_f \cdot \mathbf{T} \tag{B.3}$$

and for the transpose:

$$\int_{\Omega} (\nabla \cdot \mathbf{T})^T \, \mathrm{d}\Omega = \oint_{S} (\mathbf{n} \cdot \mathbf{T})^T \, \mathrm{dS} = \oint_{S} (\mathbf{S}_f \cdot \mathbf{T})^T \tag{B.4}$$

Bibliography

[1] H. Jasak, *Error analysis and estimation in the Finite Volume method with applications to fluid flows*. PhD thesis, Imperial College London, 1996.

[2] S. Hödl, "Particle Image Velocimetry Messung an einer Kreiselpumpe nq30," Master's thesis, Graz University of Technology, 2011.

[3] H. Jasak, H. Weller, and A. Grosman, "High resolution NVD differencing scheme for arbitrarily unstructured meshes," *International journal for numerical methods in fluids*, 1999.

[4] P. Sweby, "High resolution schemes using flux limiters for hyperbolic conservation laws," *SIAM Journal of Numerical Analysis*, 1984.

[5] J. Marsden and T. Hughes, *Mathematical Foundations of Elasticity*. Dover Publications Inc., NY, 1994.

[6] C. Stoker, *Developments of the Arbitrary Lagrangian-Eulerian method in non-linear solid mechanics*. PhD thesis, Universiteit Twente, 1999.

[7] G. Batchelor, *An Introduction to Fluid Dynamics*. Cambridge University Press, 1967.

[8] H. Steiner, "Höhere Strömungslehre und Wärmeuebertragung." Lecture script, 2011.

[9] J. Anderson, *Computational Fluid Dynamics*. McGraw Hill, 1995.

[10] A. Gehrer, *Entwicklung eines 3D-Navier-Stokes Codes zur numerischen Berechnung der Turbomaschinenströmung*. PhD thesis, TU Graz, 1998.

[11] W. Hauger, W. Schnell, and D. Gross, *Technische Mechanik 3*. Springer, Berlin, 2002.

[12] A. Runchal, "Brian Spalding: CFD and Reality," in *ICHMT International Symposium on Advances in Computational Heat Transfer*, 2008.

[13] L. Caretto, R. Curr, and D. Spalding, "Two Numerical Methods for Three-Dimensional Boundary Layers," *Computer Methods in Applied Mechanics and Engineering*, 1972.

[14] S. Patankar and D. Spalding, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," *International Journal of Heat and Mass Transfer*, 1972.

[15] P. Vanka, "Block-Implicit Multigrid Solution of Navier-Stokes Equations in Primitive Variables," *Journal of Computational Physics*, 1986.

[16] P. Galpin and G. Raithby, "Numerical Solution of problems in incompressible fluid flow," *Numerical Heat Transfer*, 1986.

[17] G. Deng, J. Piquet, P. Queutey, and M. Visonneau, "Incompressible flow calculations with a consistent physical interpolation finite volume approach," *Computers and Fluids*, 1994.

[18] M. Darwish, I. Sraj, and F. Moukalled, "A coupled finite volume solver for the solution of incompressible flows on unstructured grids," *Journal of Computational Physics*, 2008.

[19] S. Patankar, *Numerical Heat Transfer and fluid flow*. McGraw-Hill, 1980.

[20] J. V. Doormal and G. Vanka, "Enhancements of the SIMPLE method for predicting incompressible fluid flow," *Numerical Heat Transfer*, 1984.

[21] J. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*. Springer, Berlin, 1994.

[22] M. Benzi and G. G. aand J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica*, 2005.

[23] C. Rhie and W. Chow, "A numerical study of turbulent flow past an isolated airfoil with trailing edge separation," *AIAA Journal*, 1983.

[24] P. Galpin and G. Raithby, "Treatment of non-linearities in the numerical solution of the incompressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, 1986.

[25] S. Muzaferija, *Adaptive finite volume method for flow predictions using unstructured meshes and multigrid approach.* PhD thesis, University of London, 1994.

[26] M. Darwish and F. Moukalled, "A Review of Boundary Conditions and Their Implementations in CFD Codes," *International Journal for Numerical Methods in Fluids*, 2000.

[27] B. Hutchinson and G. Raithby, "A Multigrid Method based on the Additive Correction Strategy," *Numerical Heat Transfer*, 1986.

[28] S. Keller, "The additive correction multigrid method for unstructured grids." SINMEC Laboratory of Numerical Simulation in Fluid Dynamics and Computational Heat Transfer.

[29] B. Hutchinson, P. Galpin, and G. Raithby, "Application of additive correction multigrid to coupled fluid flow equations," *Numerical Heat Transfer*, 1988.

[30] S. Elias, G. Stubley, and G. Raithby, "An additive agglomeration method for additive correction multigrid," *International Journal for Numerical Methods in Engineering*, 1997.

[31] F. Menter, M. Kuntz, and R. Langtry, "Ten Years of Industrial Experience with the SST Turbulence Model," *Turbulence, Heat and Mass Transfer*, 2003.

[32] F. Menter and Y. Egorov, "A scale-adaptive simulation model using two equation models," *AIAA paper 20051095*, 2005.

[33] K. Hanjalic and B. Launder, "A Reynolds stress model of turbulence and its application to thin shear flows," *Journal of Fluid Mechanics*, 1972.

[34] P. Durbin, "Near-wall turbulence closure modeling without damping functions," *Theoretical and Computational Fluid Dynamics*, 1991.

[35] P. Spalart, W.-H. Jou, M. Strelets, and S. Allmaras, "Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach," *1st AFOSR Int. Conf. on DNS/LES. In: Advances in DNS/LES*, 1997.

[36] B. Kader, "Temperature and concentration profiles in fully turbulent boundary layers," *International Journal of Heat and Mass Transfer*, 1981.

[37] E. Casartelli, L. Mangani, and S. Hug, "Numerical comparison between model and prototype flow in a pump-turbine distributor," in *International Conference and Exhibition Innovative Approaches to Global Challenges 29 to 31*, 2012.

[38] L. Mangani, M. Buchmayr, and M. Darwish, "Development of a novel Fully Coupled Block Solver in OpenFOAM: Steady State Incompressible Turbulent Flows," *submitted to Numerical Heat Transfer Part B: Fundamentals*, 2014.

[39] L. Mangani, M. Buchmayr, and M. Darwish, "Development of a novel Fully Coupled Block Solver in OpenFOAM: Steady State Incompressible Turbulent Flows in Rotational Reference Frames," *submitted to Numerical Heat Transfer Part B: Fundamentals*, 2014.

[40] M. Darwish and F. Moukalled, "TVD Schemes for unstructured grids," *International Journal of Heat and Mass Transfer*, 2003.

[41] P. Roe, "Some contributions to the modeling of discontinous fluids," in *Proceedings of the AMS/SIAM Seminar, San Diego*, 1983.

[42] B. Leonard, "Simple high-accuracy resolution program for convective modeling of discontinuities," *International journal for numerical methods in fluids*, 1988.

[43] P. Gaskell and A. Lau, "Curvature-compensated convective transport: SMART, a new boundedness-preserving transport algorithm," *International journal for numerical methods in fluids*, 1988.

[44] P. Rajinder, *Rheology of Particulate Dispersions and Composites*. CRC Press, 2007.