Implementation and Performance Analyses of a Highly Efficient Algorithm for Pressure-Velocity Coupling

Implementierung und Untersuchung einer hoch effizienten Methode zur Druck-Geschwindigkeits-Kopplung

Master-Thesis von Fabian Gabel

Tag der Einreichung:

1. Gutachten: Prof. Dr. rer. nat. Michael Schäfer
2. Gutachten: Dipl.-Ing Ulrich Falk

TECHNISCHE UNIVERSITÄT DARMSTADT

Studienbereich CE
FNB

Implementation and Performance Analyses of a Highly Efficient Algorithm for Pressure-Velocity Coupling
Implementierung und Untersuchung einer hoch effizienten Methode zur Druck-Geschwindigkeits-Kopplung

Vorgelegte Master-Thesis von Fabian Gabel

1. Gutachten: Prof. Dr. rer. nat. Michael Schäfer
2. Gutachten: Dipl.-Ing Ulrich Falk

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 13. Februar 2015

(Fabian Gabel)

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Ma	Mach number
S	Surface
$S_{ij}, i, j \in \{1, 2, 3\}$	Symmetric part of the transpose of the jacobian of the velocity
T	Temperature
T_0	Reference temperature
V	Volume
β	Coefficient of thermal expansion
δ_{ij}	Kronecker-Delta
$\hat{p}$	Pressure without hydrostatic pressure part
κ	Thermal conductivity
$(\mathbf{e}_i)_{i=1,\dots,3}$	Cartesian canonical basis
μ	Dynamic viscosity
ρ	Density
ρ_0	Reference density at T_0
$\sigma_{ij}, i, j \in \{1, 2, 3\}$	Deviatoric stress tensor
$\tau_{ij}, i, j \in \{1, 2, 3\}$	Coefficient matrix of stress mapping T
$\mathbf{T}$	Linear stress mapping
$\mathbf{g}$	Gravitational acceleration vector
$\mathbf{k}$	Mass specific force vector
$\mathbf{n}$	Surface unit normal vector
$\mathbf{t}$	Stress vector
$\mathbf{u}$	Velocity vector
$\mathbf{x}$	Coordinate Vector
$g_i, i \in \{1, 2, 3\}$	Components of the gravitational acceleration vector
$k_i, i \in \{1, 2, 3\}$	Mass specific force vector components
$n_i, i \in \{1, 2, 3\}$	Surface unit normal vector components
p	Pressure
q_T	Source or sink of heat
t	Time
$t_i, i \in \{1, 2, 3\}$	Stress vector components
$u_i, i \in \{1, 2, 3\}$	Cartesian velocity components
$x_i, i \in \{1, 2, 3\}$	Cartesian coordinates

TODO LIST

- midpoint integration rule or midpoint rule of integration
- remove equation numbering when there is no reference
- change the matrix coefficient indexing to $a_P^{u_i,p}$ like in [**?**].
- extend nomenclature
- align equations using the **alignat** environment
- check pressure correction equation on iteration indices (gradients)
- check simple chapter, since the right hand side lacks of deferred corrector and under-relaxed velocities
- mention the boundary conditions for the pressure correction
- check all headings for correct spelling
- mention that for an unknown velocity field the partial differential equation for the temperature is non-linear as well
- consistent use of either temperature or energy equation
- check the signs in the Boussinesq approximation
- pressure weighted interpolation method for large body forces?
- not only speed but also improvement of robustness
- instationary flows?
- align exponents in the equation for the Newton-Raphson linearization
- define the term consistent
- add citations to clipper, opencascade and maple, ICEM CFD
- a priori exact solution use this formulation
- Falsche Wahl der problem domain, führt zu global nicht erfüllter kontinuitätsgleichung. residuum der druckkorrectur entspricht dem massenfluss
- reference Comparison of finite-volume numerical methods with staggered and colocated
- read klaij again and use some of its arguments grids
- emphasize the flexibility of the implementation

1 Introduction

2 Fundamentals of Continuum Physics for Thermo-Hydrodynamical Problems

This section covers the set of fundamental equations for thermo-hydrodynamical problems, which the numerical solution techniques of the following chapters are aiming to solve. Furthermore, the notation regarding the physical quantities to be used throughout this thesis is introduced. The following paragraphs are based on [**?**,**?**,**?**,**?**]. For a thorough derivation of the matter to be presented the reader may consult the mentioned sources. Since the present thesis focusses on the application of finite-volume methods, the focus lays on stating the integral forms of the relevant conservation laws. However, in the process of deriving the final set of equations, the use of differential formulations of the stated laws is required. Einstein's convention for taking sums over repeated indices is used to simplify certain expressions. For the remainder of this thesis, non-moving inertial frames in a Cartesian coordinate system with the coordinates x_i are used. This approach is also known as *Eulerian approach*.

2.1 Conservation of Mass – Continuity Equation

The conservation law of mass, also known as the continuity equation, embraces the physical concept that, neglecting relativistic and nuclear reactions, mass cannot be created or destroyed. Using the notion of a mathematical control volume, which is used to denote a constant domain of integration, one can state the integral mass balance of a control volume V with control surface S with surface unit normal vector $\mathbf{n} = (n_i)_{i=1,\dots,3}$ using Gauss' theorem as

$$\iiint_V \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i)\,\mathrm{d}V = \iiint_V \frac{\partial \rho}{\partial t}\mathrm{d}V + \iint_S \rho u_i n_i \mathrm{d}S = 0,$$

where ρ denotes the material density, t denotes the independent variable of time and $\mathbf{u} = (u_i)_{i=1,\dots,3}$ is the velocity vector field. Since this equation remains valid for arbitrary control volumes, the equality has to hold for the integrands as well. In this sense, the differential form of the conservation law of mass can be formulated as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0. \tag{1}$$

2.2 Conservation of Momentum – Cauchy-Equations

The conservation law of momentum, also known as Newton's Second Law, axiomatically demands the balance of the temporal change of momentum and the sum of all attacking forces on a body. Those forces can be divided into body forces and surface forces. Let $\mathbf{k} = (k_i)_{i=1,\dots,3}$ denote a mass specific force and $\mathbf{t} = (t_i)_{i=1,\dots,3}$ the stress vector. A first form of the integral momentum balance in the direction of x_i can be formulated as

$$\iiint_V \frac{\partial}{\partial t}(\rho u_i)\,\mathrm{d}V + \iint_S \rho u_i \left(u_j n_j\right)\mathrm{d}S = \iiint_V \rho k_i \mathrm{d}V + \iint_S t_i \mathrm{d}S. \tag{2}$$

In general, the stress vector $\mathbf{t}$ is a function not only of the location $\mathbf{x} = (x_i)_{i=1,\dots,3}$ and of the time t but also of the surface unit normal vector $\mathbf{n}$. A central simplification can be introduced, namely Cauchy's stress theorem, which states that the stress vector is the image of the unit normal vector under a linear mapping $\mathbf{T}$. With respect to the Cartesian canonical basis $(\mathbf{e}_i)_{i=1,\dots,3}$ the mapping $\mathbf{T}$ is represented by the coefficient matrix $\left(\tau_{ji}\right)_{i,j=1,\dots,3}$ and Cauchy's stress theorem reads

$$\mathbf{t}(\mathbf{x}, t, \mathbf{n}) = \mathbf{T}(\mathbf{x}, t, \mathbf{n}) = \left(\tau_{ji} n_j\right)_{i=1,\dots,3}.$$

Assuming the validity of Cauchy's stress theorem, one can derive Cauchy's first law of motion, which in differential form can be formulated as

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j\right) = \rho k_i + \frac{\partial \tau_{ji}}{\partial x_j} \tag{3}$$

and represents the starting point for the modelling of fluid mechanical problems. One should note, that Cauchy's first law of motion does not take any assumptions regarding material properties, which is why the set of equations (1,3) is not closed in the sense that there exists an independent equation for each of the dependent variables.

2.3 Closing the System of Equations – Newtonian Fluids

Resulting from the application of Cauchy's theorem, the stress vector $\mathbf{t}$ can be specified once the nine components τ_{ji} of the coefficient matrix are known. As shown in [**?**, **?**], by formulating the conservation law of angular momentum, the coefficient matrix is symmetric,

$$\tau_{ji} = \tau_{ij}, \tag{4}$$

hence the number of unknown coefficients may be reduced to six unknown components. In a first step it is assumed that the coefficient matrix can be decomposed into fluid-static and fluid-dynamic contributions,

$$\tau_{ij} = -p\delta_{ij} + \sigma_{ij},$$

where p is the thermodynamic pressure, δ_{ij} is the *Kronecker*-Delta and σ_{ij} is the so called *deviatoric stress tensor*.

In the present thesis, viscous fluids are modelled using a linear relation between the components of the deviatoric stress tensor and the symmetric part of the transpose of the Jacobian of the velocity field $\left(S_{ij}\right)_{i,j=1,\dots,3}$,

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

If one now imposes material-isotropy and the mentioned stress-symmetry (4) restriction, it can be shown [**?**] that the constitutive equation for the deviatoric stress tensor reads

$$\sigma_{ij} = 2\mu S_{ij} + \lambda S_{mm}\delta_{ij},$$

where λ and μ denominate scalars which depend on the local thermodynamical state. Taking everything into account, (3) can be formulated as the differential conservation law of momentum for Newtonian fluids, better known as the *Navier-Stokes equations* in differential form:

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j\right) = \rho k_i - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) + \frac{\partial}{\partial x_i}\left(\lambda\frac{\partial u_m}{\partial x_m}\right) \tag{5}$$

2.4 Conservation of Scalar Quantities

The modelling of the transport of scalar or vector quantities by a flow field $\mathbf{u}$, also known as convection, is necessary if the fluid mechanical problem to be analyzed includes for example heat transfer. Other scenarios that involve the necessity to model scalar transport surge, when turbulent flows are to be modeled by two-equation models like the k-ε-model [**?**].

Since this thesis focusses on the transport of the scalar temperature T, this section introduces the conservation law of energy in differential form, formulated in terms of the temperature,

$$\frac{\partial(\rho T)}{\partial t} + \frac{\partial}{x_j}\left(\rho u_j - \kappa\frac{\partial T}{\partial x_j}\right) = q_T, \tag{6}$$

where κ denotes the thermal conductivity of the modelled material and q_T is a scalar field representing sources and sinks of heat throughout the domain of the problem. This equation is also known as the temperature equation.

2.5 Necessary Simplification of Equations

Negligible viscous dissipation and pressure work source terms in the every equation (vakilipour)

The purpose of this section is to motivate and introduce further common simplifications of the previously presented set of constitutive equations.

2.5.1 Incompressible Flows and Hydrostatic Pressure

A common simplification when modelling low Mach number flows ($Ma < 0.3$), is the assumption of *incompressibility*, or the assumption of an *isochoric* flow. If one furthermore assumes homogeneous density ρ in space and time, a restrictive assumption that will be partially alleviated in the following section, the continuity equation in differential form (1) can be simplified to

$$\frac{\partial u_i}{\partial x_i} = 0. \tag{7}$$

In other words: In order for a velocity vector field $\mathbf{u}$ to be valid for an incompressible flow it has to be free of divergence, or *solenoidal* [**?**, **?**].

If furthermore one assumes the dynamic viscosity μ to be constant, which can be suitable in the case of isothermal flow or if the temperature differences within the flow are small, the Navier-Stokes equations in differential form can be reduced to

$$\frac{\partial}{\partial t}(\rho u_i) + \rho \frac{\partial}{\partial x_j}(u_i u_j) = \rho k_i - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) \tag{8a}$$

$$= \rho k_i - \frac{\partial p}{\partial x_i} + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) \tag{8b}$$

by using *Schwartz*'s lemma to interchange the order of differentiation. Another common simplification to further simplify the set of equations is the assumption of a volume specific force $\rho\mathbf{k}$ that can be modelled by a potential, such that it can be represented as the gradient of a scalar field $\Phi_{\mathbf{k}}$ as

$$-\rho k_i = \frac{\partial \Phi_{\mathbf{k}}}{\partial x_i}.$$

In the case of this thesis this assumption is valid since the mass specific force is the mass specific gravitational force $\mathbf{g} = (g_i)_{i=1,\dots,3}$ and the density is assumed to be constant, so the potential can be modelled as

$$\Phi_g = -\rho g_j x_j.$$

This term can be interpreted as the hydrostatic pressure p_{hyd} and can be added to the thermodynamical pressure p to simplify calculations

$$\begin{aligned} \rho g_i - \frac{\partial p}{\partial x_i} &= \frac{\partial}{\partial x_i}(\rho g_j x_j) - \frac{\partial p}{\partial x_i} \\ &= \frac{\partial}{\partial x_i}(\rho g_j x_j) - \frac{\partial}{\partial x_i}(\hat{p} + p_{hyd}) \\ &= -\frac{\partial \hat{p}}{\partial x_i}. \end{aligned} \tag{9}$$

Since in incompressible fluids only pressure differences matter, this has no effect on the solution. After finishing the calculations p_{hyd} can be calculated and added to the resulting pressure $\hat{p}$.

2.5.2 Variation of Fluid Properties – The Boussinesq Approximation

If modelling of an incompressible flow involves heat transfer fluid properties like the density change with varying temperature. If the variation of temperature is small one can still assume a constant density to maintain the structure of the advection and diffusion terms in (5) and only consider changes of the density in the gravitational term. If linear variation of density with respect to temperature is assumed this approximation is called *Boussinesq*-approximation [**?**] REFERENCE. In this case the Navier-Stokes equations are formulated using a reference density ρ_0 at the reference temperature T_0 and the now temperature dependent density ρ

$$\rho(T) = \rho_0(1 - \beta(T - T_0)). \tag{10}$$

Here β denotes the coefficient of thermal expansion. Under the use of the Boussinesq-approximation the incompressible Navier-Stokes equations in differential form can be formulated as

$$\begin{aligned}\rho_0 \frac{\partial (u_i)}{\partial t} + \rho_0 \frac{\partial}{\partial x_j}\left(u_i u_j\right) &= \rho_0 g_i + (\rho - \rho_0) g_i - \frac{\partial p}{\partial x_i} + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \\ &= \frac{\partial}{x_i}\left(\rho_0 g_j x_j\right) + (\rho - \rho_0) g_i - \frac{\partial p}{\partial x_i} + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \\ &= -\frac{\partial \hat{p}}{\partial x_i} + (\rho - \rho_0) g_i + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \\ &= -\frac{\partial \hat{p}}{\partial x_i} - \rho_0 \beta (T - T_0) + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\end{aligned}$$

using $\rho \mathbf{g}$ as the mass specific force.

- Talk about natural and forced convection. Differences for the solver algorithm. (s.a.) Peric P447
- Talk about flows with variation in fluid properties -> mms has to map this behaviour (Buoyancy force driven, i.e. naturally convected fluid), mixed Convection
- Also talk about non-dimensional values like Prandtl number, Rayleigh and Reynolds
- Talk about the validity of this approximation

2.6 Final Form of the Set of Equations

In the previous subsections different simplifications have been introduced which will be used throughout the thesis. The final form of the set of equations to be used is thereby presented. As further simplification the modified pressure $\hat{p}$ will be treated as p and since the use of the Boussinesq-approximation replaces the variable ρ by a linear function of the temperature T the reference density ρ_0 for the remainder of this thesis will be referred to as ρ. Note that incompressibility has been taken into account:

$$\frac{\partial u_i}{\partial x_i} = 0. \tag{11a}$$

$$\rho \frac{\partial (u_i)}{\partial t} + \rho \frac{\partial}{\partial x_j}\left(u_i u_j\right) = -\frac{\partial p}{\partial x_i} - \rho \beta (T - T_0) + \mu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{11b}$$

$$\frac{\partial (\rho T)}{\partial t} + \frac{\partial}{x_j}\left(\rho u_j T - \kappa \frac{\partial T}{\partial x_j}\right) = q_T. \tag{11c}$$

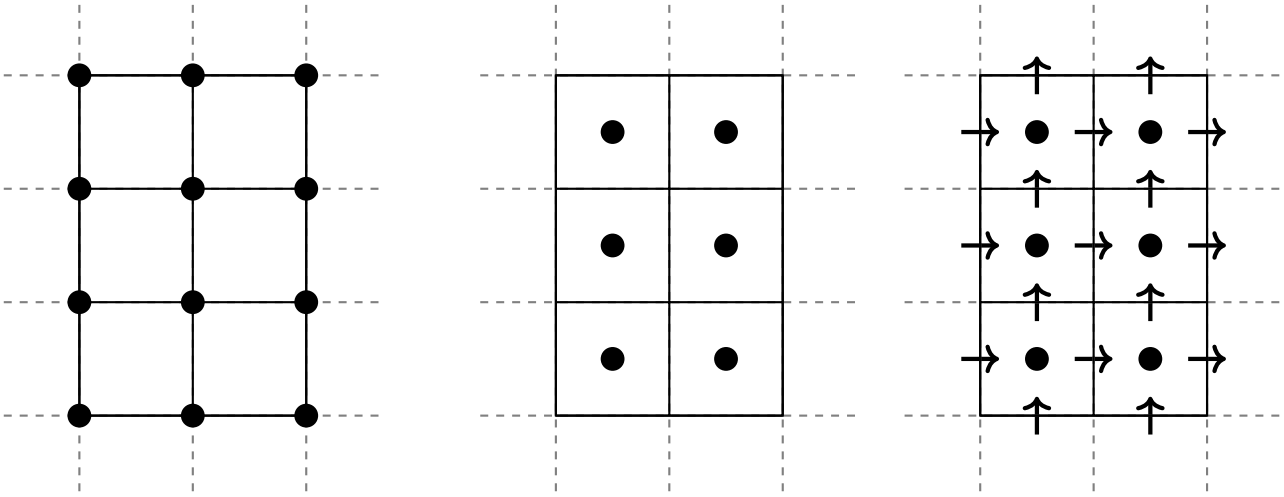

Figure 1: Vertex centered, cell centered and staggered variable arrangement

3 Finite Volume Methods for Incompressible Flows – Theoretical Basics

This section deals with the fundamentals of the numerical solution via a finite volume method of the formerly presented set of partial differential equations (11). The focus of this section is to provide an overview over the methods to be used in the present thesis. The information contained in this section is based on [**?**, **?**, **?**, **?**]. The overview starts by mentioning the different grid types to be used and the discretization techniques to be applied. On the basis of integral formulations of the equations to be solved, the therein contained integrals and differential operators have to be discretized. Since the accuracy of the default concepts for discretizing differential operators degrades with decreasing grid quality, this chapter furthermore presents different approaches to handle corrections for cases in which the cause of degrading grid quality is increased non-orthogonality.

The goal of the finite volume method is to provide linear algebraic equations which can be used to determine an approximate solution of a partial differential equation. This system of linear algebraic equations can be solved by means of algorithms to be presented in the end of this section. However since the Navier-Stokes equations are in general non-linear an intermediate step has to be taken, by linearizing the discrete equations. This leads to the need of an iteration process, the *Picard iteration*, which will also be explained briefly.

3.1 Numerical Grid

In this subsection a brief overview of the general grid structure to be used in the present thesis is given. The main idea behind finite volume methods is to solve partial differential equations by integrating them over the specified continuous problem domain and dividing this domain into a finite number of subdomains, the so called *control volumes*. The result of the this finite partition of a continuous problem domain is called the numerical grid. The grid consists of a finite number of control volumes which represent the boundaries of a discrete domain of integration. Depending on whether the numerical solution of an equation is to be calculated on the boundary vertices or in the center of a grid cell, the variable arrangement is denoted to be vertex or cell center oriented. For the remainder of the present thesis it is assumed that the variables reside in the cell centers of the respective control volumes.

Another option to arrange variables specifically for algorithms that solve the Navier-Stokes equations is the so called *staggering*. This method stores the variable values of different variables at different grid locations, which facilitates the interpolation of the respective values. As the methods of employed in the present thesis are intended to be generally applicable to complex geometries the cell centered non-staggered approach offers more flexibility. Figure 3.1 shows the different variable arrangement options on a two dimensional grid.

Regarding the treatment of domain boundaries and the ordering of the cells within the problem domain different types of numerical grids can be distinguished. The present thesis makes use of so called boundary fitted, block structured grids with hexahedron cells. A structured grid is characterized by a constant amount of of grid cells in each coordinate direction. The high regularity of structured grids benefits the computational efficiency of algorithms to be used on this type of grid. A block structured grid consists of different grid blocks of which each considered individually is structured, but if the topology of the grid is considered, it is unstructured. An example of a block structured grid with distinguishable grid blocks is given in figure 3.1. The use of block structured grids is motivated by the need to increase the adaptivity of structured grids while maintaining high computational efficiency. Furthermore it naturally embraces the concept of

domain decomposition which facilitates the implementation of parallel algorithms for the decomposed computational domain. Boundary fitted grids represent domain boundaries by means of the geometry of the control volumes residing on those boundaries. This approach may lead to skewed cells which furthermore affects the accuracy of the results obtained in these boundary cells. The mentioned affects can be alleviated by using local grid refinement techniques to refine the grid in regions of geometrically complex boundaries.

Inside a structured grid block, cells with the shape of hexahedrons are used. In addition to the geometric boundaries of each control volume a numerical grid also provides a mapping that assigns to each control volume with index P a set of indexes of neighbouring control volumes $NB(P) := \{W, S, B, T, N, E\}$, which are named after the geographic directions. Figure 3.1 shows a single grid cell with its direct neighbours. The faces $\{S_w, S_s, S, b, S_t, S_n, S_e\}$ of each hexahedral control volume represent the mentioned geometric boundaries.

- talk about grid quality
- talk about local refinement
- talk about variable arrangement

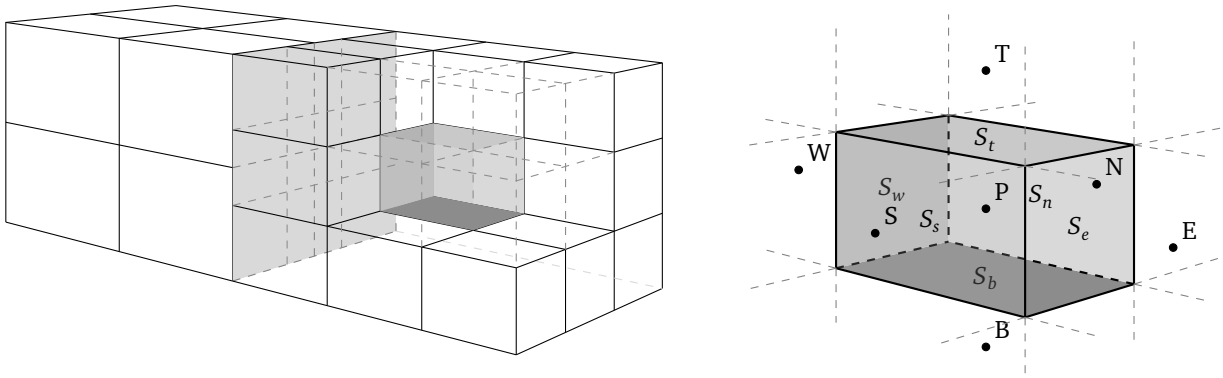

Figure 2: Block structured grid consisting of two blocks

3.2 Approximation of Integrals and Derivatives

In the course of transforming a partial differential equation into a system of linear algebraic equations, integrals and derivatives have to be approximated. The simplest method for approximating an integral is by using the *midpoint rule*. This rule is similar to the mean value theorem of integration, which states that there exists a point $\xi \in V$ for a Riemann integrable function ϕ such that $\phi(\xi)\int_V \mathrm{d}V = \int_V \phi(x)\mathrm{d}V$. For the midpoint rule ξ is taken to be the center of mass of V. If the integration domain V is indeed a volume, fortunately the calculation of $\phi(\xi)$ with $\xi := \left(\int_V x_i \mathrm{d}V / \int_V \mathrm{d}V\right)_{i=1,\dots,3}$ presents no difficulties since due to the collocated variable arrangement the value of ϕ is stored in the cell center, which corresponds to the location ξ. However if the domain of integration is a surface, a preceding interpolation step is necessary.

On the other hand to transform a partial differential equation into a linear algebraic equation it is necessary to discretize the differential operators of the equations. For numerical reasons two different discretization techniques are used in this thesis. A common task is to discretize expressions of the form

$$(\nabla\phi)_e \cdot \mathbf{n}_e,$$

where $(\nabla\phi)_e$ is the Gradient of ϕ on a boundary face S_e. One method is to directly interpret this expression as a directional derivative and approximate it with a central difference

$$(\nabla\phi)_e \cdot \mathbf{n}_e \approx \frac{\phi_P - \phi_E}{||\mathbf{x}_P - \mathbf{x}_E||_2}. \tag{12}$$

Another method would be to first calculate the cell center gradients $(\nabla\phi)_P$ and $(\nabla\phi)_E$ and interpolate them linearly before calculating the projection onto $\mathbf{n}_e$

$$(\nabla\phi)_e \cdot \mathbf{n}_e \approx \left[\gamma_e(\nabla\phi)_P + (1-\gamma_e)(\nabla\phi)_E\right] \cdot \mathbf{n}_e, \tag{13}$$

where $\gamma_e := ||\mathbf{x}_P - \mathbf{x}_e||_2 / ||\mathbf{x}_P - \mathbf{x}_E||_2$ is a geometric interpolation factor. For calculating the cell center gradients a method based on Gauss' integration theorem and the midpoint rule for volume integration is employed

$$(\nabla\phi)_{i,P} = \left(\frac{\partial\phi}{\partial x_i}\right)_P \approx \frac{\int_V \left(\frac{\partial\phi}{\partial x_i}\right)_P \mathrm{d}V}{|V|}. \tag{14}$$

Briefly explain the idea behind the quadrature via the midpoint rule. Talk about central differences and the approximation via the gauss theorem. Maybe talk about the resulting order of the truncation error.

3.3 Treatment of Non-Orthogonality of Grid Cells

Unfortunately real applications involve complex geometries, which in turn affects the orthogonality of the grid. On non-orthogonal meshes the directional derivative in direction of the face unit normal vector $\mathbf{n}_e$ can no longer be approximated as in (12). On the other side the exclusive usage of (14) is not desirable due to the bigger truncation error that comes with this approximation (PROOF?). Hence a compromise is made and the surface vector $\mathbf{S}_e := S_e\mathbf{n}_e$ is decomposed as

$$\mathbf{S}_e = \mathbf{\Delta} + \mathbf{k}, \tag{15}$$

where $\mathbf{\Delta}$ is parallel to the vector $\mathbf{d}_e := (\mathbf{x}_E - \mathbf{x}_P)$ that directly connects the center $\mathbf{x}_P$ of the control volume P with the center $\mathbf{x}_E$ of its neighbour E. This vector controls the *orthogonal* contribution to the directional derivative. The vector $\mathbf{k}$ controls the influence of the *non-orthogonal* contribution. In the next paragraphs the three main decompositions of the surface vector $\mathbf{S}_e$ will be presented by stating the respective expression for $\mathbf{\Delta}$. The resulting vector $\mathbf{k}$ can be calculated by using (15). One important characteristic that all of the presented approaches have is common, is that the non-orthogonal contribution vanishes, when an orthogonal grid is used. For simplicity the presentation of the decompositions is chosen to be two-dimensional. A geometrical interpretation of the tree approaches is given in 3.3.3. The last subsection handles the integration of one generic approach into the discretization process.

3.3.1 Minimum Correction Approach

This is the approach as proposed in [**?**]. The reader should note, that even though [**?**] references [**?**], they use a different approach to be presented in the next paragraph. This method is designed to keep the non-orthogonal contribution minimal by always choosing $\mathbf{k}$ to be orthogonal to $\mathbf{\Delta}$, which leads to

$$\mathbf{\Delta} = (\mathbf{d}\cdot\mathbf{S}_e)\frac{\mathbf{d}}{||\mathbf{d}||_2}.$$

It should be noted that the Influence of the orthogonal contribution decreases with increasing non-orthogonality of the grid.

3.3.2 Orthogonal Correction Approach

The following method for decomposing the surface normal vector is presented in [**?**] and the approach implemented in the developed solvers. In this approach a simple projection is used which is independent of the non-orthogonality of the grid. As a result the orthogonal contribution $||\mathbf{\Delta}||_2 = ||\mathbf{S}_e||_2$ and is thus modelled as

$$\mathbf{\Delta} = S_e\frac{\mathbf{d}}{||\mathbf{d}||_2}.$$

3.3.3 Over-Relaxed Approach

The last approach is used in [**?**, **?**] and is characterized by an increasing influence of the orthogonal contribution with increasing grid non-orthogonality, as opposed to the minimum correction approach. The orthogonal contribution is calculated as

$$\mathbf{\Delta} = S_e^2\frac{\mathbf{d}}{\mathbf{d}\cdot\mathbf{S}_e}.$$

3.3.4 Deferred Non-Orthogonal Correction

In order to reduce the computational stencil that would be necessary to handle the non-orthogonal correction implicitly the correction will be treated explicitly using a deferred correction which guarantees that in the case of a fully converged

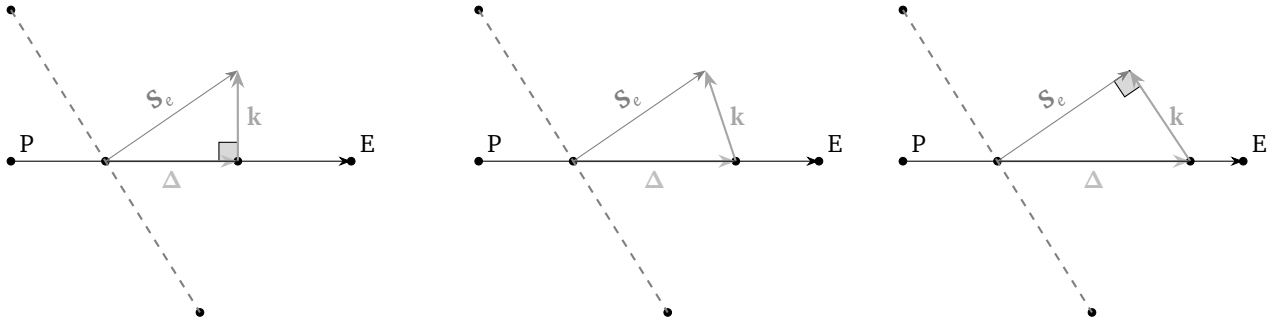

Figure 3: Minimum correction, orthogonal correction and over-relaxed approach

solution only the face normal derivative has been taken into account. Generally the discretization using a non-orthogonal correction would yield

$$(\nabla\phi)_e \cdot \mathbf{S}_e \approx (\nabla\phi)_e \cdot \boldsymbol{\Delta} + (\nabla\phi)_e \cdot \mathbf{k}.$$

Where the first term can be approximated using a central differencing scheme for the directional derivative and the second by interpolating cell center gradients. If one furthermore uses the fact that this method comes to play in a solution algorithm for a non-linear system of partial differential equations a deferred correction can be implemented which ensues a smaller error from the non-orthogonality. In the case of the previously mentioned discretization techniques for partial derivatives a possible deferred correction approach reads

$$(\nabla\phi)_e \cdot \mathbf{S}_e \approx ||\boldsymbol{\Delta}||_2 \frac{\phi_P - \phi_E}{||\mathbf{x}_P - \mathbf{x}_E||_2} - (\nabla\phi)_e^{(n-1)} \cdot (\boldsymbol{\Delta} - \mathbf{S}_e).$$

It should be noted that the use of a deferred correction in conjunction with the requirement that the non-orthogonal correction vanishes on orthogonal grid introduces an inconsistent discretization of $(\nabla\phi)_e \cdot \boldsymbol{\Delta}$.

3.4 Numerical Solution of Non-Linear Systems – Linearization Techniques

Introduce the concept of the Picard-Iteration as a linearization technique. Introduce the notions of inner and outer iterations. Refer to later chapters when it comes to deferred correction.

3.5 Numerical Solution of Linear Systems with Krylov Subspace Methods

??

- General concept of cyclic vector spaces of $\mathbb{R}^n$,
- talk about bases of krylov subspaces and the arnoldi algorithm, talk about polynomials and linear combinations
- mention the two major branches (minimum residual approach, petrov and ritz-galerkin approach)
- name some representative ksp algorithms, importance of preconditioning, not as detailed as in bachelor thesis
- in cases there is a nonempty Nullspace what happens?

4 Implicit Finite Volume Method for Incompressible Flows – Segregated Approach

The purpose of this section is to present the discretization applied to the set of equations (11). Since the system of partial differential equations to be solved always exhibits a coupling, at least between the dependent variables pressure and velocity, a first solution algorithm, namely the *SIMPLE* algorithm, addressed to resolve the pressure velocity coupling, is introduced. Furthermore an under-relaxation factor independent method of calculating mass fluxes by interpolation is introduced and the detailed derivation of all coefficients that result from the discretization process is presented. Finally the boundary conditions, that are relevant for the present thesis will be introduced.

4.1 Discretization of the Mass Balance

Integration of equation (11a) over the integration domain of a single control volume P, after the application of Gauss' integration theorem and the additivity of the Riemann integral, yields

$$\iint_{S_f} u_i n_i \mathrm{d}S = \sum_{f\in\{w,s,b,t,n,e\}} \iint_{S_f} u_i n_i \mathrm{d}S = 0.$$

In the present work the mass balance is discretized using the midpoint integration rule for the surface integrals and linear interpolation of the velocity to the center of mass of the surface. This leads to the following form of the mass balance:

$$\sum_{f\in\{w,s,b,t,n,e\}} u_{i,f} n_{f_i} S_f = 0, \tag{16}$$

where no interpolation to attain the values of u_i at the face S_f is performed yet, since the straightforward linear interpolation will lead to undesired oscillations in the solution fields. An interpolation method to circumvent this so called *checker boarding* effect is presented in subsection 4.2.

4.2 A Pressure-Weighted Interpolation Method for Velocities

The advantages of using a cell-centered variable arrangement are evident: The treatment of non-orthogonality is simplified and the conservation property of finite volume methods is retained [**?**, **?**, **?**, **?**]. A major drawback with cell centered variable arrangements is that pressure field may delink, which will then lead to unphysical oscillations in both the pressure and the velocity results. If the oscillations are severe enough the solution algorithm might even get unstable and diverge. The described decoupling occurs, when the pressure gradient in the momentum balances and the mass fluxes in the continuity equation are discretized using central differences.

A common practice to eliminate this behaviour is the use of a momentum interpolation technique, also known as *Rhie-Chow Interpolation* [**?**]. The original interpolation scheme however does not guarantee a unique solution, independent of the amount of under-relaxation. The performance of one of the algorithms that are used in the present thesis heavily relies on the under-relaxation of variables to accomplish stability. Furthermore the original method as proposed by [**?**] does not account for large body forces which also may lead to unphysical results. These issues will be addressed in this subsection which at the end will present an interpolation method that assures an under-relaxation independent solution: the *pressure-weighted interpolation method* [**?**].

The starting point of the pressure-weighted interpolation method is formed by the discretized momentum balances at node P and an arbitrary neighbouring node Q. The discretization for finite volume methods and details including the incorporation of under-relaxation factors will be handled in subsection 4.5. The semi-discrete implicit momentum balances, if one solves for the velocity at node P or Q, read

$$u_{i,P}^{(n)} = -\frac{\alpha_{\mathbf{u}_P}}{a_P^{u_i}} \left(\sum_{F\in NB(P)} a_F^{u_i} u_{i,F}^{(n)} + b_{P,u_i}^{(n-1)} - V_P \left(\frac{\partial p}{\partial x_i} \right)_P^{(n-1)} \right) + (1-\alpha_{\mathbf{u}}) u_{i,P}^{(n-1)} \tag{17a}$$

$$\text{and} \quad u_{i,Q}^{(n)} = -\frac{\alpha_{\mathbf{u}_Q}}{a_Q^{u_i}} \left(\sum_{F\in NB(Q)} a_F^{u_i} u_{i,F}^{(n)} + b_{Q,u_i}^{(n-1)} - V_Q \left(\frac{\partial p}{\partial x_i} \right)_Q^{(n-1)} \right) + (1-\alpha_{\mathbf{u}}) u_{i,Q}^{(n-1)} \quad , \tag{17b}$$

where the superscript $(n-1)$ denotes the previous outer iteration number. The reader should note, that the pressure gradient has not been discretized yet. This has the advantage that the selective interpolation technique [**?**] can be

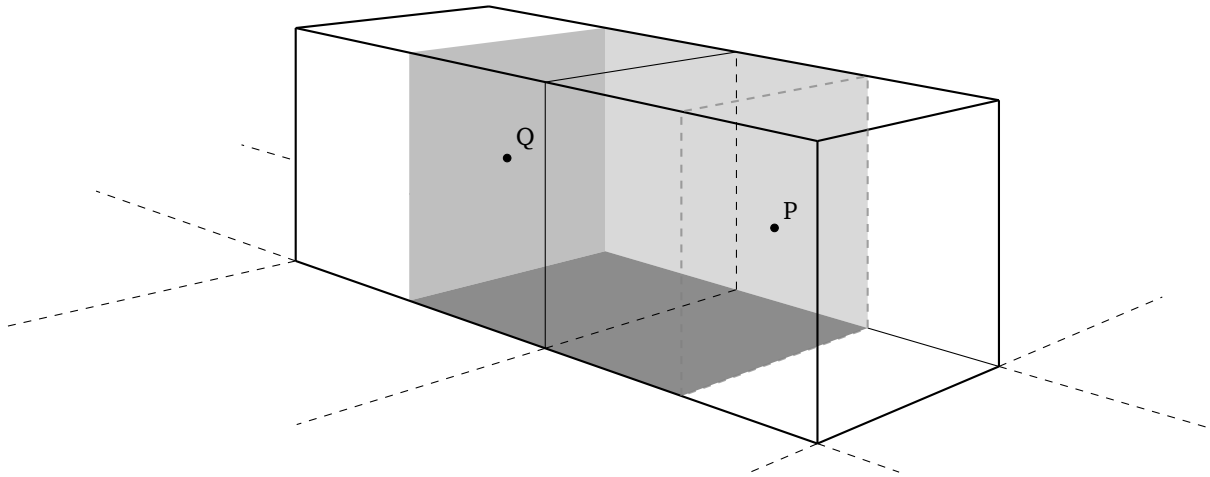

Figure 4: Possible interpretation of a virtual control volume (grey) located between nodes P and Q

applied, which is crucial for the elimination of the mentioned oscillations. In almost the same way a semi-discrete implicit momentum balance can be formulated for a virtual control volume located between nodes P and Q.

$$u_{i,f}^{(n)} = -\frac{\alpha_{\mathbf{u}_f}}{a_f^{u_i}} \left(\sum_{F \in NB(f)} a_F^{u_i} u_{i,F}^{(n)} + b_{f,u_i}^{(n-1)} - V_f \left(\frac{\partial p}{\partial x_i} \right)_f^{(n-1)} \right) + (1-\alpha_{\mathbf{u}}) u_{i,f}^{(n-1)}. \tag{18}$$

Figure 4.2 gives an interpretation of the virtual control volume. To guarantee convergence of this expression for $u_{i,f}$, under-relaxation is necessary [**?**]. To eliminate the undefined artifacts surging form the virtualization of a control volume the following assumptions have to be made to derive a closed expression for the velocity on the boundary face S_f

$$\frac{\alpha_{\mathbf{u}_f}}{a_f^{u_i}} \left(\sum_{F \in NB(f)} a_F^{u_i} u_{i,F}^{(n)} \right) \approx (1-\gamma_f) \frac{\alpha_{\mathbf{u}_P}}{a_P^{u_i}} \left(\sum_{F \in NB(P)} a_F^{u_i} u_{i,F}^{(n)} \right) + \gamma_f \frac{\alpha_{\mathbf{u}_Q}}{a_Q^{u_i}} \left(\sum_{F \in NB(Q)} a_F^{u_i} u_{i,F}^{(n)} \right), \tag{19a}$$

$$\frac{\alpha_{\mathbf{u}_f}}{a_f^{u_i}} b_{f,u_i}^{(n-1)} \approx (1-\gamma_f) \frac{\alpha_{\mathbf{u}_P}}{a_P^{u_i}} b_{P,u_i}^{(n-1)} + \gamma_f \frac{\alpha_{\mathbf{u}}}{a_Q^{u_i}} b_{Q,u_i}^{(n-1)} \tag{19b}$$

$$\text{and} \quad \frac{\alpha_{\mathbf{u}}}{a_f^{u_i}} \approx (1-\gamma_f) \frac{\alpha_{\mathbf{u}}}{a_P^{u_i}} + \gamma_f \frac{\alpha_{\mathbf{u}}}{a_Q^{u_i}}, \tag{19c}$$

$$\tag{19d}$$

where γ_f is a geometric interpolation factor.

Using the assumptions made in equation (19) the expression in equation (18) can be closed in a way that it only depends on the variable values in node P and Q

$$\begin{aligned} u_{i,f}^{(n)} \approx{}& (1-\gamma_f) \left(-\frac{\alpha_{\mathbf{u}}}{a_P^{u_i}} \sum_{F \in NB(P)} a_F^{u_i} u_{i,F}^{(n)} \right) + \gamma_f \left(-\frac{\alpha_{\mathbf{u}}}{a_Q^{u_i}} \sum_{F \in NB(Q)} a_F^{u_i} u_{i,F}^{(n)} \right) \\ &+ \frac{\alpha_{\mathbf{u}}}{a_f^{u_i}} b_{f,u_i}^{(n-1)} - \frac{\alpha_{\mathbf{u}}}{a_f^{u_i}} V_f \left(\frac{\partial p}{\partial x_i} \right)_f^{(n-1)} + (1-\alpha_{\mathbf{u}}) u_{i,f}^{(n-1)} \\ ={}& (1-\gamma_f) u_{i,P}^{(n)} - (1-\gamma_f) \frac{\alpha_{\mathbf{u}}}{a_P^{u_i}} \left(b_{P,u_i}^{(n-1)} - V_P \left(\frac{\partial p}{\partial x_i} \right)_P^{(n-1)} \right) + \gamma_f u_{i,Q}^{(n)} - \gamma_f \frac{\alpha_{\mathbf{u}}}{a_Q^{u_i}} \left(b_{Q,u_i}^{(n-1)} - V_Q \left(\frac{\partial p}{\partial x_i} \right)_Q^{(n-1)} \right) \\ &+ \frac{\alpha_{\mathbf{u}}}{a_f^{u_i}} b_{f,u_i}^{(n-1)} - \frac{\alpha_{\mathbf{u}}}{a_f^{u_i}} V_f \left(\frac{\partial p}{\partial x_i} \right)_f^{(n-1)} + (1-\alpha_{\mathbf{u}}) u_{i,f}^{(n-1)} \end{aligned}$$

$$
\begin{aligned}
&= \Big[\big(1-\gamma_f\big)u_{i,P}^{(n)} + \gamma_f u_{i,Q}^{(n)}\Big] \\
&\quad - \left[\left(\big(1-\gamma_f\big)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\right)\left(\frac{\partial p}{\partial x_i}\right)_f^{(n-1)} - \big(1-\gamma_f\big)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}}\left(\frac{\partial p}{\partial x_i}\right)_P^{(n-1)} - \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\left(\frac{\partial p}{\partial x_i}\right)_Q^{(n-1)}\right] \\
&\quad + (1-\alpha_{\mathbf{u}})\Big[u_{i,f}^{(n-1)} - \big(1-\gamma_f\big)u_{i,P}^{(n-1)} - \gamma_f u_{i,Q}^{(n-1)}\Big] \\
&\approx \Big[\big(1-\gamma_f\big)u_{i,P}^{(n)} + \gamma_f u_{i,Q}^{(n)}\Big] \\
&\quad - \left(\big(1-\gamma_f\big)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\right)\left[\underline{\left(\frac{\partial p}{\partial x_i}\right)_f^{(n-1)} - \big(1-\gamma_f\big)\left(\frac{\partial p}{\partial x_i}\right)_P^{(n-1)} - \gamma_f\left(\frac{\partial p}{\partial x_i}\right)_Q^{(n-1)}}\right] \\
&\quad + (1-\alpha_{\mathbf{u}})\Big[u_{i,f}^{(n-1)} - \big(1-\gamma_f\big)u_{i,P}^{(n-1)} - \gamma_f u_{i,Q}^{(n-1)}\Big].
\end{aligned} \tag{20}
$$

It should be noted that the argumentation that led to the last expression, is that the task of the underlined pressure gradient corrector in equation (20) is to suppress oscillations in the converged solution for the pressure field. If there are no oscillations this part should not become active. As long as the behaviour of this corrector remains consistent, i.e. that there are no oscillations in the pressure field, it can be multiplied with arbitrary constants [**?**]. This is however true on equidistant grids, where $\gamma_f = 1/2$ and central differences are used to calculate the gradients. On arbitrary orthogonal grids another modification has to be performed which is based on a special case of the mean value theorem of differential calculus and the following

Proposition. *Let $x_1, x_2 \in \mathbb{R}$ with $x_1 \neq x_2$ and $p(x) = a_0 + a_1 x + a_2 x^2$ a real polynomial function. Then*

$$
\frac{dp}{dx}\left(\frac{x_1+x_2}{2}\right) = \frac{p(x_2)-p(x_1)}{x_2-x_1},
$$

i.e. the slope of the secant equals the value of the first derivative of p exactly half the way between x_1 and x_2.

Proof. Evaluation of the derivative yields

$$
\frac{dp}{dx}\left(\frac{x_1+x_2}{2}\right) = a_1 + 2a_2\frac{x_1+x_2}{2} = a_1 + a_2(x_1+x_2).
$$

On the other hand the slope of the secant, using the third binomial rule can be expressed as

$$
\begin{aligned}
\frac{p(x_2)-p(x_1)}{x_2-x_1} &= \frac{a_0 + a_1x_2 + a_2x_2^2 - \big(a_0 + a_1x_1 + a_2x_1^2\big)}{x_2-x_1} \\
&= \frac{a_1(x_2-x_1) + a_2\big(x_2^2-x_1^2\big)}{x_2-x_1} \\
&= a_1 + a_2(x_2+x_1).
\end{aligned}
$$

The comparison of both expressions completes the proof. □

It is desirable for the pressure corrector to vanish independent of the grid spacing if the profile of the pressure is quadratic and hence does not exhibit oscillations. According to the preceding proposition this can be accomplished by modifying equation (20) to average the pressure gradients from node P and Q instead of interpolating linearly

$$
\begin{aligned}
u_{i,f}^{(n)} &= \Big[\big(1-\gamma_f\big)u_{i,P}^{(n)} + \gamma_f u_{i,Q}^{(n)}\Big] \\
&\quad - \left(\big(1-\gamma_f\big)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\right)\left[\left(\frac{\partial p}{\partial x_i}\right)_f^{(n-1)} - \frac{1}{2}\left(\frac{\partial p}{\partial x_i}\right)_P^{(n-1)} - \frac{1}{2}\left(\frac{\partial p}{\partial x_i}\right)_Q^{(n-1)}\right] \\
&\quad + \underline{(1-\alpha_{\mathbf{u}})\Big[u_{i,f}^{(n-1)} - \big(1-\gamma_f\big)u_{i,P}^{(n-1)} - \gamma_f u_{i,Q}^{(n-1)}\Big]}.
\end{aligned} \tag{21}
$$

Comparing this final expression with the standard interpolation scheme, it is evident that normally, the underlined term is not taken into consideration [**?**]. However section 7.4 shows, that neglecting this term indeed creates under-relaxation factor dependent results. This section concludes with a final

Proposition. *The pressure weighted momentum interpolation scheme (21) guarantees the converged solution for $u_{i,f}$ to be independent of the velocity under-relaxation $\alpha_{\mathbf{u}}$.*

Proof. An equivalent formulation of (21) is given by

$$\begin{aligned}\alpha_{\mathbf{u}} u_{i,f}^{(n-1)} + u_{i,f}^{(n-1)} - u_{i,f}^{(n)} = \alpha_{\mathbf{u}} \Big[\big(1-\gamma_f\big) u_{i,P}^{(n-1)} + \gamma_f \, u_{i,Q}^{(n-1)} \Big] \\ + \Big[\big(1-\gamma_f\big) \Big(u_{i,P}^{(n)} - u_{i,P}^{(n-1)} \Big) + \gamma_f \left(u_{i,Q}^{(n)} - u_{i,Q}^{(n-1)} \right) \Big] \\ - \alpha_{\mathbf{u}} \left(\big(1-\gamma_f\big) \frac{V_P}{a_P^{u_i}} + \gamma_f \frac{V_Q}{a_Q^{u_i}} \right) \left[\left(\frac{\partial p}{\partial x_i} \right)_f^{(n-1)} - \frac{1}{2} \left(\frac{\partial p}{\partial x_i} \right)_P^{(n-1)} - \frac{1}{2} \left(\frac{\partial p}{\partial x_i} \right)_Q^{(n-1)} \right].\end{aligned}$$

Upon convergence $u_{i,P}^{(n)} = u_{i,P}^{(n-1)}$ and $u_{i,f}^{(n)} = u_{i,f}^{(n-1)}$. This leads to

$$\begin{aligned}\alpha_{\mathbf{u}} u_{i,f}^{(n-1)} = \alpha_{\mathbf{u}} \Big[\big(1-\gamma_f\big) u_{i,P}^{(n-1)} + \gamma_f \, u_{i,Q}^{(n-1)} \Big] \\ - \alpha_{\mathbf{u}} \left(\big(1-\gamma_f\big) \frac{V_P}{a_P^{u_i}} + \gamma_f \frac{V_Q}{a_Q^{u_i}} \right) \left[\left(\frac{\partial p}{\partial x_i} \right)_f^{(n-1)} - \frac{1}{2} \left(\frac{\partial p}{\partial x_i} \right)_P^{(n-1)} - \frac{1}{2} \left(\frac{\partial p}{\partial x_i} \right)_Q^{(n-1)} \right],\end{aligned}$$

which shows, after division by $\alpha_{\mathbf{u}} > 0$, that $u_{i,f}$ is independent of the under-relaxation factor. □

4.3 Implicit Pressure Correction and the SIMPLE Algorithm

The goal of finite volume methods is to deduce a system of linear algebraic equations from a partial differential equation. In the case of the momentum balances the general structure of this linear equations is

$$u_{i,P}^{(n)} = -\frac{\alpha_{\mathbf{u}_P}}{a_P^{u_i}} \left(\sum_{F \in NB(P)} a_F^{u_i} u_{i,F}^{(n)} + b_{P,u_i}^{(n-1)} - V_P \left(\frac{\partial p}{\partial x_i} \right)_P^{(n-1)} \right) + (1-\alpha_{\mathbf{u}}) u_{i,P}^{(n-1)}, \tag{22}$$

where the pressure gradient has been discretized only symbolically and $b_{P,u_i}^{(n-1)}$ denotes the source term evaluated at the previous outer iteration.

At this stage the equations are still coupled and non-linear. As described in section 3.4 the Picard iteration process can be used to linearize the equations. Every momentum balance equation then only depends on the one dominant variable u_i. Furthermore the coupling of the momentum balances through the convective term $(u_i u_j)$ is resolved in the process of linearization. The decoupled momentum balances can then be solved sequentially for the dominant variable u_i. All coefficients $a_{\{P,F\}}^{u_i}$, the source term and the pressure gradient will be evaluated explicitly by using results of the preceding outer iteration $(n-1)$. For the pressure gradient this means to take the pressure of the antecedent outer iteration as a first guess for the following iteration. This guess has to be corrected in an iterative way until all the non-linear equations are fulfilled up to a certain tolerance. Section (6.4.2) presents a suitable convergence criterion and its implementation. This linearization process in conjunction with the pressure guess leads to the linear equation

$$u_{i,P}^{(n*)} = -\frac{\alpha_{\mathbf{u}}}{a_P^{u_i}} \left(\sum_{F \in NB(P)} a_F^{u_i} u_{i,F}^{(n*)} + b_{P,u_i}^{(n-1)} - V_P \left(\frac{\partial p}{\partial x_i} \right)_P^{(n-1)} \right) + (1-\alpha_{\mathbf{u}}) u_{i,P}^{(n-1)}. \tag{23}$$

Here $(*)$ indicates that the solution of this equation still needs to be corrected to also fulfill the discretized mass balance

$$\sum_{F \in NB(P)} (u_i)_f^{(n)} n_i S_f = 0. \tag{24}$$

Applying the same procedure as in section 4.2 to equation (23) results in the following expression for the face velocities after solving the discretized momentum balances using as pressure guess the pressure from the previous outer iteration

$$u_{i,f}^{(n*)} = \left[\left(1-\gamma_f\right)u_{i,P}^{(n*)} + \gamma_f u_{i,Q}^{(n*)}\right] - \left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\right)\left[\left(\frac{\partial p}{\partial x_i}\right)_f^{(n-1)} - \left(1-\gamma_f\right)\left(\frac{\partial p}{\partial x_i}\right)_P^{(n-1)} - \gamma_f\left(\frac{\partial p}{\partial x_i}\right)_Q^{(n-1)}\right] + (1-\alpha_{\mathbf{u}})\left[u_{i,f}^{(n-1)} - \left(1-\gamma_f\right)u_{i,P}^{(n-1)} - \gamma_f u_{i,Q}^{(n-1)}\right]. \tag{25}$$

The lack of an equation with the pressure as dominant variable leads to the necessity to alter the mass balance as the only equation left. Methods of this type are called projection methods. A common class of algorithms of this family of methods uses an equation for the additive pressure correction p' instead of the pressure itself and enforces continuity by correcting the velocities with an additive corrector u_i', such that

$$u_{i,P}^{(n)} = u_{i,P}^{(n*)} + u_{i,P}', \quad u_{i,f}^{(n)} = u_{i,f}^{(n*)} + u_{i,f}' \quad \text{and} \quad p_P^{(n)} = p_P^{(n-1)} + p_P'.$$

It is now possible to formulate the discretized momentum balance for the corrected velocities and the corrected pressure as

$$u_{i,P}^{(n)} = -\frac{\alpha_{\mathbf{u}}}{a_P^{u_i}}\left(\sum_{F\in NB(P)} a_F^{u_i}u_{i,F}^{(n)} + b_{P,u_i}^{(n-1)} - V_P\left(\frac{\partial p}{\partial x_i}\right)_P^{(n)}\right) + (1-\alpha_{\mathbf{u}})u_{i,P}^{(n-1)}. \tag{26}$$

It should be noted that the only difference to the equation which will be solved in the next outer iteration is that the source term b_{P,u_i} has not been updated yet. In the same way the discretized momentum balance for the face velocity $u_{i,f}$ can be formulated as

$$u_{i,f}^{(n)} = \left[\left(1-\gamma_f\right)u_{i,P}^{(n)} + \gamma_f u_{i,Q}^{(n)}\right] - \left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\right)\left[\left(\frac{\partial p}{\partial x_i}\right)_f^{(n)} - \left(1-\gamma_f\right)\left(\frac{\partial p}{\partial x_i}\right)_P^{(n)} - \gamma_f\left(\frac{\partial p}{\partial x_i}\right)_Q^{(n)}\right] + (1-\alpha_{\mathbf{u}})\left[u_{i,f}^{(n-1)} - \left(1-\gamma_f\right)u_{i,P}^{(n-1)} - \gamma_f u_{i,Q}^{(n-1)}\right]. \tag{27}$$

To couple velocity and pressure correctors one can subtract equation (23) from (26) and equation (25) from (27) to get

$$u_{i,P}' = -\frac{\alpha_{\mathbf{u}}}{a_P^{u_i}}\left(\underline{\sum_{F\in NB(P)} a_F^{u_i}u_{i,F}'} - V_P\left(\frac{\partial p'}{\partial x_i}\right)_P^{(n)}\right) \quad \text{and} \tag{28}$$

$$u_{i,f}' = \left[\left(1-\gamma_f\right)u_{i,P}' + \gamma_f u_{i,Q}'\right] - \left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\right)\left[\left(\frac{\partial p}{\partial x_i}\right)_f' - \left(1-\gamma_f\right)\left(\frac{\partial p}{\partial x_i}\right)_P' - \gamma_f\left(\frac{\partial p}{\partial x_i}\right)_Q'\right]. \tag{29}$$

The majority of the class of pressure correction algorithms has this equations as a common basis. Each algorithm then introduces special distinguishable approximations of the velocity corrections that are, at the moment of solving the pressure equation, still unknown. The method used in the present work is the SIMPLE Algorithm (Semi-Implicit Method for Pressure-Linked Equations [**?**]). The approximation this algorithm performs is severe since the term containing the unknown velocity corrections is dropped entirely. The respective term has been underlined in equation (28). Since the global purpose of the presented method is to enforce continuity by implicitly calculating a pressure correction, the velocity correction has to be expressed solely in terms of the pressure correction. This can be accomplished by inserting equation (28) into equation (29). This gives an update formula

$$u_{i,f}' = -\left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}}V_P}{a_P^{u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_Q^{u_i}}\right)\left(\frac{\partial p}{\partial x_i}\right)_f', \tag{30}$$

which is then, together with (25), inserted into the discretized continuity equation (24) to obtain

$$\sum_{F\in NB(P)} \left(\left(1-\gamma_f\right) \frac{\alpha_{\mathbf{u}} V_P}{a_P^{u_i}} + \gamma_f \frac{\alpha_{\mathbf{u}} V_F}{a_F^{u_i}} \right) \left(\frac{\partial p}{\partial x_i} \right)_f' n_i S_f = b_{p,p} \quad , \tag{31}$$

where the right hand side $b_{p,p}$ is defined as

$$b_{p,p} := \sum_{F\in NB(P)} u_{i,f}^{(n*)} n_i S_f . \tag{32}$$

The complete discretization with central differences as approximation for the gradient of the pressure correction is straightforward and will be presented in subsection 4.4.

The approximation performed in the SIMPLE algorithm affects convergence in a way that the pressure correction has to be under-relaxed with a parameter $\alpha_p \in [0,1]$

$$p_P^{(n)} = p_P^{(n-1)} + \alpha_{\mathbf{p}} p_P' . \tag{33}$$

It should be noted that there are better approximations for the pressure correction which will not rely on pressure under-relaxation for convergence. On example that uses a more consistent approximation is the *SIMPLEC* algorithm (SIMPLE Consistent) [**?**]. A similar derivation of the pressure weighted interpolation method for the SIMPLEC algorithm can be found in [**?**].

As shown in section 4.2 the behaviour of the pressure weighted interpolation method on non-equidistant grids can be improved by replacing the linear interpolation of pressure gradients with a simple average in equation (25). This leads to the following equation for calculating mass fluxes

$$\begin{aligned} u_{i,f}^{(n*)} = & \left[\left(1-\gamma_f\right) u_{i,P}^{(n*)} + \gamma_f u_{i,Q}^{(n*)} \right] \\ & - \left(\left(1-\gamma_f\right) \frac{\alpha_{\mathbf{u}} V_P}{a_P^{u_i}} + \gamma_f \frac{\alpha_{\mathbf{u}} V_Q}{a_Q^{u_i}} \right) \left[\left(\frac{\partial p}{\partial x_i} \right)_f^{(n-1)} - \frac{1}{2} \left(\frac{\partial p}{\partial x_i} \right)_P^{(n-1)} - \frac{1}{2} \left(\frac{\partial p}{\partial x_i} \right)_Q^{(n-1)} \right] \\ & + (1-\alpha_{\mathbf{u}}) \left[u_{i,f}^{(n-1)} - \left(1-\gamma_f\right) u_{i,P}^{(n-1)} - \gamma_f u_{i,Q}^{(n-1)} \right] . \end{aligned} \tag{34}$$

Generally the SIMPLE algorithm can be represented by an iterative procedure as shown in Algorithm 5.1.

Algorithm 1 SIMPLE Algorithm

INITIALIZE variables
while (convergence criterion not accomplished) **do**
  SOLVE linearized momentum balances, equation **(23)**
  CALCULATE mass fluxes using **(27)** or **(34)**
  SOLVE pressure correction equation to assure continuity, equation **(31)**
  UPDATE pressure using **(33)**
  UPDATE velocities and mass fluxes using **(28)**
  if (Coupled scalar equation) **then**
    SOLVE scalar equation as described in **(4.6)**
  end if
end while

4.4 Discretization of the Mass Fluxes and the Pressure Correction Equation

Subsections 4.2 and 4.3 introduced the concept of pressure weighted interpolation to avoid oscillating results and an algorithm to calculate a velocity field that obeys continuity. The derived equations have not been discretized completely, furthermore the approach has not been generalized to non-orthogonal grids.

The discretized mass balance (16) only depends on the normal velocities $u_{i,f}\,n_{i,f}$. By analogy with equation (34) an interpolated normal face velocity and thus the mass flux can be calculated as

$$
\begin{aligned}
u_{i,f}^{(n*)} n_{i,f} = &\left[\left(1-\gamma_f\right) u_{i,P}^{(n*)} + \gamma_f u_{i,Q}^{(n*)}\right] n_{i,f} \\
&- \left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}} V_P}{a_P^{u_i}} + \gamma_f \frac{\alpha_{\mathbf{u}} V_Q}{a_Q^{u_i}}\right)\left[\left(\frac{\partial p}{\partial n}\right)_f^{(n-1)} - \frac{1}{2}\left(\frac{\partial p}{\partial n}\right)_P^{(n-1)} - \frac{1}{2}\left(\frac{\partial p}{\partial n}\right)_Q^{(n-1)}\right] \\
&+ (1-\alpha_{\mathbf{u}})\left[u_{i,f}^{(n-1)} - \left(1-\gamma_f\right) u_{i,P}^{(n-1)} - \gamma_f\, u_{i,Q}^{(n-1)}\right] n_{i,f},
\end{aligned}
\tag{35}
$$

where the scalar product of pressure gradients and the normal vector has been replaced by a directional derivative in the direction of the face normal vector. In the present work pressure gradients in (35) and pressure correction gradients in equation (31) will be discretized by central differences

$$
\left(\frac{\partial p}{\partial n}\right)_f \approx \frac{p_P - p_Q}{\left(\mathbf{x}_P - \mathbf{x}_Q\right)\cdot \mathbf{n}_f} \quad \text{and} \quad \left(\frac{\partial p'}{\partial n}\right)_f \approx \frac{p'_P - p'_Q}{\left(\mathbf{x}_P - \mathbf{x}_Q\right)\cdot \mathbf{n}_f}.
$$

This discretization can then be inserted into the semi-discretized pressure correction equation (31)

$$
\sum_{F\in NB(P)} \left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}} V_P}{a_P^{u_i}} + \gamma_f \frac{\alpha_{\mathbf{u}} V_F}{a_F^{u_i}}\right) \frac{p'_P - p'_F}{(\mathbf{x}_P - \mathbf{x}_F)\cdot \mathbf{n}_f} S_f = b_{P,p}. \tag{36}
$$

The resulting coefficients for the pressure correction equation

$$
a_P^{p'} p'_P + \sum_{F\in NB(P)} a_F^{p'} p'_F = b_{P,p'},
$$

can be calculated as

$$
a_F^{p'} = -\left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}} V_P}{a_P^{u_i}} + \gamma_f \frac{\alpha_{\mathbf{u}} V_F}{a_F^{u_i}}\right) \frac{S_f}{(\mathbf{x}_P - \mathbf{x}_F)\cdot \mathbf{n}_f} \quad \text{and} \quad a_P^{p'} = -\sum_{F\in NB(P)} a_F^{p'}. \tag{37}
$$

The right hand side can be calculated as in equation (32), if the presented discretization is applied.

4.5 Discretization of the Momentum Balance

The stationary momentum balance integrated over a single control volume P reads as

$$
\underbrace{\iint_S (\rho u_i u_j) n_j \mathrm{d}S}_{\text{convective term}} - \underbrace{\iint_S \left(\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) n_j \mathrm{d}S}_{\text{diffusive term}} = - \underbrace{\iiint_V \frac{\partial p}{\partial x_i}\mathrm{d}V}_{\text{pressure sourceterm}} - \underbrace{\iiint_V \rho\beta\left(T - T_0\right)\mathrm{d}V}_{\text{temperature sourceterm}}, \tag{38}
$$

where the different terms to be addressed individually in the following sections are indicated. The reader should note that the form of this equation has been modified by using Gauss' integration theorem. The terms residing on the left will be treated in an implicit and due to deferred corrections also in an explicit way, whereas the terms on the right will be treated exclusively in an explicit way.

4.5.1 Linearization and Discretization of the Convective Term

The convective term $\left(\rho u_i u_j\right)$ of the Navier-Stokes equations is the reason for the non-linearity of the equations. In order to deduce a set of linear algebraic equations from the Navier-Stokes equations this term has to be linearized. As introduced in section (3.4), the non linearity will be dealt with by means of an iterative process, the Picard iteration. The part dependent on the non dominant dependent variable therefore will be approximated by its value from the previous iteration as $\rho u_i^{(n)} u_j^{(n)} \approx \rho u_i^{(n)} u_j^{(n-1)}$. However this linearization will not be directly visible because it will be covered by the mass flux $\dot{m}_f = \iint_{S_f} \rho u_j^{(n-1)} n_j \mathrm{d}S$.

Using the additivity of the Riemann integral the first step is to decompose the surface integral into individual contributions from each boundary face of the control volume P

$$\iint_S \rho u_i u_j n_j \mathrm{d}S = \sum_{f\in\{w,s,b,t,n,e\}} \iint_{S_f} \rho u_i u_j n_j \mathrm{d}S = \sum_{f\in\{w,s,b,t,n,e\}} F^c_{i,f},$$

where $F^c_{i,f} := \iint_{S_f} \rho u_i^{(n)} u_j^{(n-1)} n_j \mathrm{d}S$ is the convective flux of the velocity u_i through the boundary face S_f.

To improve diagonal dominance of the resulting linear system while maintaining the smaller discretization error of a higher order discretization, a blended discretization scheme is applied and combined with a deferred correction. Since due to the non-linearity of the equations to be solved, an iterative solution process is needed by all means, the overall convergence does not degrade noticeably when using a deferred correction [**?**]. Blending and deferred correction result in a decomposition of the convective flux into a lower order approximation, which is treated implicitly, and the explicit difference between the higher and lower order approximation for the same convective flux. Since for coarse grid resolutions the use of higher order approximations may lead to oscillations of the solution, which in turn may degrade or even impede convergence, the schemes can be blended by a control factor $\eta \in [0,1]$. Furthermore the use of low-order approximations increases the diagonal dominance of the matrix, which then benefits the convergence of iterative solvers for the linear systems [**?**].

To show the generality of this approach all further derivations are presented for the generic boundary face S_f that separates control volume P from its neighbour $F \in NB(P)$. This decomposition then leads to

$$F^c_{i,f} \approx \underbrace{F^{c,l}_{i,f}}_{\text{implicit}} + \eta \underbrace{\left[F^{c,h}_{i,f} - F^{c,l}_{i,f}\right]^{(n-1)}}_{\text{explicit}}.$$

It should be noted that the convective fluxes carrying an l for *lower* or an h for *higher* as exponent, already have been linearized and discretized. The discretization applied to the convective flux in the present work is using the midpoint integration rule and blends the upwind interpolation scheme with a linear interpolation scheme. Applied to above decomposition one can derive the following approximations

$$\begin{aligned} F^{c,l}_{i,f} &= u_{i,F} \min(\dot{m}_f, 0) + u_{i,P} \max(0, \dot{m}_f) \\ F^{c,h}_{i,f} &= u_{i,F}\, \gamma_f + u_{i,P}\, (1-\gamma_f), \end{aligned}$$

where the variable values have to be taken from the previous iteration step $(n-1)$ as necessary and the mass flux $\dot{m}_f$ has been used as result of the linearization process. The results can now be summarized by presenting the convective contribution to the matrix coefficients a_{F,u_i} and $a_P^{u_i}$ and the right hand side b_{P,u_i} which are calculated as

$$a_F^{u_i,c} = \min(\dot{m}_f, 0), \qquad a_P^{u_i,c} = \sum_{F\in NB(P)} \max(0, \dot{m}_f) \tag{39a}$$

$$\text{and} \quad b^c_{P,u_i} = \sum_{F\in NB(P)} \eta\left(u^{(n-1)}_{i,F}\left(\min(\dot{m}_f,0) - \gamma_f\right)\right) + \eta\left(u^{(n-1)}_{i,P}\left(\max(0,\dot{m}_f) - \left(1-\gamma_f\right)\right)\right). \tag{39b}$$

4.5.2 Discretization of the Diffusive Term

The diffusive term contains the first partial derivatives of the velocity as a result of the material constitutive equation that characterizes the behaviour of Newtonian fluids. As pointed out in section 3.3, directional derivatives can be discretized using central differences on orthogonal grids or in the more general case of non-orthogonal grids using central differences implicitly and an explicit deferred correction comprising the non-orthogonality of the grid. As seen in equation (8) the diffusive term of the Navier-Stokes equations can be simplified using the mass balance in the case of an incompressible flow with constant viscosity μ. To sustain the generality of the presented approach this simplification will be omitted.

As before, by using the additivity and furthermore linearity of the Riemann integral, the integration of the diffusive term will be divided into integration over individual boundary faces S_f

$$\iint_S \left(\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\right) n_j \mathrm{d}S\mathrm{d}S = \sum_{f\in\{w,s,b,t,n,e\}} \left[\iint_{S_f} \mu\underline{\frac{\partial u_i}{\partial x_j} n_j \mathrm{d}S} + \iint_{S_f} \mu\frac{\partial u_j}{\partial x_i} n_j \mathrm{d}S\right] = \sum_{f\in\{w,s,b,t,n,e\}} F^d_{i,f},$$

where $F^d_{i,f}$ denotes the diffusive flux through an individual boundary face. Section 3.3 only covered the non-orthogonal corrector for directional derivatives. Since the velocity is a vector field and not a scalar field, the results of section 3.3 may only be applied to the underlined term. The other term will be treated explicitly since it is considerably smaller than the underlined term, furthermore does not cause oscillations and thus will not derogate convergence [**?**]. First all present integrals will be approximated using the midpoint integration rule. The diffusive flux $F^d_{i,f}$ for a generic face S_f located between the control volumes P and F then reads

$$F^d_{i,f} \approx \underline{\mu\left(\frac{\partial u_i}{\partial x_j}\right)_f n_j S_f} + \mu\left(\frac{\partial u_j}{\partial x_i}\right)_f n_j S_f.$$

Using central differences for the implicit discretization of the directional derivative and furthermore using the *orthogonal correction* approach from 3.3.2 the approximation can be derived as

$$\begin{aligned} F^d_{i,f} &\approx \mu\left(\underline{||\boldsymbol{\Delta}_f||_2 \frac{u_{P_i}-u_{F_i}}{||\mathbf{x}_P-\mathbf{x}_F||_2} - (\nabla u_i)^{(n-1)}_f \cdot (\boldsymbol{\Delta}_f - \mathbf{S}_f)}\right) + \mu\left(\frac{\partial u_j}{\partial x_i}\right)^{(n-1)}_f n_{f_i} \\ &= \mu\left(\underline{S_f \frac{u_{P_i}-u_{F_i}}{||\mathbf{x}_P-\mathbf{x}_F||_2} - \left(\frac{\partial u_i}{\partial x_j}\right)^{(n-1)}_f (\xi_{f_i}-n_{f_i}) S_f}\right) + \mu\left(\frac{\partial u_j}{\partial x_i}\right)^{(n-1)}_f n_{f_i}, \end{aligned}$$

where the unit vector pointing in direction of the straight line connecting control volume P and control volume F is denoted as

$$\xi_f = \frac{\mathbf{x}_P-\mathbf{x}_F}{||\mathbf{x}_p-\mathbf{x}_F||_2}.$$

The interpolation of the cell center gradients to the boundary faces is performed as in (13). Now the contribution of the diffusive part to the matrix coefficients and the right hand side can be calculated as

$$a_F^{u_i,d} = -\frac{\mu S_f}{||\mathbf{x}_P-\mathbf{x}_F||_2}, \qquad a_P^{u_i,d} = \sum_{F\in NB(P)} \frac{\mu S_f}{||\mathbf{x}_P-\mathbf{x}_F||} \tag{40a}$$

$$\begin{aligned} b^d_{F,u_i} &= \sum_{F\in NB(P)} \left(\frac{\partial u_i}{\partial x_j}\right)^{(n-1)}_f (\xi_{f_i}-n_{f_i}) S_f - \mu\left(\frac{\partial u_j}{\partial x_i}\right)^{(n-1)}_f n_{f_i} S_f \\ &= \sum_{F\in NB(P)} \left(\frac{\partial u_i}{\partial x_j}\right)^{(n-1)}_f \xi_{f_i} S_f - \mu\left(\left(\frac{\partial u_i}{\partial x_j}\right)^{(n-1)}_f - \left(\frac{\partial u_j}{\partial x_i}\right)^{(n-1)}_f\right) n_{f_i} S_f. \end{aligned} \tag{40b}$$

4.5.3 Discretization of the Source Terms

Since in the segregated solution approach in every equation all other variables but the dominant one are treated as constants and furthermore the source terms in equation (38) do not depend on the dominant variable the discretization is straightforward. The source terms of the momentum balance are discretized using the midpoint integration rule for volume integrals, which leads to the source term

$$-\iiint_V \frac{\partial p}{\partial x_i}\mathrm{d}V - \iiint_V \rho\beta\,(T-T_0)\,\mathrm{d}V \approx -\left(\frac{\partial p}{\partial x_i}\right)^{(n-1)}_P V_P - \rho\beta\left(T_P^{(n-1)}-T_0\right)V_P = b^{sc}_{P,u_i}. \tag{41}$$

4.6 Discretization of the Temperature Equation

The discretization of the temperature equation is performed by the same means as for the momentum balance. The only difference is a simpler diffusion term. The integral form of the temperature equation, after applying the Gauss' theorem of integration, is

$$\underbrace{\iint_S \rho u_j T n_j \mathrm{d}S}_{\text{advective term}} - \underbrace{\iint_S \kappa \frac{\partial T}{\partial x_j} n_j \mathrm{d}V}_{\text{diffusive term}} = \underbrace{\iiint_V q_T \mathrm{d}V}_{\text{source term}}.$$

Proceeding as in the previous subsections one can now discretize the advective, the diffusive term and the source term. Since this process does not provide further insight, just the final results will be presented. The discretization yields the matrix coefficients as

$$a_F^T = \min(\dot{m}_f, 0) + \frac{\kappa S_f}{||\mathbf{x}_P - \mathbf{x}_F||_2} \tag{42a}$$

$$a_P^T = \sum_{F \in NB(P)} \max(0, \dot{m}_f) - \frac{\kappa S_f}{||\mathbf{x}_P - \mathbf{x}_F||_2} \tag{42b}$$

$$\begin{aligned} b_{P,T} = & \sum_{F \in NB(P)} \eta\left(T_F^{(n-1)}\left(\min(\dot{m}_f, 0) - \gamma_f\right)\right) + \eta\left(T_P^{(n-1)}\left(\max(0, \dot{m}_f) - \left(1 - \gamma_f\right)\right)\right) \\ & + \sum_{F \in NB(P)} \left(\frac{\partial T}{\partial x_j}\right)_f^{(n-1)} \left(\xi_{f_j} - n_{f_j}\right) S_f \\ & + q_{T_P} V_P. \end{aligned} \tag{42c}$$

Again it is possible though not always necessary, as in the case of the velocities, to under-relax the solution of the resulting linear system with a factor α_T. This can be accomplished as shown in subsection 4.9.

4.7 Boundary Conditions

As the antecedent subsections showed, it is possible to deduce a linear algebraic equation for each control volume by the finite volume method. The approach presented in the preceding subsections however did not cover the treatment of control volumes at domain boundaries yet. This subsection introduces the boundary conditions which are relevant for the present work and furthermore deals with transitional conditions at block boundaries.

4.7.1 Dirichlet Boundary Conditions

The first boundary condition is the Dirichlet boundary condition. This type of boundary condition is used to model inlet conditions for flow problems. For the temperature equation it may also be used at walls as will be shown in subsection 4.7.2. It is characterized by specifying the value of the variable for which the equation is solved explicitly. As a result boundary fluxes can be calculated directly. Especially the mass flux $\dot{m}_f$ is known and hence does not have to be calculated using the pressure weighted interpolation method. Since no special modifications have to be made, as the resulting coefficient for a neighbouring control volume laying past the boundary is considered on the right hand side of the linear system, the implementation approach will be presented only for the temperature equation. Since there is no boundary condition that fixes the gradient at Dirichlet boundaries it is assumed that the partial derivatives of the respective variable are constant and can hence be extrapolated

$$\left(\frac{\partial T}{\partial x_j}\right)_f \approx \left(\frac{\partial T}{\partial x_j}\right)_P.$$

The modification to the central coefficient of the linear equation can be recursively formulated as

$$a_P^T = a_P^T + \left(\max(0, \dot{m}_f) - \frac{\kappa S_f}{||\mathbf{x}_P - \mathbf{x}_f||_2}\right),$$

whereas the contribution to the right hand side reads

$$b_{P,T} = b_{P,T} - \left(\min(\dot{m}_f, 0) - \frac{\kappa S_f}{||\mathbf{x}_P - \mathbf{x}_f||_2} \right) T_f + \left(\frac{\partial T}{\partial x_j} \right)_P^{(n-1)} \left(\xi_{f_j} - n_{f_j} \right) S_f$$

The reader should note, that even though the gradient discretization at domain boundaries is realized by a one sided forward differencing scheme instead of a central differencing scheme. This does not drastically affect accuracy because the distance used in the differential quotient is half the distance used on a central difference inside the domain [**?**].

4.7.2 Treatment of Wall Boundaries

A common boundary to the solution domain is given by solid walls. For all kind of flows this boundary condition first of all has a kinematic character, since the concept of impermeable walls dictates a zero normal velocity at the wall. In viscous flows wall boundaries can be interpreted furthermore as a no-slip condition, i.e. a Dirichlet boundary condition for the velocities. Convective fluxes through solid walls are thus zero by definition however the diffusive fluxes require special treatment not only for the velocities but also for the temperature. To approximate the fluid behavior on a wall boundary correctly, special modifications have to be taken into account to model the normal a shear tension. Furthermore diffusive fluxes for the temperature can be given by Neumann or Dirichlet boundary conditions.

The derivation of the discretized diffusive flux through wall boundaries starts from the integral momentum balance (2) for the vector $\mathbf{u}$. Here only the term for surface forces is needed

$$\mathbf{F}_w = \iint_{S_w} \mathbf{t} \mathrm{d}S = \iint_{S_w} \mathbf{T}(\mathbf{n}_w) \mathrm{d}S \tag{43}$$

For the purpose of treating wall boundary conditions it is appropriate to use a local coordinate system n, t, s where n denotes the wall normal coordinate, t denotes the coordinate tangential to the wall shear force and, s is the binormal coordinate. (FIGURE). With respect to this coordinate system the wall normal vector is represented by $\mathbf{n}_w = (1,0,0)^T$ and the image of the wall normal vector $\mathbf{T}(\mathbf{n_w})$ is represented by

$$\mathbf{T}(\mathbf{n}_w) = \begin{bmatrix} \tau_{nn} & \tau_{nt} & \tau_{ns} \\ \tau_{nt} & \tau_{tt} & \tau_{ts} \\ \tau_{ns} & \tau_{ts} & \tau_{ss} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \tau_{nn} \\ \tau_{nt} \\ \tau_{ns} \end{bmatrix}$$

The velocity of the wall is assumed to be constant, so the directional derivative of the tangential velocity vanishes on the wall

$$\tau_{tt} = \frac{\partial u_t}{\partial x_t} = 0,$$

which in conjunction with the continuity equation in differential form leads to

$$\frac{\partial u_n}{\partial x_n} + \frac{\partial u_t}{\partial x_t} = \frac{\partial u_n}{\partial x_n} = 0,$$

what is equivalent to $\tau_{nn} = 0$ at the wall. A physical interpretation would be that the transfer of momentum at the wall occurs by shear forces exclusively. Furthermore the coordinate direction t is chosen to be parallel to the shear force which is no restriction because of the possibility to rotate the coordinate system within the plane. This leads to $\tau_{ns} = 0$. The absolute value of the surface force hence only depends on the normal derivative of the velocity tangential to the wall. After transforming the coordinates back to the system (x_1, x_2, x_3) the surface force can be calculated and the integral can be discretized using the midpoint integration rule by

$$\mathbf{F}_w = \iint_{S_w} \mathbf{t_w}\, \tau_{nt} \mathrm{d}S = \iint_{S_w} \mathbf{t_w}\, \mu \frac{\partial u_t}{\partial n} \mathrm{d}S = \mathbf{t_w}\, \mu \left(\frac{\partial u_t}{\partial n} \right)_w S_w, \tag{44}$$

where $\mathbf{t_w}$ denotes the transformed tangential vector $(0,1,0)^T$ with respect to the coordinate system (x_1, x_2, x_3). In the discretization process this tangential vector will be calculated from the velocity vector as

$$\mathbf{t_w} = \frac{\mathbf{u}_t}{||\mathbf{u}_t||_2} \quad \text{, where} \quad \mathbf{u}_t = \mathbf{u} - (\mathbf{u} \cdot \mathbf{n}_w)\, \mathbf{n}_w.$$

According to [**?**] this force should not be handled explicitly for convergence reasons. On the other side, if the surface force is expressed by the velocities u_i the discretization process would lead to different central matrix coefficients, which would affect memory efficiency.

The discretization approach used in the present thesis uses a simpler implicit discretization coupled with a deferred correction that combines the explicit discretization of the surface force (44) with a central difference. At first the contained directional derivative is discretized implicitly by

$$\left(\frac{\partial u_t}{\partial n}\right) t_{wi} \approx \left(\frac{\partial u_i}{\partial \xi}\right) \approx \frac{u_{i,P} - u_{i,w}}{||\mathbf{x}_P - \mathbf{x}_w||_2}$$

and explicitly by

$$\left(\frac{\partial u_t}{\partial n}\right) t_{wi} \approx \frac{\left(u_{i,P} - u_{i,w}\right) - \left(u_{j,P} - u_{j,w}\right) n_j n_i}{(\mathbf{x}_P - \mathbf{x}_w) \cdot \mathbf{n}_w}.$$

Therefore the contributions to the central coefficient and the right hand side of the linear equation that results from the presented discretization process are

$$a_P^{u_i} = a_P^{u_i} + \mu \frac{S_f}{||\mathbf{x}_P - \mathbf{x}_w||_2} \quad \text{and}$$

$$b_{P,u_i} = b_P^{u_i} + \mu \frac{S_f}{||\mathbf{x}_P - \mathbf{x}_w||_2} u_{i,P}^{(n-1)} + \frac{\left(u_{i,P}^{(n-1)} - u_{i,w}\right) - \left(u_{j,P}^{(n-1)} - u_{j,w}\right) n_j n_i}{(\mathbf{x}_P - \mathbf{x}_w) \cdot \mathbf{n}_w}.$$

The reader should note that since the deferred correction uses a Dirichlet boundary condition, no correction of the value of the wall velocity $u_{i,w}$ has to be accounted for.

If the solution of the flow field is coupled to the solution of a temperature equation, different options for the boundary condition at walls may be chosen. If the wall temperature is known, a Dirichlet boundary condition for the temperature is the choice. A wall of this type is called to be *isothermal*. If on the other side only the heat flux is known a Neumann boundary condition is used. In the special case of zero heat flux the wall is called to be *adiabatic*. For adiabatic walls, which are besides isothermal walls used for the present thesis the implementation is straight forward since no coefficients have to be calculated.

4.7.3 Treatment of Block Boundaries

If block structured grids are used to decompose the problem domain, the characterizing property of structured grids, which is the constant amount of grid cells in each direction, gives each inner cell exactly six neighbours. If more then one grid block is used this property is violated if the number of grid cells of each block is chosen to be arbitrary. The arbitrariness of the grid resolution is a main benefactor for the adaptivity of block structured grids.

To maintain the conservation property of finite volume methods block boundaries should not be interpreted as boundaries in the classical sense. Instead conditions have to be formulated to guarantee that flows through block boundaries are conserved. The method to treat block boundaries used in the present thesis was presented by [**?**]. Another method was presented in [**?**]. However the first method was chosen since it allows a fully implicit consideration of the block boundary fluxes. This method calculates fluxes through separate face segments S_l that come up on *non-matching* grid blocks. Each of this face segments gets assigned a control volume L and a control volume R which exclusively share this face segment. Figure **??** shows a block boundary when non matching blocks are used. An algorithm to provide this needed information will be presented in REFERENCE.

Once the geometric data including the information regarding the interpolation has been provided, the fluxes through this boundaries can be calculated as presented in the antecedent subsections. The connectivity of neighbouring control volumes of different grid blocks is represented by matrix coefficients a_L^ϕ and a_R^ϕ, where $\phi \in \{u_1, u_2, u_3, p, T\}$.

4.8 Treatment of the Singularity of the Pressure Correction Equation with Neumann Boundaries

It has to be noted that the derived pressure correction equation is a Poisson equation. As can be proven **??**, the linear $N \times N$ systems surging from the presented discretization on a grid with N control volumes have a nullspace of dimension one, i.e.

$$\text{null}(A_{p'}) = \text{span}(\mathbb{1}),$$

where $\mathbb{1} = (1)_{i=1,\dots,N} \in \mathbb{R}^N$ is the vector spanning the nullspace. This singularity accounts for the property of incompressible flows, that pressure can only be determined up to a constant. To fix this constraint various possibilities exist [**?**]. A common method is to set the pressure correction to zero in one reference control volume and hence fix the pressure at

one reference point in the problem domain. This can be done before solving the system by applying this Dirichlet-type condition, or it can be done afterwards when pressure is calculated from the pressure correction. This approach is not suitable for grid convergence studies since without proper interpolation it is not guaranteed that the reference pressure correction is taken at the correct location.

Since some comparisons performed in the present work rely on grid convergence studies another approach for reducing the lose constraint of the pressure correction system has been used: The reference pressure correction is taken to be the mean value of the pressure correction over the domain

$$p'_{\text{ref}} = \frac{\iiint_V p' \mathrm{d}V}{\iiint_V \mathrm{d}V} \approx \frac{\sum_{P=1}^{N} p'_P V_P}{\sum_{P=1}^{N} V_P}$$

such that the net pressure correction amounts to zero. This modifies equation (33) to read

$$p_P^{(n)} = p_P^{(n-1)} + \alpha_{\mathbf{p}} \left(p'_P - p'_{\text{ref}} \right). \tag{45}$$

4.9 Structure of the Assembled Linear Systems

The objective of a finite volume method is to create a set of linear algebraic equations by discretizing partial differential equations. In the case of the discretized momentum balance, taking all contributions together leads to the following linear algebraic equation for each control volume P

$$a_P^{u_i} u_{P_i} + \sum_{F \in NB(P)} a_F u_{F_i} = b_{P,u_i},$$

where the coefficients are composed as

$$a_P^{u_i} = a_P^{u_i,c} - a_P^{u_i,d} \tag{46a}$$

$$a_F^{u_i} = a_F^{u_i,c} - a_F^{u_i,d} \tag{46b}$$

$$b_{P,u_i} = b_{P,u_i,c} - b_{P,u_i,d} + b_{P,u_i}^{sc}. \tag{46c}$$

Similar expressions for the pressure correction equation and the temperature equation exist. In the case of control volumes located at boundaries some of the coefficients will be calculated in a different way. This aspect is addressed in section 4.7.

For the decoupled iterative solution process of the Navier-Stokes equations it is necessary to reduce the change of each dependent variable in each iteration. Normally this is done by an *under-relaxation* technique, a convex combination of the solution of the linear system for the present iteration (n) and from the previous iteration $(n-1)$ with the under-relaxation parameter $\alpha_{u_i} \in (0, 1]$, where $\alpha_{u_i} = 1$ refers to no under-relaxation. Generally speaking this parameter can be chosen individually for each equation. Since there are no rules for choosing this parameters in a general setting the under-relaxation parameter for the velocities is chosen to be equal for all three velocities, $\alpha_{u_i} = \alpha_{\mathbf{u}}$ [**?**]. This has the further advantage that, in case the boundary conditions are implemented with the same intention, the linear system for each of the velocities remains unchanged except for the right hand side. This helps to increase memory efficiency.

Let the solution for the linear system without under-relaxation be denoted as

$$\tilde{u}_{P_i}^{(n)} := \frac{b_{P,u_i} - \sum_{F \in NB(P)} a_F u_{F_i}}{a_P^{u_i}},$$

Which is only a formal expression for the case the underlying linear system is solved exactly. A convex combination as described yields

$$\begin{aligned} u_{P_i}^{(n)} &:= \alpha_{\mathbf{u}} \tilde{u}_{P_i}^{(n)} + (1 - \alpha_{\mathbf{u}}) u_{P_i}^{(n-1)} \\ &= \alpha_{\mathbf{u}} \frac{b_{P,u_i} - \sum_{F \in NB(P)} a_F u_{F_i}}{a_P^{u_i}} + (1 - \alpha_{\mathbf{u}}) u_{P_i}^{(n-1)}, \end{aligned}$$

an expression that can be modified to derive a linear system whose solution is the under-relaxed velocity

$$\frac{a_P^{u_i}}{\alpha_{\mathbf{u}}} u_{i,P} + \sum_{F \in NB(P)} a_F^{u_i} u_{i,F} = b_{P,u_i} + \frac{(1-\alpha_{\mathbf{u}})\, a_P^{u_i}}{\alpha_{\mathbf{u}}} u_{i,P}^{(n-1)}.$$

After all matrix coefficients have been successfully calculated the linear system system can be represented by a system matrix A and a right hand side vector b. Figure 5.4 shows the non-zero structure of a linear system for a grid consisting in a $2 \times 2 \times 2$ cell and a $3 \times 3 \times 3$ cell block.

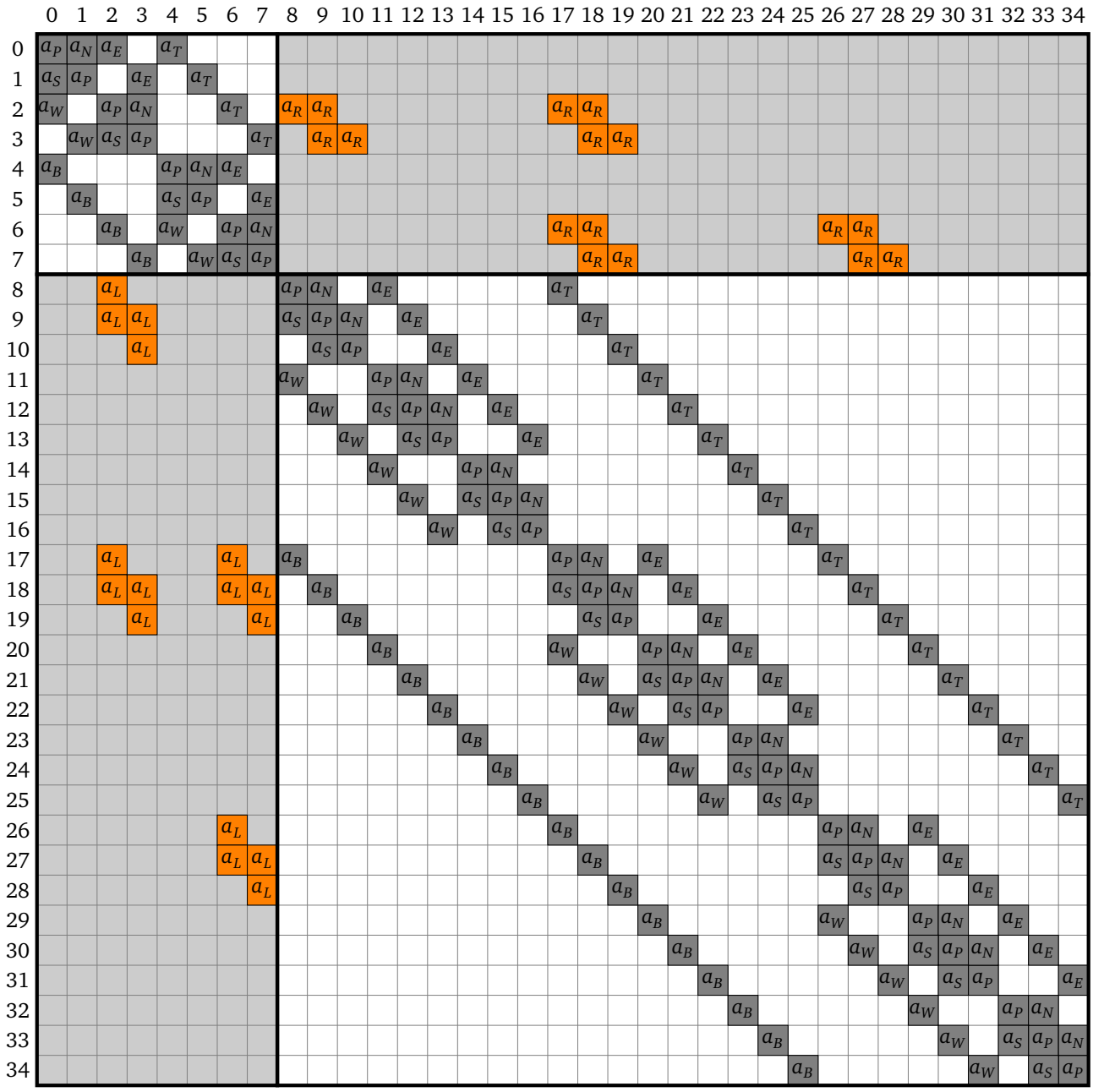

Figure 5: Non-zero structure of the linear systems used in the SIMPLE algorithm for a block structured grid consisting of one $2 \times 2 \times 2$ cell and one $3 \times 3 \times 3$ cell block

5 Implicit Finite Volume Method for Incompressible Flows – Fully Coupled Approach

Since the antecedent section 4 already discussed the discretization details on the involved equations, this section aims at a comparison at the algorithmic level of the SIMPLE algorithm, presented in section 4.3, as a method to resolve the pressure-velocity coupling, and an implementation of a fully coupled solution algorithm. It should be noted that the discretization of the equations to be solved is not changed in any way to maintain comparability, so the presented differences are exclusively due to difference in the solution algorithm. Successful implementations of a fully coupled solution algorithm for incompressible Navier-Stokes equations have been presented in [**?**, **?**, **?**, **?**]. In addition the presented work will extend the solution approach presented in [**?**] to three-dimensional domains. Furthermore this section will present various approaches to incorporate different degrees of velocity-to-temperature and temperature-to-velocity/pressure coupling. Finally the structure of the resulting linear system to be solved is discussed.

5.1 The Fully Coupled Algorithm – Pressure-Velocity Coupling Revised

This subsection motivates the use of a fully coupled algorithm to resolve pressure-velocity coupling and mentions the differences to the approach presented in subsection 4.3. The mentioned subsection presented a common solution approach to solve incompressible Navier-Stokes equations: After the linearization of the equations, momentum balances were solved using the pressure from the previous iteration as a guess. Since in general the velocity field obtained by solving a momentum balance with a guessed pressure does not obey continuity, the velocity field and the pressure field had to be corrected. This in turn would lead to an inferior solution regarding the residual of the momentum balances. To avoid this iterative guess-and-correct solution process another class of approaches to the pressure-velocity coupling problematic, represented by algorithms that are *fully coupled*, will be introduced now. .

The central aspect of fully coupled solution methods for Navier-Stokes equations is, that instead of solving for the velocities and pressure corrections sequentially, the velocity field and the pressure are solved for simultaneously [**?**], so every velocity field that is calculated obeys conservation of momentum and mass, without needing to be corrected. As a result the under-relaxation of pressure and velocities is no longer required, which accelerates convergence significantly in terms of needed outer iterations. The only reason for still needing an iterative solution process is the non-linearity of the Navier-Stokes equations which is accounted for by the use of the Picard iteration process, which has been presented in subsection 3.4. The iterations fortunately make room for deferred corrections as introduced in subsection 4.5.

On the downside implementations of fully coupled solution methods require significantly more system memory than segregated methods. This is due to the higher amount of information that has to be available at the same time and the resulting linear systems which also require more memory, due to the increased amount of unknowns to solve for. Furthermore the bad condition of the linear algebraic system [**?**] tends to slow down the convergence of the equation solver algorithm.

As in the case of segregated methods it is not advisable to solve the mass balance equation directly [**?**]. Instead of deriving an equation for the pressure correction p', as shown in 4.3, an equation for the pressure is derived by using the pressure-weighted interpolation method, which has been introduced in 4.2 and is used to approximate the velocities in the discretized mass balance (16). Concretely equation (21) is adapted for the use in a fully coupled algorithm

$$u_{i,f}^{(n)} = \left[\left(1-\gamma_f\right)u_{i,P}^{(n)} + \gamma_f u_{i,Q}^{(n)}\right] - \left(\left(1-\gamma_f\right)\frac{V_P}{a_{P,u_i}} + \gamma_f \frac{V_Q}{a_{Q,u_i}}\right)\left[\underline{\left(\frac{\partial p}{\partial x_i}\right)_f^{(n)}} - \frac{1}{2}\left(\left(\frac{\partial p}{\partial x_i}\right)_P^{(n-1)} + \left(\frac{\partial p}{\partial x_i}\right)_Q^{(n-1)}\right)\right].$$

The changes comprise the removal of last term of (21) accounting for the under-relaxation, since no under-relaxation is needed in fully coupled algorithms which is equivalent to an under-relaxation factor $\alpha_{\mathbf{u}} = 1$. Furthermore the underlined partial derivatives are now going to be treated implicitly. This leads to the semi-implicit pressure equation

$$\begin{aligned}\sum_{F\in NB(P)} \rho\left[\left(1-\gamma_f\right)u_{i,P}^{(n)} + \gamma_f u_{i,F}^{(n)}\right] S_f - \rho\left(\left(1-\gamma_f\right)\frac{V_P}{a_{P,u_i}} + \gamma_f \frac{V_F}{a_{F,u_i}}\right)\left[\left(\frac{\partial p}{\partial x_i}\right)_f^{(n)}\right] S_f \\ = -\sum_{F\in NB(P)} \rho\left(\left(1-\gamma_f\right)\frac{V_P}{a_{P,u_i}} + \gamma_f \frac{V_F}{a_{F,u_i}}\right)\left[\frac{1}{2}\left(\left(\frac{\partial p}{\partial x_i}\right)_P^{(n-1)} + \left(\frac{\partial p}{\partial x_i}\right)_F^{(n-1)}\right)\right] S_f. \end{aligned} \tag{47}$$

Even though this equation is solved for the pressure p while equation (31) is solved for the pressure correction p' the matrix coefficients from the pressure correction discretization of equation (37) and the calculation of the right hand

side is given in equation (32). For the implicit coupling to the velocities u_i additional matrix coefficients have to be considered. These can be calculated as

$$a_F^{p,u_i} = \rho\,\gamma_f S_f \quad \text{and} \quad a_P^{p,u_i} = \sum_{F \in NB(P)} \rho\,(1-\gamma_f) S_f .$$

On the other side the discretized momentum balance gives matrix coefficients to account for the implicit velocity-pressure coupling. They are calculated as

$$a_F^{u_i,p} = \gamma_f S_f \quad \text{and} \quad a_P^{u_i,p} = \sum_{F \in NB(P)} (1-\gamma_f) S_f .$$

If the Boussinesq approximation is used implicitly, an additional coefficient $a_P^{u_i,T}$ to account for the velocity temperature coupling has to be considered. This will be handled in subsection 5.2

Algorithm 2 Fully Coupled Solution Algorithm

```
INITIALIZE variables
while (convergence criterion not accomplished) do
    if (Temperature coupling) then
        SOLVE the linear system for velocities, pressure and temperature
    else
        SOLVE the linear system for velocities and pressure
    end if
    CALCULATE mass fluxes using REFERENCE
    if (Decoupled scalar equation) then
        SOLVE scalar equation as described in (4.6)
    end if
end while
```

5.2 Coupling to the Temperature Equation

Since the present work also aims at analyzing the efficiency of different methods to couple the temperature equation to the velocities and vice-versa this subsection discusses different approaches to handle the coupling of the temperature equation to the Navier-Stokes equations if the Boussinesq-Approximation, as introduced in subsection 2.5.2, is used. The effectiveness of realizing a strong coupling of the temperature equation to the Navier-Stokes equations depends on the flow problem. The physical coupling tends to increase in flow scenarios of natural convection, where the temperature difference and hence the buoyancy term in the Navier-Stokes equations dominates the fluid movement. Generally speaking: The higher the physical coupling of temperature and flow, the greater are the benefits of using an implementation that uses a strong coupling to the temperature equation. The following subsection present different approaches that can be used for different intensities of coupling.

5.2.1 Decoupled Approach – Explicit Velocity-to-Temperature Coupling

The decoupled approach is similar to the treatment of the velocity-temperature coupling described in subsection 4.5. If this approach is chosen no special measures have to be taken. The momentum balances receive a contribution $b_P^{u_i,T}$ with

$$b_P^{u_i,T} := -\rho\beta\left(T_P^{(n-1)} - T_0\right) V_P,$$

that handles the coupling explicitly by using the temperature result of the previous outer iteration. In some cases it might be necessary to under-relax the temperature each iteration

5.2.2 Implicit Velocity-to-Temperature Coupling

In the same way it is possible to realize the coupling described in the previous subsection by implicitly considering the temperature in the momentum balances, which leads to an additional matrix coefficient $a_P^{u_i,T}$ and an additional contribution to the right hand side accounting for the coupling to the temperature equation

$$a_P^{u_i,T} = \rho\beta V_P \quad \text{and} \quad b_P^{u_i,T} = \rho\beta T_0 V_P .$$

5.2.3 Temperature-to-Velocity/Pressure Coupling – Newton-Raphson Linearization

The previous section discussed the velocity temperature coupling and so far only the momentum balances were affected by the realization of the coupling methods. Independently it is possible to couple the temperature equation not only to the momentum balances but also to the pressure equation. In section 4.6 the non-linear partial differential equation was linearized using a Picard iteration method. Specifically the convective term $\rho u_j T$ was linearized by taking the mass flux from the antecedent solve of the momentum and pressure correction equations. If the decoupled approach is used, this treatment remains valid however if an implicit temperature coupling sought a coupling of the temperature equation to the momentum balances which considers the mass fluxes is desirable. Methods that exhibit the described property can be found in [**?**, **?**, **?**, **?**]. A common denomination of the therein used linearization method is called *Newton-Raphson linearization* which can be interpreted as a bilinear approximation of the convective term by the linear terms of the respective Taylor polynomial.

The Newton-Raphson linearization technique is applied to the convective term of the temperature equation as follows. A first order Taylor approximation for the mass specific convective flux through the boundary face S_f can be formulated as

$$\begin{aligned}
\left(u_{j,f}\,T_f\right)^{(n)} &\approx \left(u_{j,f}\,T_f\right)^{(n-1)} + \frac{\partial}{\partial u_{j,f}}\left(u_{j,f}\,T_f\right)^{(n-1)}\left(u_{j,f}^{(n)} - u_{j,f}^{(n-1)}\right) + \frac{\partial}{\partial T_f}\left(u_{j,f}\,T_f\right)^{(n-1)}\left(T_f^{(n)} - T_f^{(n-1)}\right) \\
&= (u_{j,f}\,T_f)^{(n-1)} + {T_f}^{(n-1)}({u_{j,f}}^{(n)} - {u_{j,f}}^{(n-1)}) + {u_{j,f}}^{(n-1)}({T_f}^{(n)} - {T_f}^{(n-1)}) \\
&= \underline{{u_{j,f}}^{(n-1)}{T_f}^{(n)}} + {u_{j,f}}^{(n)}{T_f}^{(n-1)} - {u_{j,f}}^{(n-1)}{T_f}^{(n-1)}.
\end{aligned}$$

A comparison with REFERENCE shows, that the underlined term is the same as in the usual linearization. The first of the two new terms will be treated implicitly after using the pressure-weighted interpolation method from section **??** to interpolate the value of $u_{j,f}$ and the second term will be treated explicitly. The use of the pressure-weighted interpolation method creates not only temperature-to-velocity but also temperature-to-pressure coupling. Applying the pressure-weighted interpolation the equation for the mass specific convective flux reads

$$\begin{aligned}
\left(u_{j,f}\,T_f\right)^{(n)} \approx\ & \underline{{u_{j,f}}^{(n-1)}{T_f}^{(n)}} \\
& + {T_f}^{(n-1)}\left[\left(1-\gamma_f\right)u_{i,P}^{(n)} + \gamma_f u_{i,Q}^{(n)}\right] \\
& - \left(\left(1-\gamma_f\right)\frac{\alpha_{\mathbf{u}}V_P}{a_{P,u_i}} + \gamma_f\frac{\alpha_{\mathbf{u}}V_Q}{a_{Q,u_i}}\right)\left[\left(\frac{\partial p}{\partial x_i}\right)_f^{(n)} - \left(1-\gamma_f\right)\left(\frac{\partial p}{\partial x_i}\right)_P^{(n)} - \gamma_f\left(\frac{\partial p}{\partial x_i}\right)_Q^{(n)}\right] \\
& + (1-\alpha_{\mathbf{u}})\left[u_{i,f}^{(n-1)} - \left(1-\gamma_f\right)u_{i,P}^{(n-1)} - \gamma_f\, u_{i,Q}^{(n-1)}\right]. \\
& - {u_{j,f}}^{(n-1)}{T_f}^{(n-1)}.
\end{aligned}$$

5.3 Boundary Conditions on Domain and Block Boundaries

5.3.1 Dirichlet Boundary Condition for Velocity

Mention that the Newton-Raphson linearization is not considered on Dirichlet boundaries.

5.3.2 Wall Boundary Condition

5.3.3 Block Boundary Condition

5.4 Assembly of Linear Systems – Final Form of Equations

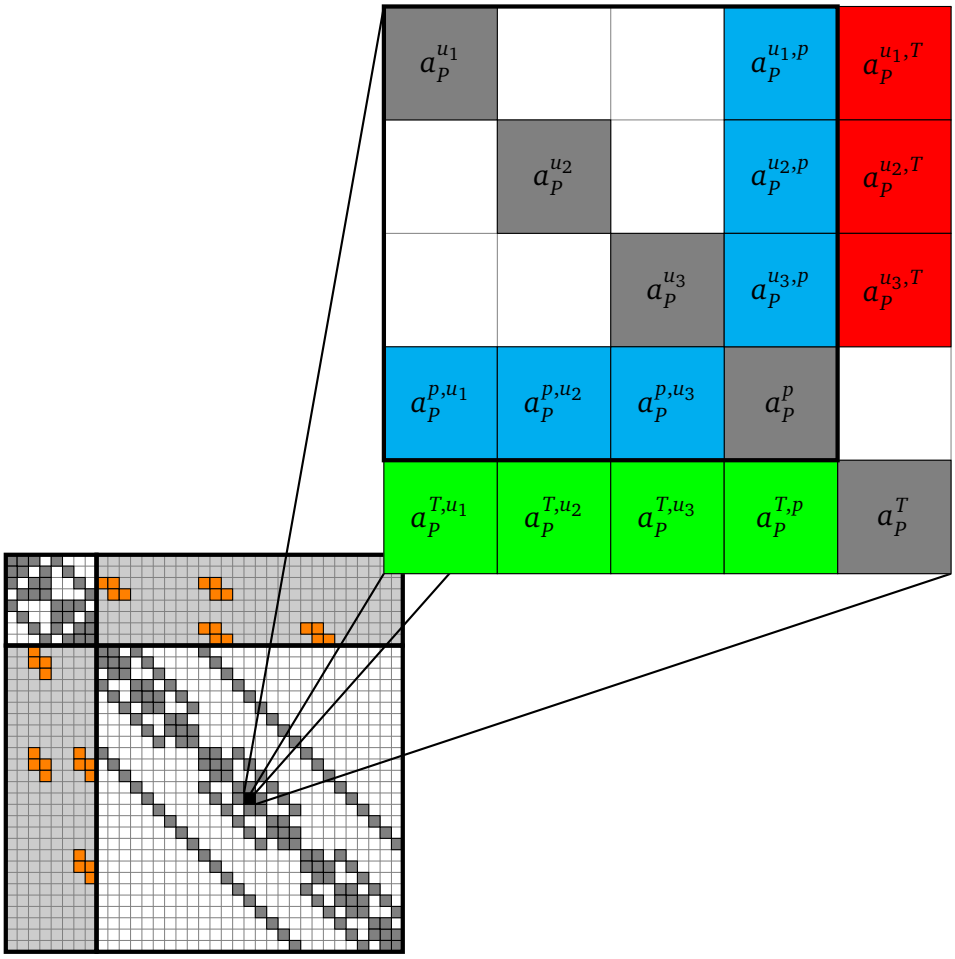

Figure 6: Non-zero structure of the linear systems used in the coupled solution algorithm for a block structured grid consisting of one $2 \times 2 \times 2$ cell and one $3 \times 3 \times 3$ cell block

6 CAFFA Framework

This section presents the CAFFA framework that has been extended for the present thesis. *CAFFA* is an acronym and stands for *Computer Aided Fluid Flow Analysis*. The solver framework is based on Perić's CAFFA code [**?**,**?**] that has been modified to handle three-dimensional problem domains and block structured grids with non-matching blocks. Furthermore, the framework has been parallelized making extensive use of the PETSc library. One characteristic of the solver framework is, that arbitrary block boundaries are handled in a fully implicit way. Details on how those transitional conditions are handled are located in section 4.7.3. The framework consists of two different solver algorithms, a segregated solver and a fully coupled solver, which have been introduced in sections 4 and 5 respectively. In addition, the framework features an own implementation of a grid generator for block structured grids as well as a mapper program to convert grids created with ICEM CFD [**?**]. To prepare the grid data for the later use in the solvers, a preprocessor program has been integrated into the framework. Furthermore the solver programs exhibit different export functionalities to visualize the numerical grid and the results within ParaView [**?**] and the export of binary vectors and matrices to MATLAB ® [**?**].

FIGURE

The following subsections will now briefly introduce the PETSc framework, present the components of the CAFFA framework and finally sketch the program flow.

6.1 PETSc Framework

PETSc is an acronym for *Portable Extensible Toolkit for Scientific Calculations* and is a software suite that comprises data structures and an extensive set of linear solver routines [**?**, **?**]. A great part of the data structures is designed to work in parallel on high performance computers, also the solver routines have been parallelized using the MPI standard for message passing. PETSc provides the tools that can be used to build large-scale application codes for scientific calculations. The great advantage of PETSc is that adds multiple additional layers of abstraction that make its use in applications straightforward and allow the users to focus on the essential implementations of their own code instead of having to deal with parallel data structures and algorithms, which represents an additional source for programming errors. PETSc provides different interfaces through which the user can parallelize his code, what furthermore increases readability and maintainability of the developed code. Last but not least, the implementations used for data structures and solvers in PETSc have are not only verified and updated on a regular basis, they have furthermore proven to be highly efficient in the creation of large-scale computer applications on high performance clusters REFERENCE.

The data structures PETSc offers reflect the commonly used elements in scientific codes. Vectors, matrices and index sets are among this basic data structures. PETSc offers a variety of options for its data structures. Vectors can either be used sequentially or parallelly in which case PETSc distributes the vector components to the processes inside a MPI communicator. Furthermore parallel matrix formats are available that allow the processes to assemble different parts of one global matrix simultaneously. PETSc contains other data structures that can be used to fetch and distribute either vector or matrix elements from remote processes.

On the other side PETSc contains a large collection of parallel solvers for linear systems. The major part comprises Krylov subspace methods which have been introduced in section **??**. Among the variety of solver options that are available for further optimization of the performance, the Krylov subspace methods can be combined with elements of another thorough collection of preconditioners. For further details on the available data structures and subroutines the reader is referred to [**?**, **?**].

In addition to the mentioned tools PETSc also comes with an internal profiler tool that collects a variety of information on the used PETSc objects and bundles them into a human readable log file. The log file and other dynamic output which can be triggered by command line arguments facilitate the optimization process of developed applications and allow the accurate determination of performance measures.

FIGURE

6.2 Grid Generation and Preprocessing

A grid generator has been developed to generate hexahedral block structured grids. To provide flexibility for testing locally refined and skewed grids the grid generator deploys a random number generator which moves grid points within the domain and varies the number of grid cells of each block optionally. Figure REFERENCE shows an example grid, that was generated with the developed grid generator.

FIGURE

Key feature of the developed framework is the handling of block structured locally refined grids with non-matching block interfaces. To neighbouring relations are represented by a special type of boundary conditions; Random number generator to move grid points within an epsilon neighbourhood while maintaining the grid intact.

6.3 Preprocessing

Matching algorithm – the idea behind clipper and the used projection technique; alt.: Opencascade. Efficient calculation of values for discretization. Important for dynamic mesh refinement, arbitrary polygon matching, parallelizable due to easier interface

6.4 Implementation of CAFFA

This section provides an overview of the implementation aspects of the developed CAFFA framework. Since applications for modern computers have to be able to efficiently use the provided resources the concept for the parallelization of the application is presented. A separate section presents how convergence is monitored and controlled. Later on, this section presents ho to use the data structures that store the variables and how to realize the domain composition as central part of the parallelization process.

6.4.1 The Message-Passing Model

The PETSc framework makes thoroughly use of the message-passing computational model [**?**]. This model assumes that a set of processes with local memory is able to communicate with other processes by passing, i.e. sending and receiving messages. *MPI* is an acronym for *Message Passing Interface* and is a software library that collects features of message-passing systems. More importantly it provides a widely used standard that assures portability of application that make use of this library as does PETSc.

The message-passing model is not only reflected in the application as a programming paradigm, but also in the design of parallel computers and their communication network. Under this perspective the message-passing model is complementary to the distributed memory design, which encapsulates the idea of local memory and the exchange of data through a network. High performance computers are seldom designed exclusively as a distributed memory system, rather they incorporate features of another model, namely the shared memory model. This memory model is characterized by a common address space, which each of the processes can access uniformly, i.e. with a location independent latency, or non-uniformly. Latter systems are also called *NUMA* systems (*Non-Uniform Memory Access*). A widely used system design, as for the system the performance tests of section REFERENCE are performed on, comprises features of both models by using NUMA computing nodes that share local memory and are able to exchange messages with other NUMA nodes via an interconnect.

One important advantage of using software that uses the message passing model and hence applications that were parallelized within this programming paradigm, is their compatibility with computers that use combinations of distributed and shared memory.

6.4.2 Convergence Control

One part of the PETSc philosophy is to provide great flexibility in choosing solvers and parameters from within the command line. However, the developed framework implements a non-linear iteration process without using the SNES objects provided by PETSc. This creates the need for a hard-coded implementation of convergence control. Convergence control for the used Picard iteration method comprises two parts. One part controls the overall convergence, and determines when the calculations have finished. The other part controls convergence on an outer iteration basis. The convergence criteria are met if the actual residual falls below the upper bound for the residual r_{final}. The bound for final convergence is hereby determined to

$$r_{final} = r_{initial} * 10^{-8},$$

which states that a decrease of the initial residual of 10^{-8} indicates convergence. This criterion will be fixed for all further analyses since it controls the overall accuracy of the solver results and hence maintains comparability of solver performance.

The other convergence criterion can be handled with more flexibility, since it controls the relative decrease of the residual in each outer iteration. This parameter should not affect the overall accuracy but instead convergence speed, within the individual bounds on the solver algorithm. This means that, depending on the coupling algorithm, low relative solver tolerances will not always benefit convergence speed. On the other hand low relative solver tolerances will affect the number of inner iterations and hence the execution time of the solver program. The convergence criterion for each outer iteration is implemented as

$$r^{(k)}_{final} = r^{(k)}_{initial} * \text{rtol},$$

where rtol indicates the relative decrease of the residual in each outer iteration.

6.4.3 Indexing of Variables and Treatment of Boundary Values

All variables needed by the CAFFA solver are represented by one-dimensional arrays. To establish a mapping between a location (i, j, k) of the numerical grid and the respective variable values location (ijk) inside the array the following formula is used

$$(ijk) = N_i N_j (k-1) + N_j (i-1) + j,$$

where N_i and N_j are the Number of grid cells in the respective coordinate direction plus two additional boundary cells. The inclusion of the boundary cells circumvents the need to provide an additional data structure for boundary values. Furthermore the same mapping rule can be used for the grid vertex coordinates. This variable indexing will be used to assemble the matrices surging from the discretization process. This will lead to rows for the boundary values, that that are decoupled from the rest of the linear system. PETSc provides two different approaches to handle boundary values. In the first approach another abstraction layer is introduces via an object, that redistributes the data and removes the rows containing the boundary values. This approach has been tested with the conclusion that the redistribution not only causes significant overhead but also hides important data for debugging solver convergence. Section REFERENCE compares solves with a redistribution object with those that retain the boundary values. In the present work a second approach is used, that retains boundary values and adapts the right hand side accordingly. This approach has proven to be more efficient with the drawback of memory overhead and the need to reset the boundary values after a linear system has been solved with a high relative tolerance.

This simple indexing model is not capable to handle neither block structured grids nor grids that have been distributed across various processors within the MPI programming model. This characteristic is implemented through the use of additional mappings that return a constant offset $(ijk)_b$) for each block and $(ijk)_p$ for each processor

$$(ijk) = (ijk)_p + (ijk)_b + N_i N_j (k-1) + N_j (i-1) + j.$$

The introduction of the processor offset is needed to provide a mapping between locally and globally indexed values. Within a process only the local indexing

$$(ijk)_{loc} = (ijk)_b + N_i N_j (k-1) + N_j (i-1) + j.$$

will be used, since these indexes can be mapped directly to memory addresses. For inter process communication however, global indices have to be provided to the respective PETSc subroutines by adding the processor offset

$$(ijk)_{glo} = (ijk)_{loc} + (ijk)_p. \tag{48}$$

It should be noted that the presented mappings do not include the index mappings from FORTRAN applications, which use 1-based indexing to the PETSc subroutines which use 0-based indices.

As presented in section REFERENCE, the use of a coupling algorithm leads to a global linear system that contains the linear algebraic equations for different variables. The implementation used for the present thesis interlaces this values, which increases data locality and hence improves memory efficiency. The presented mapping is easily extended to handle arrays of values that contain n_{eq} variables and use an interlaced storing approach by multiplying (48) with the number of interlaced values

$$(ijk)_{interglo} = n_{eq} * (ijk)_{glo}$$

6.4.4 Domain Decomposition, Exchange of Ghost Values and Parallel Matrix Assembly

The developed CAFFA framework is able to solve a given problem across different processors. For this, before each solve, the relevant data has to be distributed among all involved processes. Within the solver parallelization three types of this data distribution are considered. The first refers to the distribution of stationary data that will not change throughout the solution process. This refers mostly to geometry related data as the coordinates of the grid points. The second type of data has to be distributed in a regular way since this data changes at least every outer iteration. The necessity to interchange values, also known as *ghosting*, between processors surges when calculations of variable gradients or coefficients are made for block boundary control volumes. The last type of data distribution refers to global reduce operations, in which a single processes gathers data from all other processes and after some modification redistributes the data. A common scenario for global reduce operations is the calculation of the mean pressure throughout the domain which has been introduced in section REFERENCE.

The distribution of stationary data takes place throughout the preprocessing program within the developed framework before the CAFFA solver is launched. Each processor is assigned a binary file with the respective data. Since the solver is not able to handle adaptive grid refinement or dynamic load balancing this approach is straightforward.

The present solver framework treats the calculation of matrix coefficients for block boundary control volumes in a special way. To reduce communication overhead and redundant data only one of the two involved processors calculates this coefficients and then sends them to the respective neighbour. The data needed to calculate the matrix coefficients of a neighbouring processor embraces not only the values of the dominant variables from the last outer iteration but also the values of the cell center gradients. For the ghosting of variable values PETSc offers special vector types, that comprise a user friendly interface to use the gather and scatter routines provided by PETSc. During the creation of these vectors a set of global indices referring to the values that are to be ghosted has to be provided. After this, the interchange of values is realized through a single subroutine call, since PETSc handles the concrete communication process necessary to accomplish the value interchange. The local representation of a ghosted vector hence not only comprises the data local to one processor but provides space for ghosted values that are repeatedly updated. The layout for one ghosted vector is shown for a simple two-dimensional solution domain that consists of two grid blocks.

FIGURE

7 Verification of the developed CAFFA Framework

The systematical verification of program code is an essential part of the development process of software for scientific calculations, since it insures that the respective equations are solved correctly [**?**]. Scientific applications that solve partial differential equations can be verified using the method of manufactured solutions [**?**]. In addition, the method of manufactured solutions will be used to proof that both solvers within the developed framework, namely the segregated and the fully coupled solver, use the same discretization of the underlying partial differential equations. Before the performance of the developed CAFFA framework will be discussed in section **??**, this section presents the results of the verification process performed in the present work via the *Method of Manufactured Solutions*. After mentioning the main aspects of this procedure, this section will furthermore present the theoretical discretization error of the finite volume method as it is applied in the present thesis. The subsections that follow will then present a concrete manufactured solution and the results of using it in the verification process.

7.1 The Method of Manufactured Solutions for Navier-Stokes Equations

The method of manufactured solutions comprises a systematical, formal procedure for code verification based on analytical solutions of the partial differential equations to be solved [**?**]. Using this analytical solutions the accuracy of the produced results can be assessed and, after the establishment of an acceptance criterion, used to verify the computer program. A common acceptance criterion is the *Order-of-Accuracy* criterion, since in addition to the formal order of accuracy of a developed solver algorithm it verifies its consistency. It should be noted that by the method of manufactured solutions it is not possible to detect errors in the physical model.

The basic idea of the method of manufactured solutions is the inversion of the solution process of partial differential equations. Instead of trying to find the solution x to a given equation $F(x) = b$ with source term b, a solution x is *manufactured* and the source term is constructed by applying F to x. This has the advantage of choosing a solution that exercises all parts of the solution process and hence provides a thorough testing environment. On the downside the integration of boundary conditions other than Dirichlet boundary conditions REFERENCE might be challenging. Furthermore the program code has to be able to handle arbitrary source terms.

The different guidelines that have to be followed in order to apply the method of manufactured solutions successfully, can be found in [**?**]. According to the reference, manufactured solutions should be composed of infinitely often differentiable analytic functions whose derivatives are bounded by small constants. Furthermore the solution domain should not be chosen to be symmetric, but as arbitrary as possible within the limitations of the code. This may however conflict with the application of the method of manufactured solutions to verify a code to solve the Navier-Stokes equations. Subsection 7.2 presents the manufactured solution including the resulting source terms for the set of partial differential equations (11).

The verification of a solver for incompressible Navier-Stokes equations comes with additional conditions, that have to be considered: First, if the solver is not able to handle source terms in the continuity equation and respectively the pressure correction equation, a velocity field should be chosen that is inherently divergence free. This can be achieved by defining the velocity field $\mathbf{u}$ through the curl of a vector field $\mathbf{\Psi}$ as

$$\mathbf{u} = \nabla \times \mathbf{\Psi}.$$

Using the property that the divergence of the rotation of a vector field vanishes, one gets

$$\nabla \cdot \mathbf{u} = \nabla \cdot (\nabla \times \mathbf{\Psi}) = 0,$$

i.e. a locally divergence free velocity field, which implies that globally the continuity equation also is fulfilled in an integral sense for arbitrary domains of integration. This is not necessarily true in the discrete sense as integrals are to be approximated by, in the case of the present thesis, the midpoint rule of integration, as shown in subsection 3.2. In the general case for arbitrary solution domains global mass conservation cannot be guaranteed even though the integrands are exact quantities at the domain boundaries. Thus in the case of finding manufactured solutions the problem domain should be fixed before manufacturing a solution. The velocity field for the manufactured solution should then either vanish on the domain boundaries, which is the case of the commonly used Taylor-Green vortex [**?**], or should exhibit further symmetry that leads to cancelling non-zero mass fluxes across the domain boundaries. The studies performed within the scope of the present showed, that convergence of the pressure correction equation can no longer be guaranteed if the velocity field does not obey continuity in the discrete sense at the domain boundaries.

7.2 Manufactured Solution for the Navier-Stokes Equations and the Temperature Equation

This subsection presents the manufactured solution for the system of partial differential equations (11). The manufactured solutions and the computations needed to derive the respective source terms were performed by the computer

algebra system Maple ® [?]. After the computation the terms were directly translated into FORTRAN source code, such that the manufactured solution could be directly integrated into the solver framework. In the following paragraphs the used solutions for the velocity vector (u_1, u_2, u_3), the pressure p and the temperature T will be formulated. The manufactured solution for the velocities was formulated as

$$u_1 = 2\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2 + 2\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_3$$

$$u_2 = 2\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_3 - 2\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1$$

$$u_3 = -2\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1 - 2\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2.$$

The manufactured solution for the pressure p was created as

$$p = \sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right).$$

Since the velocity field was created as a solenoidal field no pressure source had to be calculated. The manufactured solution for the temperature T reads

$$T = \sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\sin\left({x_3}^2\right).$$

The resulting source terms for the momentum balances read

$$\begin{aligned}
b_1 = {} & 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_1}^2 x_2 + 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_2}^3 + 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_3}^2 x_2 \\
& + 8\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_1}^2 x_3 + 8\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_2}^2 x_3 + 8\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_3}^3 \\
& + 7\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\sin\left({x_3}^2\right) + 4\left(\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\right)^2 x_1 + 4\left(\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\right)^2 x_3 \\
& - 4\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2 - 20\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_3 + 20\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2 - 6x_1
\end{aligned}$$

$$\begin{aligned}
b_2 = {} & -8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_1}^3 + 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_1}^2 x_3 - 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_2}^2 x_1 \\
& - 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_3}^2 x_1 + 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_2}^2 x_3 + 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_3}^3 \\
& + 13\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\sin\left({x_3}^2\right) - 4\left(\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\right)^2 x_2 - 4\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1 \\
& - 4\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_3 - 20\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1 + 20\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_3 - 2x_2
\end{aligned}$$

$$\begin{aligned}
b_3 = {} & -8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_1}^2 x_2 - 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_2}^3 - 8\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_3}^2 x_2 \\
& - 8\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_1}^3 - 8\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_2}^2 x_1 - 8\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right){x_3}^2 x_1 \\
& - 19\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\sin\left({x_3}^2\right) + 4\left(\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\right)^2 x_1 + 4\left(\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\right)^2 x_3 \\
& - 4\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2 + 20\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1 - 20\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2 - 6x_3.
\end{aligned}$$

Finally the source term for the temperature equation was calculated as

$$\begin{aligned}
b_T = {} & -4\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\cos\left({x_3}^2\right)\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2x_3 - 4\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\cos\left({x_3}^2\right)\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1x_3 \\
& + 4\sin\left({x_1}^2\right)\sin\left({x_3}^2\right)\sin\left({x_2}^2\right)\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1x_2 - 4\sin\left({x_1}^2\right)\sin\left({x_3}^2\right)\sin\left({x_2}^2\right)\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_2x_3 \\
& + 4\cos\left({x_2}^2\right)\sin\left({x_3}^2\right)\cos\left({x_1}^2\right)\cos\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1x_2 + 4\cos\left({x_2}^2\right)\sin\left({x_3}^2\right)\cos\left({x_1}^2\right)\sin\left({x_1}^2 + {x_2}^2 + {x_3}^2\right)x_1x_3 \\
& + 4{x_1}^2\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\sin\left({x_3}^2\right) + 4\sin\left({x_1}^2\right)\cos\left({x_2}^2\right){x_2}^2\sin\left({x_3}^2\right) \\
& + 4{x_3}^2\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\sin\left({x_3}^2\right) - 2\sin\left({x_1}^2\right)\cos\left({x_2}^2\right)\cos\left({x_3}^2\right) \\
& + 2\sin\left({x_1}^2\right)\sin\left({x_3}^2\right)\sin\left({x_2}^2\right) - 2\cos\left({x_2}^2\right)\sin\left({x_3}^2\right)\cos\left({x_1}^2\right).
\end{aligned}$$

7.3 Measurement of Error and Calculation of Order

In this subsection the result of the verification process via the formerly in subsection 7.2 stated manufactured solutions will be presented. To verify the solver framework using the Order-of-Accuracy test criterion an error measure has to be chosen on which the evaluation is based upon. According to [**?**] the error measure for each variable ϕ will be calculated at the end of each computation as the normalized L_2-norm of the deviation of the exact solution $\tilde{\phi}$, sometimes referred to as the normalized global error or the RMS error

$$\mathrm{err}(\phi) = \sqrt{\frac{\iiint_V \left(\phi - \tilde{\phi}\right)^2 \mathrm{d}V}{\iiint_V \mathrm{d}V}} \approx \sqrt{\frac{\sum_{n=1}^{N} \left(\phi_n - \tilde{\phi}_n\right)^2 V_n}{\sum_{n=1}^{N} V_n}}.$$

Herein N denotes the number of control volumes of the discretized solution domain V. Each control volume has the volume V_n. The numerical and the exact solution are evaluated at the center of each control volume for the error calculations. In order to calculate the normalized global error of the pressure p the same modification as in subsection 4.8 has to be performed on the analytical solution $\tilde{p}$, which in the case of the present work means to subtract the weighted mean value

$$\mathrm{mean}\left(\tilde{p}\right) = \frac{\iiint_V \tilde{p}\mathrm{d}V}{\iiint_V \mathrm{d}V} \approx \frac{\sum_{n=1}^{N} \tilde{p}_n V_n}{\sum_{n=1}^{N} V_n}$$

from it and to calculate the error as

$$\mathrm{err}(p,N) \approx \sqrt{\frac{\sum_{n=1}^{N} \left(p_n - \tilde{p}_n - \mathrm{mean}\left(\tilde{p}\right)\right)^2 V_n}{\sum_{n=1}^{N} V_n}}.$$

The following tables will list the calculated error terms for the velocities, the pressure and the temperature for different grid resolutions and both solver algorithms. The grid resolutions ($n \times n \times n$) are chosen such that n, the number of control volumes in each coordinate direction, will be a power of two. The Errors were evaluated after the maximal relative residual of all linear systems in one outer iteration fell below the threshold 1×10^{-14}. The concrete implementation of the tolerance criterion to accept a converged solution is found in section 6.4.2.

A comparison of the results leads to the first conclusion, namely that the results from the segregates solver algorithm and the coupled solver algorithm coincide which implies that the same discretization is used in both cases. According to [**?**] the next step comprises the calculation of the formal order of accuracy $\hat{p}_\phi$ for each variable that the program solves for. This can be done by evaluating the quotient of two different resolutions $n1$ and $n2$ as

$$\hat{p}_\phi = \frac{\log\left(\frac{\mathrm{err}(\phi,n1)}{\mathrm{err}(\phi,n2)}\right)}{\log\left(\frac{n1}{n2}\right)}.$$

The discretization techniques used in the present thesis yield an asymptotic discretization error of two [**?**]. Comparing with the results in the following tables REFERENCE shows the asymptotic behavior of the calculated order which leads to the conclusion that the discretization has been implemented with success.

7.4 Influence of the Under-Relaxation Factor for the Velocities

In section **??** the main purpose of introducing the pressure weighted interpolation method was to assure comparability of the generated results from the segregated solver algorithm and the coupled solver algorithm. Section **??** showed good agreement of the error calculated with the segregated solver using the pressure weighted interpolation method and the coupled solver. This section presents the results that are attained if the standard Rhie-Chow interpolation technique with different under-relaxation factors $\alpha_{\mathbf{u}}$ is used to interpolate the velocities to the boundary faces instead of the pressure weighted interpolation technique. Figure 7 shows the normalized L_2 norm of the error of the results for the velocity u_1 for different under-relaxation factors. The L_2 norms have been normalized by the L_2 norm of the respective calculation that uses the pressure weighted interpolation method.

It is evident that the higher the under-relaxation factor the smaller the deviation of the results obtained with the standard Rhie-Chow interpolation technique from the results obtained with the pressure weighted interpolation method. This can be justified by the fact that, for an under-relaxation factor $\alpha_{\mathbf{u}} = 1$, the pressure weighted interpolation and the

Resolution	Error of **u** from Segregated Solver	Error of **u** from Coupled Solver	Observed Order $\hat{p}$
8x8x8	3.1704502301157865E-002	3.1704502290288525E-002	
	3.0370110307466745E-002	3.0370110306195783E-002	
	3.1636095150284566E-002	3.1636095141306442E-002	
16x16x16	7.9143566730061794E-003	7.9143566730065350E-003	2.0021
	7.5343146124403392E-003	7.5343146124409056E-003	2.0111
	7.8113167415358713E-003	7.8113167415361021E-003	2.0179
32x32x32	1.9333028867476348E-003	1.9333250814097063E-003	2.0334
	1.8675991971385400E-003	1.8676512523175990E-003	2.0123
	1.8948139100889829E-003	1.8948223205995562E-003	2.0435
64x64x64	4.7468941949359882E-004	4.7468942001527124 2E-004	2.0260
	4.6292599339559415E-004	4.6292599348776502 7E-004	2.0123
	4.6300082035502932 8E-004	4.6300082038027822 9E-004	2.0330
128x128x128	1.1722669722909498 1E-004		2.0177
	1.1500240264982475 9E-004		2.0091
	1.1404711828463543 7E-004		2.0214
256x256x256			

Table 1: Comparison of the errors of the velocity calculated by the segregated and the coupled solver for different grid resolutions and the resulting order of accuracy

Resolution	Error of p from Segregated Solver	Error of p from Coupled Solver	Observed Order $\hat{p}$
8x8x8	0.18303589312168636	0.18303589466468098	
16x16x16	8.9677276839834036E-002	8.9677276839825765E-002	1.0293
32x32x32	3.7964856888161008E-002	3.7964815814009345E-002	1.2401
64x64x64	1.4387801006225458 1E-002	1.4387801006156549 6E-002	1.3998
128x128x128	5.1607056488520370 1E-003		1.4792
256x256x256			

Table 2: Comparison of the errors of the pressure calculated by the segregated and the coupled solver for different grid resolutions and the resulting order of accuracy

Resolution	Error of T from Segregated Solver	Error of T from Coupled Solver	Observed Order $\hat{p}$
8x8x8	4.0675323775412001E-003	4.0675323775378296E-003	
16x16x16	1.1324773183947739E-003	1.1324773183947821E-003	1.8447
32x32x32	2.9310098015820527E-004	2.9310096243402903E-004	1.9500
64x64x64	7.4100219744699974 0E-005	7.4100219743344219 9E-005	1.9838
128x128x128	2.9310098015820527E-004		1.9951
256x256x256			

Table 3: Comparison of the errors of the temperature calculated by the segregated and the coupled solver for different grid resolutions and the resulting order of accuracy

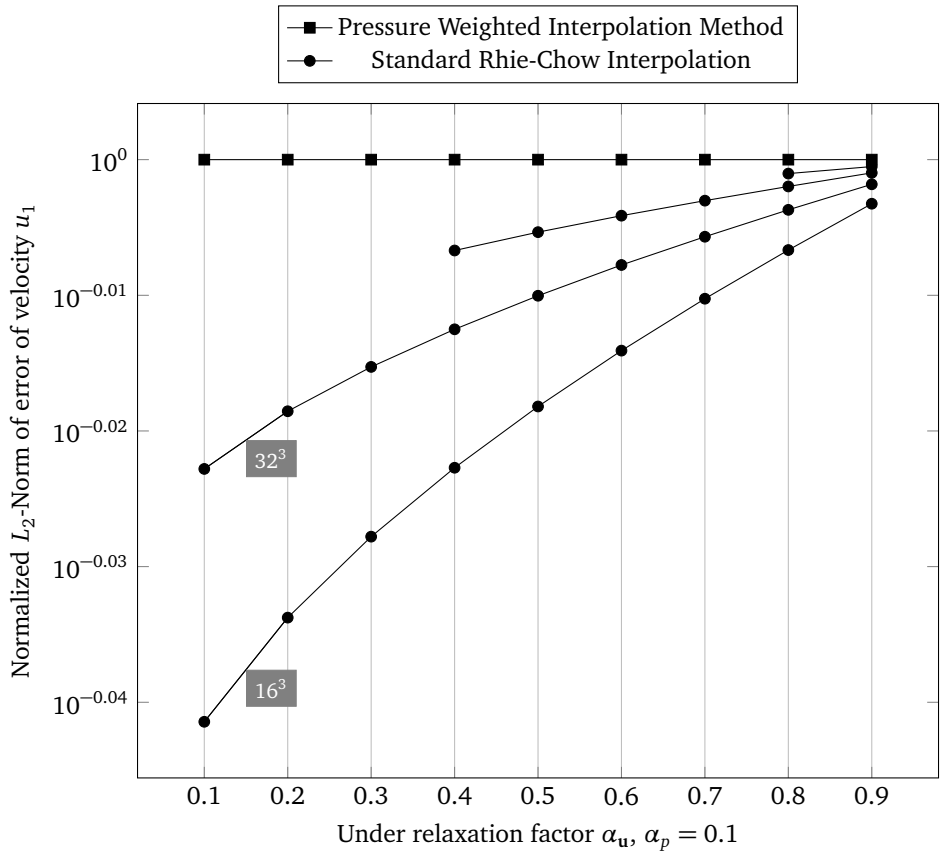

Figure 7: Comparison of calculated error for different under relaxation factors $\alpha_{\mathbf{u}}$ on a grid with different grid resolutions unknowns

Rhie-Chow interpolation would yield the same result according to equation (21). Furthermore it can be observed that the deviation decreases with increasing mesh resolution which is the expected behaviour [**?**].

8 Comparison of Solver Concepts

8.1 Convergence Behaviour on Locally Refined Block Structured Grids with Different Degrees of Coupling

Show how the implicit treatment of block boundaries maintains (high) convergence rates. Plot Residual over number of iterations. Plot Wall time for a single block using SIP, KSP, COUPLED for different grid resolutions.

8.2 Parallel Performance

In many cases, scientific code is used to solve complex problems regarding memory requirements to make calculation results available within short time. In both scenarios, code that is able to run in parallel can alleviate the mentioned challenges. Code that runs in parallel can allocate more memory resources which makes the calculation of complex problems feasible. If the code is scalable the program execution can be shortened by using more processors to solve a problem of constant size.

The solver framework that has been developed in the course of the present thesis, has been parallelized using the PETSc library. After introducing the used hardware and software, the central measures of parallel performance are presented. Then preliminary test results using low-level benchmarks are performed, which establish upper performance bounds on the parallel efficiency and the scalability of the developed solver framework. The results of the efficiency evaluation of the solver framework is presented in the last subsections.

8.2.1 Employed Hardware and Software – The Lichtenberg-High Performance Computer

All performance analyses that are presented in this thesis were conducted on the Lichtenberg-High Performance Computer, also known as *HHLR* (*Hessischer Hochleistungsrechner*) [**?**]. The cluster consist of different sections according to the used hardware. Throughout the thesis, tests were performed using the first and the second MPI section of the cluster. The first section consists of 705 nodes of which each runs two Intel®Xeon®E5-2670 processors and offers 32GB of memory. The second section consists of 356 nodes of which each runs two Intel Xeon E5-2680 v3 processors and offers 64GB of memory. As interconnect for both sections FDR-14 InfiniBand is used.

All tests programs were compiled using the Intel compiler suite version 15.0.0 and the compiler options

```
−O3 −xHost
```

As MPI implementation Open MPI version 1.8.2 was chosen. Furthermore the PETSc version 3.5.3 was configured using the options

```
−−with−blas−lapack−dir=/shared/apps/intel/2015/composer_xe_2015/mkl/lib/intel64/ \
−−with−mpi−dir=/shared/apps/openmpi/1.8.2_intel \
COPTFLAGS="−O3 −xHost" \
FOPTFLAGS="−O3 −xHost" \
CXXOPTFLAGS="−O3 −xHost" \
−−with−debugging=0 \
−−download−hypre \
−−download−ml
```

It should be noted that as the configurations options show, to maximize the efficiency of PETSc, a math kernel library should be used that has been optimized for the underlying hardware architecture as is in the case of the present thesis the Intel *MKL* (*Math Kernel Library*). It should be noted that also the Open MPI library has been compiled using the Intel compiler suite.

8.2.2 Measures of Performance

This section establishes the needed set of measures to evaluate the performance of a solver program, which will be used in the following sections. The first measure is the plain measure of runtime T_P taken by a computer to solve a given problem, where $P \in \mathbb{N}$ denotes the number of involved processes. This so called *wall-clock* time can be measured directly by calling subroutines of the underlying operating system and corresponds to the human perception of the time, that has passed. It must be noted, that this time does not correspond to the often mentioned *CPU* time. In fact, CPU time is only one contributor to wall-clock time. Wall-clock time further contains the time needed for communication and I/O and hence considers idle states of the processor. On the other side CPU time only considers the time in which the processor is actively working. This makes wall-clock time not only a more complete but also more accurate time measure when dealing with parallel processors, since processor idle times due to communication are actively considered while neglected in CPU time.

While wall-clock time is an absolute measure that can be used to compare different solver programs, further relative measures are needed to evaluate the efficiency of one program regarding the parallelisation implementation. The main purpose of these measures is to attribute the different causes of degrading efficiency due to heavy parallelisation to the different contributing factors. A simple model [**?**, **?**] considers three contributions, that form the total efficiency

$$E_P^{tot} = E_P^{num} \cdot E_P^{par} \cdot E_P^{load}.$$

The *numerical efficiency*

$$E_P^{num} := \frac{\text{FLOPS}(1)}{P \cdot \text{FLOPS}(P)}$$

considers the degradation of the efficiency of the underlying algorithm due to the parallelisation. Many efficient algorithms owe their efficiency to recursions inside the algorithm. In the process of decomposing this recursions, the efficiency of the algorithm degrades. It follows that this efficiency is completely independent of the underlying hardware.

The *parallel efficiency*

$$E_P^{par} := \frac{\text{TIME(parallel Algorithm on one processor)}}{P \cdot \text{TIME(parallel Algorithm on } P \text{ processors)}}$$

describes the impact of the need for inter process communication, if more than one processor is involved in the solution process. It should be noted, that this form of efficiency does explicitly exclude any algorithm related degrading, since the time measured corresponds to the exact same algorithms. It follows that the parallel efficiency only depends on the implementation of the communication and the hardware related latencies.

The *load balancing efficiency*

$$E_P^{load} := \frac{\text{TIME(calculation on complete domain)}}{P \cdot \text{TIME(calculation on biggest subdomain)}}$$

is formed by the quotient of the wall times needed for the complete problem domain and partial solves on subdomains. This measure does neither depend on hardware nor on the used implementation. Instead it directly relates to the size and partition of the grid.

It is not possible to calculate all three efficiencies at the same time using only plain wall clock time measurements of a given application. Different solver configurations have to be used to calculate them separately. Since the focus of investigation of the present thesis does not lie on load balancing, for the remainder of the thesis $E_P^{load} = 100\%$ is assumed. This does not present a considerable drawback, since an ideal load balancing is easily obtainable nowadays by the use of sophisticated grid partitioning algorithms [**?**] REFERENCES. Using identical algorithms for different numbers of involved processes implicitly achieves $E_P^{num} = 100\%$. In this case the parallel efficiency of an application can be measured through the quotient of the needed wall clock time. To measure the numerical efficiency of an algorithm the respective hardware counters have to be evaluated. This can be done using the built in log file functionality of PETSc as presented in section REFERENCE. Hence the determination of numerical efficiency does not rely on wall clock time.

Another common performance measure is the *Speed-Up*

$$S_P = \frac{T_1}{T_P} = P \cdot E_P^{tot}.$$

Speedup and parallel efficiency characterize the parallel scalability of an application and determine the regimes of efficient use of hardware resources.

8.2.3 Preliminary Upper Bounds on Performance – The STREAM Benchmark

Scientific applications that solve partial differential equations rely on sparse matrix computations, which usually exhibit the sustainable memory bandwidth as bottleneck with respect to the runtime performance of the program [**?**]. The purpose of this section is to establish a frame in terms of an upper bound on performance in which the efficiency of the developed solver framework can be evaluated critically. As common measure for the maximum sustainable bandwidth, low-level benchmarks can be used, which focus on evaluating specific properties of the deployed hardware architecture. In this case the STREAM benchmark suite [**?**, **?**] provides apt tests, which are designed to work with data sets that exceed the cache size of the involved processor architecture. This forces the processors to stream the needed data directly from

the memory instead of reusing the data residing in their caches. These types of tests can be used to calculate an upper bound on the memory bandwidth for the CAFFA framework.

In terms of parallel scalability, the STREAM benchmark can also be used as an upper performance bound. According to [**?**] the parallel performance of memory bandwidth limited codes correlates with the parallel performance of the STREAM benchmark, i.e. a scalable increase in memory bandwidth is necessary for scalable application performance. The intermediate results of the benchmark can then be used to test different configurations that bind hardware resources to the involved processes. Before presenting results the different binding configurations will be explained.

The first configuration sequentially binds the processes to the cores beginning on the first socket. When every core has a bound process the binding algorithm binds the following processes to cores of the second socket. The second configuration binds the processes in a round robin manner regarding the sockets. This configuration in difference to the second configuration binds one process to three cores. Figures 8.2.3,8.2.3 and 8.2.3 demonstrate the different binding options for two sockets and processors with twelve cores each, when eight processes are to be bound to the resources.

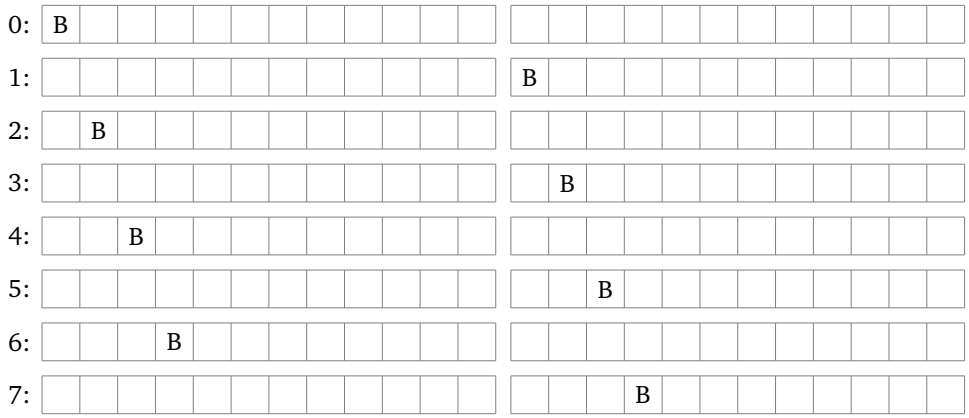

Figure 8: Default binding using Open MPI on a node with two sockets and processors with each twelve cores

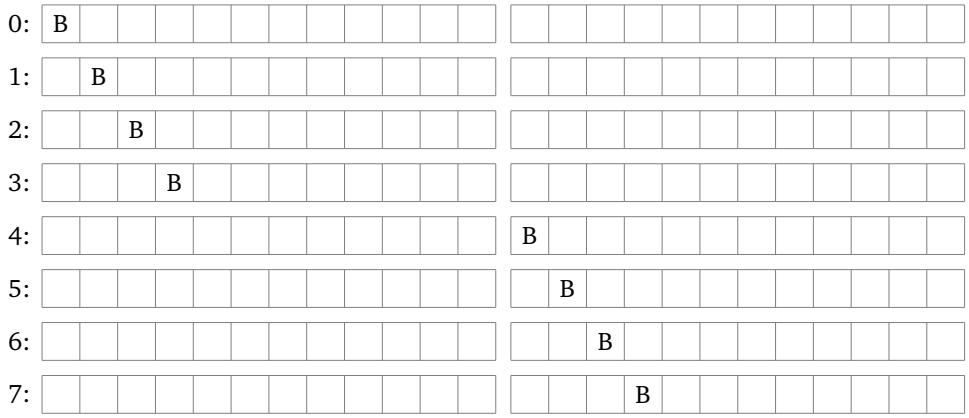

Figure 9: Process binding using Open MPI and map-by ppr:8:node map-by ppr:4:socket on a node with two sockets and processors with each twelve cores

8.2.4 Optimization of Sequential Solver Configuration

Compare runtime for different solver configurations BiCGStab+ICC, different multigrid algorithms

8.2.5 Evaluation of Numerical Efficiency of the Solver Algorithm

8.2.6 Discussion of Results for Parallel Efficiency

8.2.7 Speedup Measurement for Analytic Test Cases

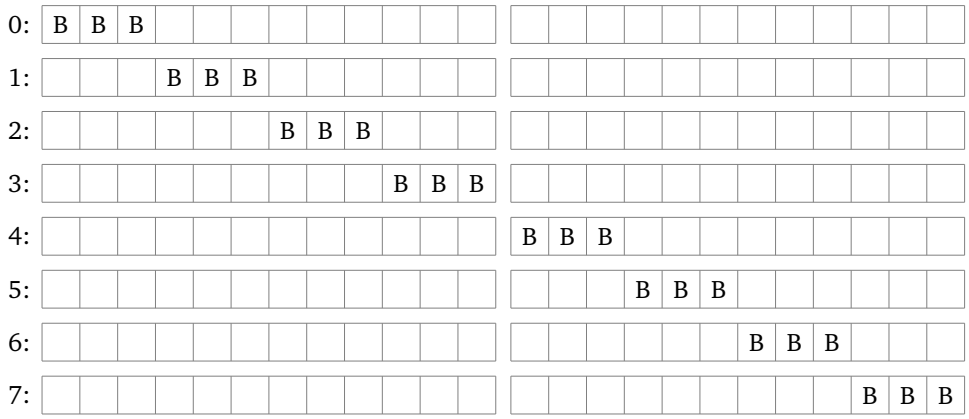

Figure 10: Process binding using Open MPI and map-by ppr:8:node map-by ppr:4:socket:PE=3 on a node with two sockets and processors with each twelve cores

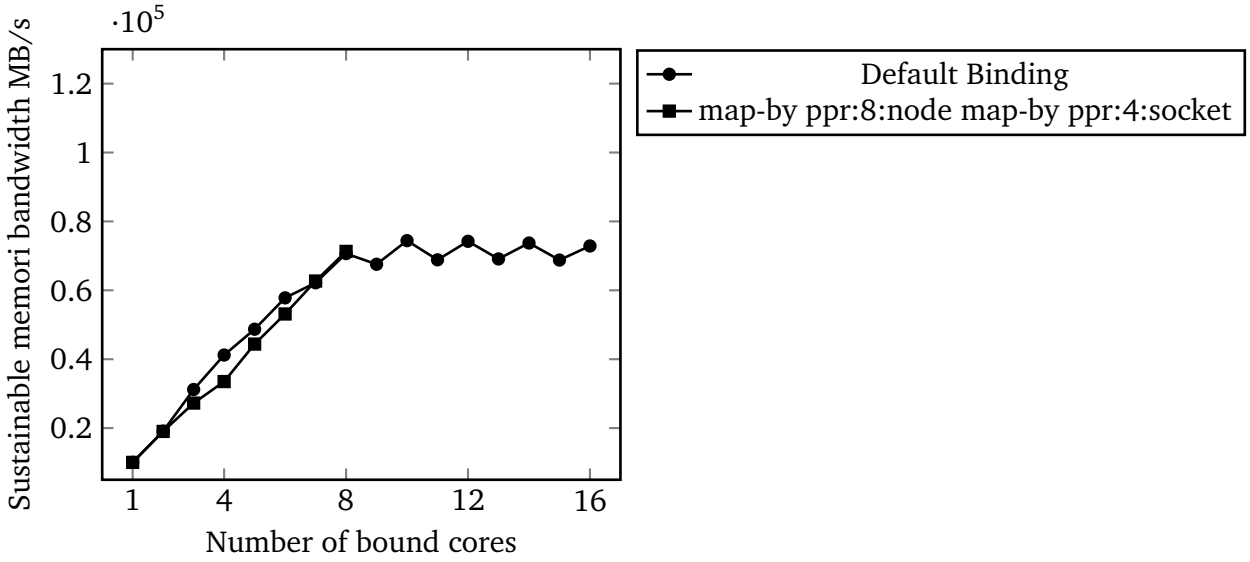

Figure 11: Sustainable memory bandwidth as determined by the STREAM benchmark (Triad) for different process binding options on one node of the MPI1 section

8.3 Test Cases with Varying Degree of Non-Linearity

As Peric says I want to prove that the higher the non-linearity of NS, the better relative convergence rates can be achieved with a coupled solver. Fi

8.3.1 Transport of a Passive Scalar – Forced Convection

8.3.2 Buoyancy Driven Flow – Natural Convection

8.3.3 Flow with Temperature Dependent Density – A Highly Non-Linear Test Case

Maybe I could consider two test cases, one with oscillating density and one with a quadratic polynomial. Interesting would be also to consider the dependence of convergence on another scalar transport equation

8.4 Realistic Testing Scenarios – Benchmarking

Also consider simple load balancing by distributing matrix rows equally

8.4.1 Flow Around a Cylinder 3D – Stationary

Describe Testing Setup (Boundary conditions and grid). Present results and compare them with literature.

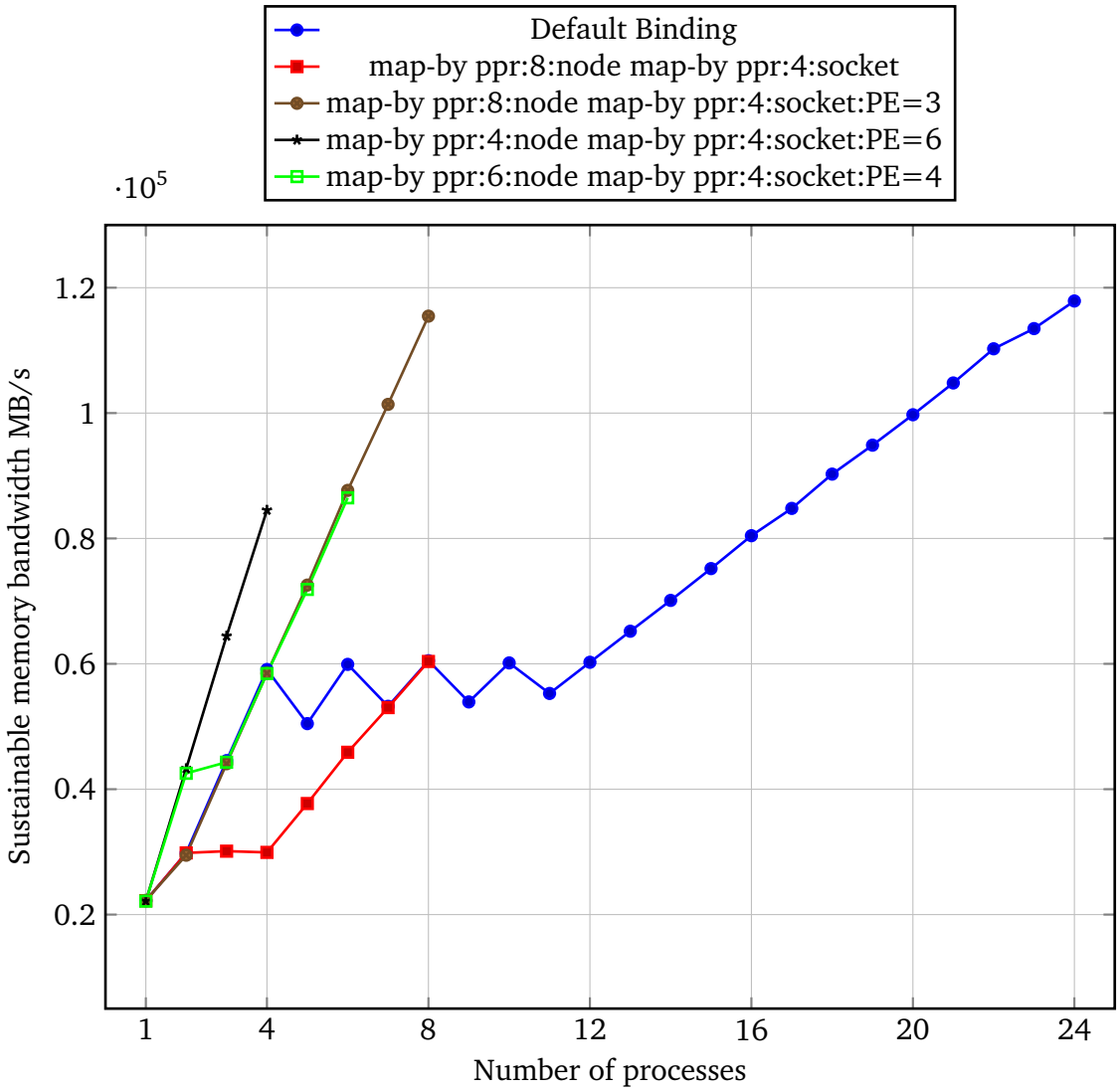

Figure 12: Sustainable memory bandwidth as determined by the STREAM benchmark (Triad) for different process binding options on one node of the MPI2 section

8.4.2 Flow Around a Cylinder 3D – Instationary

- http://www.featflow.de/en/benchmarks/cfdbenchmarking/flow/dfg_flow3d/dfg_flow3d_configuration.html

Describe Testing Setup (Boundary conditions and grid). Present results and compare them with literature.

8.4.3 Heat-Driven Cavity Flow

- http://www.featflow.de/en/benchmarks/cfdbenchmarking/mit_benchmark.html

Describe Testing Setup (Boundary conditions and grid). Present results and compare them with literature.

8.5 Realistic Testing Scenario – Complex Geometry

9 Conclusion and Outlook

Turbulence (turbulent viscosity has to be updated in each iteration), Multiphase (what about discontinuities), GPU-Accelerators, Load-Balancing, dynamic mesh refinement, Counjugate Heat Transfer with other requirements for the numerical grid, grid movement, list some papers here) Identify the optimal regimes / conditions for maximizing performance. Each solver concept has its strengths and weaknesses. Try other variants of Projection Methods like SIMPLEC, SIMPLER, PISO or PIMPLE (OpenFOAM) Develop physics based preconditioner extend field interlacing to other variables (slight performance tuning) use accelerators to solve linear systems use dynamic loadbalancing and adaptive refinement techniques SIMPLET

References