DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2017

A predictor corrector method for a Finite Element Method for the variable density Navier-Stokes equations

ISABEL HAASLER

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

A predictor corrector method for a Finite Element Method for the variable density Navier-Stokes equations

ISABEL HAASLER

Degree Projects in Scientific Computing (30 ECTS credits)
Degree Programme in Applied and Computational Mathematics (120 credits)
KTH Royal Institute of Technology year 2017
Supervisor at KTH: Johan Hoffman, Johan Jansson
Examiner at KTH: Michael Hanke

TRITA-MAT-E 2017:64
ISRN-KTH/MAT/E--17/64--SE

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

Constructing Finite Element Methods for the incompressible Navier-Stokes equations that are computationally feasible requires the use of pressure segregation methods such as the predictor corrector scheme. From an algebraic point of view these methods correspond to a preconditioned Richardson iteration on the pressure Schur complement. In this work the theory is extended to a variable density model. We suggest a modified preconditioner and investigate its performance experimentally.

Sammanfattning

En prediktorkorrigeringsmetod för en Finita Elementmetoden för Navier-Stokes-ekvationer med variabel densitet

Konstruktion av Finita Elementmetoder för de inkompressibla Navier-Stokes ekvationer som är beräkningsmässigt genomförbara kräver användning av trycksegregeringsmetoder såsom prediktorkorrigeringsschemat. Från en algebraisk synvinkel motsvarar dessa metoder en förkonditionerad Richardsoniteration på tryck-Schur-komplementet. I detta arbete utvidgas teorin till en variabel-densitet-modell. Vi föreslår en modifierad preconditioner och undersöker prestanda experimentellt.

Acknowledgements

First of all, I would like to express my gratitude to my supervisors Johan Hoffman and Johan Jansson for their guidance, advice and input to my work, offering comments on the report and giving me the opportunity to visit the research group at the Basque Institute of Applied Mathematics (BCAM).

Moreover, I want to thank everyone who I got the chance to meet at BCAM for their guidance, support and hospitality, especially Daniel, Massimiliano, Magaly, Christina and Andrea. You made my stay in Bilbao a fruitful, inspirational and enjoyable time.

Special thanks to my fellow student and friend Georg Neumüller whom I shared this experience the past few months with, going through good and challenging times together.

Finally and most importantly, I want to thank my parents for their invaluable support and encouragement for everything I do.

Contents

Chapter 1

Introduction

The Navier-Stokes equations are one of the foundation for engineering applications concerned with the description of fluids. Among others in mechanical, chemical and biomedical engineering the simulation and analysis of the dynamics of fluids is a fundamental topic.

As the Navier-Stokes equations are analytically solvable only under some special conditions and computational possibilities advance quickly, the development of efficient numerical solving strategies is of great interest and a particularly active research area. One field in Computational Fluid Dynamics (CFD) is concerned with the development of Finite Element Methods (FEMs) for the Navier-Stokes equations.

This approach is particularly challenging when the Reynolds number is large, as vortices appear at a large variety of scales and resolving all of them requires a massive amount of computational power that exceeds even the capability of supercomputing clusters. A residual-based numerical stabilization of the FEM has been shown to serve as a parameter free turbulence model [14]. With this approach small scale vortices do not need to be resolved in full detail to extract certain averaged information about the fluid mechanics. Hence, FEM simulations are viable even for high Reynolds numbers if computations are parallelized.

The FEM for the Navier-Stokes equations results in large algebraic systems, which should be solved with iterative methods, as these require less memory and are more easily parallelizable. Unfortunately, it is troublesome to find well-scaling preconditioners for the monolithic problem [7]. For incompressible flow it is efficient to decouple the velocity and pressure equations of the Navier-Stokes system and

solve them separately. In order for the solutions to converge to the monolithic solution, the decoupled equations are solved iteratively in the form of a fixed-point method. With a suitable preconditioner this approach has proven to be sufficiently efficient.

However, there have not yet been many attempts to extend this method to the variable density incompressible Navier-Stokes equations. A variable density model can be used to model the interaction of two fluids. Such a two-phase model is for example required for applications in marine engineering. The efficient numerical modeling of floating structures with breaking waves, at high Reynolds numbers, is an ongoing challenge.

The objective of this master thesis is to extend the iterative solving strategy to an FEM for the variable density incompressible Navier-Stokes equations. In particular, we aim to find an effective preconditioner.

This work is outlined as follows. Chapter 2 gives an introduction to the mathematical background for the following work. After presenting the Navier-Stokes equations a stabilized FEM is derived. Pressure segregation methods are introduced and two schemes presented. We describe two approaches to model interfaces, the interface tracking Arbitrary Lagrangian-Eulerian (ALE) method and the interface capturing level set method. A short introduction to the FEniCS project which forms the computational framework for this master thesis is given.

Our model for the variable density incompressible Navier-Stokes equations is portrayed in Chapter 3. A stabilized Finite Element Method is developed and explained. Also, the test cases we base our numerical investigation on are presented.

In Chapter 4 we follow an algebraic approach to predictor corrector methods. This allows us to derive a suitable preconditioner for the FEM with a constant but arbitrarily chosen density. With some additional considerations and heuristics we extend the derived method to suggest a preconditioner for the variable density model.

The performance of the proposed preconditioner is investigated numerically in Chapter 5. We will see that, as for the constant density model, the time step size can be increased significantly when a Schur preconditioning term is inserted into the method. The convergence of the fixed point iteration is examined for the system without preconditioning and with different parameters for the preconditioner. The accumulated number of fixed point iterations is compared for these

different preconditioners and various time step sizes. For the 3D experiments we also measure the computation time and put it in relation to the number of fixed point iterations.

In Chapter 6 we summarize our observations and give some conclusive remarks.

Chapter 2

Background

The present master thesis project is concerned with computational aspects for the incompressible variable density Navier-Stokes equations. In this chapter the mathematical foundation for the following chapters is laid.

In Section 2.1 the incompressible Navier-Stokes equations are introduced. The Finite Element Method (FEM) is presented and a stabilized FEM for the Navier-Stokes equations is formulated. We show the form of the algebraic system induced through the FEM as we shall later follow an algebraic approach to the Finite Element formulation.

These systems can become enormous in size, especially when high Reynolds number flows are considered and vortices are resolved at all scales. This gives rise to the need of high performance computing methods. Such methods perform well when the velocity and pressure parts of the Navier-Stokes equations are decoupled and solved separately. In Section 2.2 we present two schemes that follow this principle, the incremental projection method and the predictor corrector method. A short introduction to the implementation and solving strategy in our problem formulation is given for decoupled schemes.

One aim of the variable density solver is to simulate floating objects. In this context one needs to model both the interaction of water and air and the interaction of a structure with the fluids. For these two kinds of problems different methods are advantageous. In Section 2.3 the interface tracking Arbitrary Lagrangian-Eulerian (ALE) method is presented, which is used for the fluid structure interaction. Section 2.4 is an introduction to the mathematical description of two phase flow and gives an overview to level set methods that are often used to

model the interaction of two fluids.

Finally, the FEniCS project which gives the computational framework for the simulations in this thesis is presented in Section 2.5.

2.1 A stabilized Finite Element Method for the Navier-Stokes equations

This work deals with the numerical modeling of fluid dynamics. The mathematical basis for describing the motion of viscous fluids is given by the Navier-Stokes equations. Although the existence and uniqueness of solutions to the Navier-Stokes equations are still an open problem, one can find solutions that fulfill the equations in a weak sense with the help of numerical methods.

In this chapter, a Finite Element Method (FEM) for solving the Navier-Stokes equations is introduced. FEMs are a special case of Galerkin's method, which is a numerical approach to solving differential equations. The basic idea is to approximate solutions in a finite-dimensional function space by satisfying an orthogonality condition. Functions fulfilling this Galerkin orthogonality are called weak solutions. A certain choice of piecewise polynomial basis functions for the function space defines the Finite Element Method.

Moreover, a stabilization strategy is presented that allows to search for the solution in a space of piecewise linear functions.

The descriptions in this section are primarily based on the works by Hoffman et al. [15, 11] and Eriksson et al. [21].

2.1.1 The incompressible Navier-Stokes equations

The basis for mathematical modeling of fluid mechanics are the Navier-Stokes equations. In this thesis only incompressible Newtonian fluids are considered. For now, we assume a fluid with constant density ρ. We divide the momentum and pressure equation by this density and introduce the kinematic viscosity $\nu = \mu/\rho$, where μ is the dynamic viscosity. Hence, the fluid is characterized solely by its kinematic viscosity ν and the Navier-Stokes equations take the form

$$\dot{u} + u \cdot \nabla u - \nu \Delta u + \nabla p = f \tag{2.1}$$

$$\nabla \cdot u = 0, \tag{2.2}$$

where $u(x,t)$ is the velocity vector and $p(x,t)$ the pressure of the described fluid. The source f models an external force acting on the fluid. In the following, we consider solutions in a space domain $\Omega \subset \mathbb{R}^3$ and a time interval $I = [0,T]$.

It is often convenient to express the Navier-Stokes equations in its dimensionless form. Therefore, we introduce a characteristic length L and a characteristic velocity U. These define the characteristic time $T = L/U$. With these quantities dimensionless parameters can be defined as $x' = x/L$, $u' = u/U$ and $t' = t/T$. Hence, dropping the prime notation, the dimensionless form of the incompressible momentum equations reads

$$\dot{u} + u \cdot \nabla u - Re^{-1}\Delta u + \nabla p = f,$$

where the pressure and the force are rescaled and the Reynolds number is defined as

$$Re = \frac{UL}{\nu}.$$

Hence, small viscosities ν correspond to a large Reynolds number Re and vice versa. The Reynolds number can be used to describe flow characteristics. At a Reynolds number of order $10^2 - 10^3$ flows transition from laminar to turbulent flow [11]. In this thesis we are occupied with flows where the Reynolds number is very high and that are turbulent. As it becomes computationally expensive to resolve the vortices at all scales, numerical methods need to be exploited to make computations feasible.

2.1.2 Weak form

Fundamental to the FEM is the formulation of weak solutions introduced in the section. First we express the Navier-Stokes equations (2.1)-(2.2) in residual form, that is

$$R(\hat{u}) = 0 \quad \text{in } \Omega \times I, \tag{2.3}$$

with $\hat{u} = (u,p)$ and

$$R(\hat{u}) = \begin{pmatrix} \dot{u} + u \cdot \nabla u - \nu\Delta u + \nabla p - f \\ \nabla \cdot u \end{pmatrix}$$

Instead of finding an exact solution $\hat{u}$ satisfying (2.3), Galerkin methods seek a solution fulfilling the Galerkin orthogonality

$$(R(\hat{u}), \hat{v}) = 0 \tag{2.4}$$

for all $\hat{v} \in V$, where V and $(\cdot, \cdot)$ are an appropriate test space and a suitable inner product.

While the residual in (2.3) has to become pointwise zero, condition (2.4) requires the residual to vanish only in an averaged sense. Condition (2.3) is thereby relaxed and we call solutions satisfying the Galerkin orthogonality weak solutions.

Let us consider the function space

$$V = \left\{ \hat{v} \in \left[H^1(\Omega \times I) \right]^4 : v = 0 \text{ on } \partial\Omega \times I \right\}$$

and let $(\cdot, \cdot)$ denote the $L^2(\Omega \times I)^m$ scalar product, where $m = 1, 3$.

The weak formulation reads then: Find $\hat{u} = (u, p) \in V$ such that

$$r(\hat{u}, \hat{v}) = (\dot{u} + u \cdot \nabla u + \nabla p - f, v) + (\nu \nabla u, \nabla v) + (\nabla \cdot u, q) = 0 \quad (2.5)$$

for all $\hat{v} = (v, q) \in V$. The functions $\hat{v} \in V$ are called test functions and $\hat{u} \in V$ trial functions. Note that well-posedness of all the terms in (2.5) requires some further smoothness assumptions on $\hat{u}$. For a more thorough understanding the reader is referred to [15, 11].

2.1.3 Finite element formulation

To define a Finite Element Method the weak formulation (2.5) needs to be discretized in time and space.

For the $cG(1)cG(1)$ Galerkin Finite Element method [15, 11] we use continuous piecewise linear functions as both test and trial functions. The time integral is discretized according to

$$0 = t_0 < t_1 < \cdots < t_N = T,$$

which results in intervals $I_n = (t_{n-1}, t_n)$ of length $k_n = t_n - t_{n-1}$.

At each time step t_n consider a mesh T_n and let $W^n \subset H^1(\Omega)$ be the function space of all continuous piecewise linear functions on T_n. The subspace of functions satisfying homogeneous Dirichlet boundary conditions is denoted $W_0^n \subset W^n$.

For the $cG(1)cG(1)$ method we seek solutions $\hat{u}^n = (u^n, p^n) = (u(t_n), p(t_n))$, that are continuous piecewise linear in space and time. The finite element formulation for the Navier-Stokes equations reads then: For $n = 1, \ldots, N$, find $\hat{u}^n \in [W_0^n]^3 \times W^n$ such that

$$r(\hat{u}^n, \hat{v}^n) = 0 \quad (2.6)$$

for all $\hat{v} \in [W_0^n]^3 \times W^n$. The weak residual is defined as

$$r(\hat{u}^n, \hat{v}) = \left((u^n - u^{n-1})k_n^{-1} + \bar{u}^n \cdot \nabla \bar{u}^n + \nabla p^n - f, v\right) + (\nu \nabla \bar{u}^n, \nabla v) + (\nabla \cdot \bar{u}^n, q) + SD_\delta\left(\bar{u}^n, p^n; v, q\right), \tag{2.7}$$

where

$$\bar{u}^n = \theta u^n + (1-\theta)u^{n-1} \quad \text{with } \theta \in [0,1].$$

The parameter θ determines the time stepping method. With $\theta = 0$ we get a forward (explicit) Euler scheme and with $\theta = 1$ a backward (implicit) scheme. In this project $\theta = \frac{1}{2}$ is used, which corresponds to a Crank-Nicholson scheme.

Moreover, a weigthed Least-Squares stabilization term is added, which takes the form

$$SD_\delta\left(\bar{u}^n, p^n; v, q\right) = (\delta_1(\bar{u}^n \cdot \nabla \bar{u}^n + \nabla p^n - f), \bar{u}^n \cdot \nabla v + \nabla q) + (\delta_2 \nabla \cdot \bar{u}^n, \nabla \cdot v). \tag{2.8}$$

Without the stabilization term, the solution to (2.6) is only stable if the Reynolds number is low and the velocity and pressure spaces satisfy the Ladyzhenskaya-Babuska-Brezzi condition, that is stated as follows, for the ideal case of Stokes flow with $Re = 0$.

Ladyzhenskaya-Babuska-Brezzi (LBB) inf-sup condition ([29, 9])**.** The Finite Element Method for the Stokes problem is stable for $h \to 0$ if for the velocity space V_h and the pressure space P_h it exists a mesh-independent constant γ such that

$$\min_{p \in P_h} \max_{v \in V_h} \frac{(p, \nabla \cdot v)}{\|p\| \|\nabla v\|} \geq \gamma > 0$$

with $\|\cdot\|$ the norm of the corresponding velocity and pressure spaces.

A way to satisfy the LBB condition is to use a higher order polynomial for the velocity than for the pressure. Since we want to use first order polynomials in both velocity and pressure, the LBB condition is not satisfied. The FEM is instead stabilized by adding the term (2.8) to the method.

2.1.4 The algebraic system

The finite element formulation (2.6) can be expressed as an algebraic system which has to be solved in every time step. With a basis for

the test and trial function spaces the solution to (2.6) can be written as a linear combination of the basis functions. For now we drop the time index for clearer reading. Let $\{\phi_1, \ldots, \phi_{m_u}\}$ denote a basis for W_0 and $\{\psi_1, \ldots, \psi_{m_p}\}$ denote a basis for W. The dimension of W_0 is the number of inner nodes m_u, while the dimension of W is the number of total nodes m_p, including nodes lying on the boundary. Then the solution $\hat{u} = (u, p)$ to the finite element method can be written as

$$u_i(x) = \sum_{j=1}^{m_u} \xi_{i,j}\phi_j(x) \quad \text{for } i = 1, 2, 3 \tag{2.9}$$

$$p(x) = \sum_{j=1}^{m_p} \zeta_j \psi_j(x) \tag{2.10}$$

with $\xi_{i,j}, \zeta_j \in \mathbb{R}$ the coefficients for each basis function.

Recall that we use piecewise linear trial and test functions in our finite element formulation. A suitable basis is then defined by the piecewise linear functions on the mesh T with nodes x_k such that

$$\begin{aligned} \phi_j(x_k) &= \delta_{jk}, \quad \text{where } j, k \in \{1, \ldots, m_u\} \\ \psi_j(x_k) &= \delta_{jk}, \quad \text{where } j, k \in \{1, \ldots, m_p\}. \end{aligned}$$

These functions are usually called hat functions in the one dimensional case and tent functions in three dimensions. Note that most pairs of basis functions have no common support. More specifically, the product $\phi_j\phi_k$ is only non-zero when the nodes x_j and x_k share a common edge.

The algebraic system is obtained by plugging the linear combinations (2.9) and (2.10) into the finite element formulation and solving for the coefficients $\xi_{i,j}$ and ζ_j. Since (2.6) has to hold for all $\hat{v} \in [W_0^n]^3 \times W^n$, one can instead demand (2.7) to hold for all functions ϕ_j and ψ_j which form a basis of the test space. Hence, we need to solve a system of $3m_u + m_p$ equations: Find $\xi_{i,j}$ and ζ_j such that

$$r\left((u, p), (\phi_{j_1}, \phi_{j_2}, \phi_{j_3}, \psi_k)\right) = 0$$

for all $j_1, j_2, j_3 = 1, \ldots, m_u$ and $k = 1, \ldots, m_p$.

As a simple example the matrix corresponding to the first term in the finite element formulation is derived for the one dimensional case.

Example (Mass matrix in one dimensional case)**.** Let W_0 be the space of piecewise linear continuous functions on I with a basis of hat functions

$\{\phi_1, \ldots, \phi_m\}$ defined by $\phi_j(x_k) = \delta_{jk}$, where $x_1 < x_2 < \cdots < x_m$ denote the inner nodes. Let $v \in W_0$ be the trial functions and $u \in W_0$ the test functions.

The term (u, v) in the finite element formulation can be expressed as a matrix by plugging in the linear combination $u = \sum_{j=1}^{m} \xi_j \phi_j$ and demanding the weak residual to be zero for every choice of ϕ_j as trial function v. Hence,

$$(u, v) = \int_\Omega \sum_{j=1}^{m} \xi_j \phi_j(x) \phi_i(x) \, \mathrm{d}x = \sum_{j=1}^{m} \xi_j \int_\Omega \phi_j(x) \phi_i(x) \, \mathrm{d}x \text{ for all } i = 1, \ldots, m.$$

Considering the coefficients ξ_j as a vector $\xi = (\xi_1, \ldots, \xi_m)^T$, his term can be written as a matrix vector multiplication $M\xi$, where the entries of the matrix are defined by

$$(M)_{ij} = \int_\Omega \phi_j(x) \phi_i(x) \, \mathrm{d}x.$$

This matrix is called the mass matrix. In the one dimensional case the mass matrix is tridiagonal, since due to the construction of the basis functions as hat functions, the product $\phi_j \phi_i$ is zero whenever $|i-j| > 1$. In higher dimensions (of large size) it is a sparse matrix with non-zero diagonal.

2.2 Pressure segregation methods

In many relevant applications of the Navier-Stokes equations turbulent phenomena appear. Vortices occur in small scales, as in turbulent boundary layers close to objects, and on a range of scales, as in the turbulent wake behind obstacles. To resolve the full flow an enormous number of mesh points are required. The resulting algebraic systems cannot be solved on one single computer in reasonable time, instead the computations rely on large computer clusters.

When solving the Navier-Stokes equations on these supercomputers, one should resort to methods performing well on them. Large algebraic systems are commonly solved best with iterative methods, as these require less memory and are more easily parallelizable. Unfortunately, it is not easy to find well-scaling preconditioners for the monolithic problem [7].

The usual approach to avoid this problem is to split the monolithic system and solve it in several stages. In the 1960s, Chorin and Temam to this end developed projection methods [3, 30].

From an algebraic point of view these schemes correspond to preconditioned Richardson iterations of the Schur complement of the system. This shall be analyzed in more detail in Chapter 4.

2.2.1 Incremental projection method

Projection methods were first introduced by Chorin and Temam [3, 30]. The idea of these schemes is to first compute an intermediate velocity from the momentum equation without the pressure terms. As this velocity does not satisfy the divergence-free criterion, it is in a second step projected onto a divergence-free space. Hence the name, projection method.

There have been various modifications and improvements to the original formulation by Chorin and Temam. Shen proposed inserting the pressure in the first step and accounting for it in the second step [28]. These methods are called incremental projection methods. We are here taking a look at the incremental projection scheme, as stated in [7]: For each time step the following steps are performed.

- Momentum step: Find the intermediate velocity $\tilde{u}^{n+1}$ such that

$$\frac{\tilde{u}^{n+1} - u^n}{k} + (u^n \cdot \nabla)\tilde{u}^{n+1} - \nu\Delta\tilde{u}^{n+1} = f - \nabla p^n.$$

- Continuity/Projection step: Find the pressure p^{n+1} such that

$$k\Delta p^{n+1} = \nabla \cdot \tilde{u}^{n+1} + k\Delta p^n.$$

- Correction step: Update the velocity according to

$$u^{n+1} = \tilde{u}^{n+1} - k\nabla(p^{n+1} - p^n).$$

Note that from the correction step together with the condition $u^{n+1} \cdot n|_\Gamma = 0$ it follows that $\nabla(p^{n+1} - p^n) \cdot n|_\gamma = 0$, which implies that

$$\nabla p^{n+1} \cdot n|_\gamma = \nabla p^n \cdot n|_\gamma = \cdots = \nabla p^0 \cdot n|_\gamma.$$

Hence, the projection method enforces an artificial Neumann boundary condition on the pressure [19]. This is a widely disputed aspect of

projection methods. The Neumann boundary condition induces a numerical boundary layer, which limits the accuracy of the scheme. On the other hand, some argue that the size of the layer is so small that it is negligible [8].

2.2.2 Predictor corrector method

In the present work the predictor-corrector method, a modification of the incremental projection method, is used. The drawback of projection schemes is that they, due to the artificial Neumann boundary condition on the pressure, do not solve the same problem as the original monolithic system.

In order to reach convergence to the monolithic solution, the steps of the incremental projection scheme are repeated. For time step n, the following steps are iterated over i until convergence.

- Momentum step: Find the intermediate velocity $\tilde{u}^{n+1}$ such that

$$\frac{\tilde{u}^{n+1} - u^n}{k} + (u^{n+1,i} \cdot \nabla)\tilde{u}^{n+1} - \nu\Delta\tilde{u}^{n+1} = f - \nabla p^{n+1,i}.$$

- Continuity/Projection step: Find the pressure $p^{n+1,i+1}$ such that

$$k\Delta p^{n+1,i+1} = \nabla \cdot \tilde{u}^{n+1} + k\Delta p^{n+1,i}.$$

- Optional Correction step: Update the velocity according to

$$u^{n+1,i+1} = \tilde{u}^{n+1} - k\nabla(p^{n+1,i} - p^{n,i}).$$

Note that when convergence is reached, the first and second step become the momentum and continuity equation respectively. The converged correction step only yields $u^{n+1,i+1} = \tilde{u}^{n+1}$ and can therefore be omitted.

An alternative way to derive predictor corrector methods is by considering the discretized Navier-Stokes equations as an algebraic system. Then, a preconditioned Richardson iteration can be applied on the corresponding pressure Schur complement system. With an appropriate choice for the preconditioner one obtains a predictor corrector scheme. The incremental projection scheme is obtained by performing only one iteration per time step. A closer look at this approach will be taken in Section 4.

2.2.3 Implementation of a predictor corrector method

In the present project the predictor corrector method is used and analyzed. Formally it is obtained by adding the term

$$k\left(\nabla(p^i - p^{i-1}), \nabla q\right) \tag{2.11}$$

to the weak residual (2.7). This term stems from the Laplacian operators applied to the pressure in the continutity step. A further motivation is presented when we consider an algebraic approach to velocity pressure splitting in Section 4.

Decoupling the velocity and pressure equations from the adjusted weak residual and then solving the discrete system with a preconditioner Richardson iteration on the pressure Schur complement system leads to the predictor corrector scheme.

Decoupled equations

To segregate the equations, one component of the trial function $\hat{v} = (v, q)$ is varied in the weak residual (2.7) with the term (2.11), while the other term is set to zero. In this manner the decoupled system is derived as

$$r_m(\hat{u}^n, v) = \left((u^n - u^{n-1})k_n^{-1} + \bar{u}^n \cdot \nabla \bar{u}^n + \nabla p^n - f, v\right) + (\nu \nabla \bar{u}^n, \nabla v)$$
$$r_c(\hat{u}^n, q) = k\left(\nabla(p^i - p^{i-1}), \nabla q\right) + (\nabla \cdot \bar{u}^n, q)$$

The decoupled equations are solved iteratively until convergence. As a stopping criterion for this fixed point iteration we use the relative difference

$$\frac{\|u^i - u^{i-1}\|_{L_2(\Omega)}}{\|u^i\|_{L_2(\Omega)}} \leq tol,$$

where tol is a tolerance to be chosen, typically of the order 10^{-2}.

Newton's method

Each of the decoupled equations can be solved separately using Newton's method, which is here only shown for the momentum equation:

Start with an initial guess u_0^n and then iterate the following steps.

1. Compute the Jacobian of r_m, denoted $J = \frac{\partial}{\partial u_i^n} r_m(\hat{u}_i^n, v)$.
2. Solve the equation $Ju_{i+1}^n = Ju_i^n - r_m(\hat{u}_i^n, v)$ for u_{i+1}^n.

For the pressure equation Newton's method takes an equivalent form. The linear system in step 2 is solved with a GMRES method.

Time step restriction

Solving the decoupled equations iteratively until convergence equates to a fixed point iteration. To guarantee convergence of the method a CFL type restriction on the time step has to hold [20, 11], that is

$$k \leq C_{CFL} \frac{h}{|u|}$$

with a constant C_{CFL}. The velocity $|u|$ is often given by the inflow velocity. The test cases in this thesis are driven through gravitation only and have no inflow velocity. Therefore, we consider $|u|$ as a characteristic velocity of uniform size.

Adding the term (2.11) to the method allows the CFL constant to be chosen larger. Increasing the time step can decrease the computational cost substantially. It is therefore of great interest to find a well-working splitting scheme.

2.3 Arbitrary Lagrangian-Eulerian methods

In the context of simulating a floating object, one needs to model the interaction of the two fluids water and air and the interaction of the structure with the two fluids. Different approaches are applied for these two kinds of problems. In the following the Arbitrary Lagrangian-Eulerian (ALE) method is presented, which is used for the structure-fluid interaction. This Section is mainly based on [10]. Details on the implementation of ALE methods in Unicorn can be found in [16].

2.3.1 Lagrangian vs. Eulerian formulation

Classically, there are two approaches to describe motions in continuum mechanics, Lagrangian and Eulerian formulations. The central difference lies in the point of view the observer takes. A short introduction to both approaches is given here.

In a Lagrangian formulation each material particle is followed during the motion. For mesh based methods like Finite Element Methods this means that each node of the mesh describes one particle of the material. As the material is deformed, the mesh is thus deformed in the same way. This representation is intuitive in structure mechanics and widely used. However, for larger distortions of the material the mesh

becomes irregular very quickly and has to be remeshed frequently. It is therefore generally not common to apply the Lagrangian description to fluid flows.

Eulerian formulations on the other hand describe material properties such as velocity and pressure at a fixed point in space. In this case the computational mesh remains the same over the whole time of observation. This description is usually used in fluid dynamics, as it can easily handle large deformations of the continuum. However, due to the relative motion between the observed material and mesh, convective terms occur. These nonlinear terms introduce numerical difficulties and make it harder to follow deforming material interfaces. All notions in this work up to this point were based on Eulerian formulations.

2.3.2 ALE formulation

The ALE formulation combines the advantages of the classical Lagrangian and Eulerian formulations. Mesh points are neither held constant as in the Eulerian formulation, nor are they moved with the material as in the Lagrangian formulation. Instead a third configuration is used as a reference, such that the mesh points are moved in a way that can be arbitrarily specified. This freedom to specify the mesh movement allows to track the fluid structure interface explicitly while preventing the mesh from deforming in an irregular way as it easily happens in Lagrangian formulation. After some algebraic derivations, which can be followed in [10], the ALE FEM boils down to subtracting the mesh velocity β from various convective velocities that appear in the Finite Element formulation. That is, the convective term $u \cdot \nabla u$ is modified to $(u - \beta) \cdot \nabla u$ and the weak form (2.5) reads: Find $\hat{u} = (u, p) \in V$ such that

$$r(\hat{u}, \hat{v}) = (\dot{u} + (u - \beta) \cdot \nabla u + \nabla p - f, v) + (\nu \nabla u, \nabla v) + (\nabla \cdot u, q) = 0$$

for all $\hat{v} = (v, q) \in V$.

The mesh velocity β can be defined arbitrarily. It is straight forward to set the mesh velocity at the structure boundary equal to the velocity of the structure. With appropriate mesh regularization methods the movement for the other nodes is determined in such a way that the mesh cells are not distorted excessively. In the present project a Laplacian smoother is applied in which a Poisson equation is solved

to determine the mesh velocities such that nodes are moved in a regular way.

2.4 Two-phase flow

In the previous Section 2.3 the ALE method to simulate the fluid structure interaction was introduced. We will now present a method to describe the water air interaction. As the interface is moving much more than the fluid structure boundary, it is not feasible to track the interface explicitly as in the ALE method. Instead it is tracked implicitly with an interface capturing method. A common approach is to describe the boundary with the help of a level set function as first proposed by Sethian and Osher [26]. After introducing a model for two-phase flow an overview of level set methods is given. The following argumentation is mainly based on the book by Gross [25] and the articles by Sussman [23] and Jahn [22]. The latter is developing a level set toolbox in the FEniCS framework.

2.4.1 Model for two-phase flow

Recall the incompressible Navier-Stokes equations (2.1)–(2.2). A fluid is characterized by its density ρ and viscosity ν. The domain Ω shall now contain several fluids. Hence, different sets of material parameters (ρ, ν) need to be used in the Navier-Stokes equations, depending on which part of the domain Ω is modeled.

Let us assume the domain Ω contains two incompressible fluids that do not mix. It is therefore divided into two open subdomains $\Omega_1(t)$ and $\Omega_2(t)$. These regions depend on t, that is they may change over time.

At each time $t \in I$ the two subdomains

- fill the whole domain, $\overline{\Omega} = \overline{\Omega_1}(t) \cup \overline{\Omega_2}(t)$,
- and are disjoint, $\Omega_1(t) \cap \Omega_2(t) = \emptyset$,

The interface of the two phases is defined to be the separating boundary

$$\Gamma(t) = \overline{\Omega_1}(t) \cap \overline{\Omega_2}(t).$$

As each domain describes the territory of one of the fluids, the material properties ρ and ν depend on the location in both space and time. The parameters are denoted ρ_i and μ_i on $\Omega_i(t)$, $i = 1, 2$.

Hence, the full model can be written as

$$\begin{aligned} \rho_i(\dot{u} + u \cdot \nabla u) - \nu_i \Delta u + \nabla p = +\rho_i g + f_{surface} \quad & \text{on } \Omega_i,\ i = 1, 2 \\ \nabla \cdot u = 0 \quad & \text{on } \Omega, \end{aligned}$$

where g is the gravity constant and $f_{surface}$ is the surface tension, which is acting on the interface between the two fluids. It is usually modeled as by Sussman et al. [23] by the term

$$f_{surface} = \sigma \kappa \delta(d) n,$$

where σ denotes the surface tension coefficient, κ the curvature and d the distance to the interface and δ is the Dirac-delta.

2.4.2 An Overview of level set methods for two-phase flow

Level set methods were first introduced by Sethian and Osher in 1988 [26]. The basic idea is to represent the interface by the zero level set of a signed distance function. Thereby the position of the interface does not need to be determined, which is advantageous especially when it is moving relatively fast or the domains are changing topology. These methods that describe the interface implicitly are also called interface capturing methods as opposed to interface tracking methods, in which the interface is represented explicitly. An example of the latter is the ALE method described in Section 2.3, which moves the mesh as the interface is moving.

Level set function

As before, the space and time domain are denoted as $\Omega \in \mathbb{R}^3$ and $I = [0, t]$. Let us consider a continuous scalar function $\phi : \Omega \times I \to \mathbb{R}$. At each time $t \in I$, the zero level set of $\phi(t)$ is defined as

$$\Gamma(t) = \{x \in \Omega : \phi(x, t) = 0\}.$$

The function ϕ is called the level set function. It can be used to divide the domain Ω in two subdomains $\Omega^1(t)$ and $\Omega^2(t)$ with a separating

interface $\Gamma(t)$ by defining

$$x \in \begin{cases} \Omega^1(t), & \text{if } \phi(x,t) > 0 \\ \Omega^2(t), & \text{if } \phi(x,t) < 0 \\ \Gamma(t), & \text{if } \phi(x,t) = 0 \end{cases}$$

Moreover, these subdomains fulfill the conditions

$$\Omega = \Omega_1(t) \cup \Omega_2(t) \cup \Gamma(t) \text{ and } \Omega_1(t) \cap \Omega_2(t) \cap \Gamma(t) = \emptyset \text{ for all } t \in I.$$

That is, we have found a suitable description for the model described in the previous Section 2.4.1.

Typically, the level set function is defined as a signed distance function

$$\phi(x) = \begin{cases} \min\limits_{y \in \Gamma} ||x - y||, & x \in \Omega_1(t) \\ -\min\limits_{y \in \Gamma} ||x - y||, & x \in \Omega_2(t) \end{cases}$$

This yields the advantage that $||\nabla \phi|| = 1$ on the whole domain. Thereby, well-definedness of the normal and curvature of Γ are guaranteed. These can be computed by $n = \frac{\nabla \phi}{||\nabla \phi||}$ and $\kappa = -\nabla \cdot n$, respectively.

Transport equation for the level set function

Following [25], an advection equation for the level set function can be derived. In Langrangian coordinates, the movement of a particle $X(t)$ on the interface is described by $\frac{\mathrm{d}}{\mathrm{d}t} X(t) = u(x,t)$, where u is the velocity field. The level set function is constant on each particle path, $\phi(X(t),t) = const.$ for all $t \in I$. Hence, the material derivative of ϕ vanishes, that is

$$\phi_t + u \cdot \nabla \phi = 0. \tag{2.12}$$

We have found a transport equation for the level set function.

Reinitialization

Unfortunately, due to numerical issues, the level set function ϕ loses its signed-distance property over time, i.e. $||\nabla \phi|| \neq 1$. This distortion of the level set function introduces numerical errors. To restore the signed-distance property, reinitialization methods are applied after each time step. Two common approaches are presented in the following.

PDE approach Based on an approach by Ruoy and Tourin [5] who proposed an iterative method for reinitializing ϕ, Sussman et al. [23] introduced the following method. After each time step the current level set function ϕ_0 is used as an initial equation to solve the equation

$$\phi_\tau = \text{sign}(\phi_0)\,(1 - ||\nabla\phi||)$$

until ϕ reaches a steady state. Note that the time τ is merely a pseudo-time and has no connection to the physical time t. For numerical reasons, the sign function is replaced by a smooth representation

$$S_\epsilon(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + \epsilon^2}},$$

which approaches the sign function as $\epsilon \to 0$.

Fast Marching Method For the Fast Marching Method [27] the distance function is first corrected for the vertices of all elements lying on the interface. After this initialization phase the set of vertices for which the distances are already computed serves as an input to the extension phase. In this iterative second phase the distances of all vertices neighboring the set of computed distances are computed and then added to the set of already computed distances. For a more detailed description of the algorithm the reader is referred to [25] or [22].

2.5 The FEniCS project

The computational framework for the present project is the FEniCS environment [1]. The FEniCS project was started in 2003 as an umbrella for software components with the aim of automating the solution of mathematical models based on partial differential equations by means of the Finite Element Method.

The user specifies the weak form to a PDE based problem in Unified Form Language (UFL) and the code is generated automatically by the FEniCS component FFC. Solvers are written in the high-level Python and C++ interfaces to FEniCS.

3D simulations for the present master thesis are relying on the high performance computing branch of FEniCS (FEniCS-HPC). An introduction to the FEniCS-HPC components can be found in [13].

Chapter 3

The variable density model

In the present work an approach to multiphase flow is described that is a level set method in the wider sense, but does not rely on signed distance functions. Instead the transport equation for the level set method is directly integrated in the system in the form of a density equation. The model described in this chapter is implemented in the FEniCS framework and was first presented in [17] in the context of marine engineering applications.

After formulating the stabilized Finite Element Method for the variable density model, the test cases for the numerical experiments in Chapter 5 are presented.

3.1 The variable density Navier-Stokes equations

As in Chapter 2, the computational domain is denoted by $\Omega \times I$. The incompressible Navier-Stokes equations are extended by a variable density ρ, such that they are now solved for the set (u, ρ, p).

The transport equation (2.12) for the level set function is directly integrated in the finite element method. The variable density incompressible Navier-Stokes equations can then be written in residual form as

$$R(\hat{u}) = \begin{pmatrix} \rho(\dot{u} + u \cdot \nabla u) + \nabla p - \mu \Delta u - \rho g \\ \dot{u} + (u \cdot \nabla)\rho \\ \nabla \cdot u \end{pmatrix} = 0, \tag{3.1}$$

where $\hat{u} = (u, \rho, p)$.

The total residual is expressed as a vector $R = (R_m, R_d, R_c)^T$, where the residual components are the strong residuals of momentum, density and continuity equation, respectively. The residual of the density equation, R_d takes the same form as the transport equation (2.12) derived for the level set function in the standard method. Hence, it describes the motion of the density interface. The momentum and continuity equation residuals are the same as in the standard model.

Note that although we have introduced a variable density ρ, the flow is still considered incompressible. Therefore the density does not vary along streamlines and the continuity equation remains the same as for the constant density incompressible Navier-Stokes equations. This approach is motivated in [11].

In the high Reynolds number applications we target in this thesis the viscosity is small, often approximated zero, and therefore we let the viscosity be constant over the full domain for simplicity.

3.1.1 Finite Element formulation for the variable density model

Multiplying the strong residual (3.1) by a test function $\hat{v} = (v, \eta, q)$ and integrating over the domain gives the weak formulation of the problem. The weak form is then discretized in space and time as in Section 2.1.3 with an additional function space W^n for the density. Finally, the Finite Element Method reads:

For $n = 1, \ldots, N$, find $\hat{u}^n \in [W_0^n]^3 \times W^n \times W^n$ such that

$$r(\hat{u}^n, \hat{v}^n) = 0$$

for all $\hat{v} \in [W_0^n]^3 \times W^n \times W^n$. The weak residual reads

$$\begin{aligned} r(\hat{u}^n, \hat{v}^n) = &\big((\rho u^n - u^{n-1})k_n^{-1} + \rho\bar{u}^n \cdot \nabla\bar{u}^n + \nabla p^n - \bar{\rho}g, v\big) + (\mu\nabla\bar{u}^n, \nabla v) \\ &+ \big((\rho^n - \rho^{n-1})k_n^{-1}, \eta\big) + (\bar{u} \cdot \nabla\bar{\rho}, \eta) \\ &+ (\nabla \cdot \bar{u}^n, q), \end{aligned} \tag{3.2}$$

As before the bar notation denotes $\bar{u} = \theta u^n + (1-\theta)u^{n-1}$. In the present work $\theta = 1/2$, which corresponds to a Crank-Nicholson scheme, is used.

The presented model does not include any explicit reinitialization methods as the ones presented for the standard level set method in

Section 2.4.2. These would have to be executed after every time step and thus increase the computational cost substantially. Instead a number of terms are inserted in the method to sustain the immiscibility property of the flow and stabilize the computations. These terms are introduced in the following.

Phase separation term

To reduce the mixing of the fluids, a phase separation term is added. It takes the form

$$-c_{sep}(\rho_1 - \bar{\rho})(\rho_2 - \bar{\rho})(\mathbf{1}, \nabla\eta),$$

where $\mathbf{1} = (1, 1, 1)$ and the separation coefficient is defined as

$$c_{sep} = \frac{0.2}{0.1\|\nabla\bar{\rho}\|_1 + \bar{\rho}_0}.$$

The norm of the gradient $\|\nabla\bar{\rho}\|_1$ is small far away from the interface and becomes large only on the interface. The separation coefficient c_{sep} becomes large when $\|\nabla\bar{\rho}\|_1$ is small and will in this case force the term $(\rho_1 - \bar{\rho})(\rho_2 - \bar{\rho})(\mathbf{1}, \nabla\eta)$ to become small. That is, $\bar{\rho} = \frac{1}{2}(\rho^n + \rho^{n-1})$ must take a value close to either ρ_1 or ρ_2. Assuming $\rho^{n-1} \approx \rho_1$ it results that $\rho^n \approx \rho_1$ as well. Through the phase separation term the diffusion of the interface is thus prevented. Hence, the phases are driven away from mixing. The term has therefore a similar function as the reinitialization needed in the standard level set methods presented in Section 2.4.2.

Stabilization terms

A weigthed least squares stabilization is applied by adding the term

$$LS = (d\bar{R}, R_v) + (d\bar{R}, R_\eta) + (d\bar{R}, R_q),$$

where d is a stabilization parameter of size $h^{3/2}$ and the strong residuals are defined as

$$\bar{R} = \begin{pmatrix} \bar{\rho}(\bar{u}\cdot\nabla)\bar{u} + \nabla p - \bar{\rho}g \\ \bar{u}\cdot\nabla\bar{\rho} - \mathbf{1}\cdot(c_{sep}(\rho_1 - \bar{\rho})\bar{\rho}) \\ \nabla\cdot\bar{u} \end{pmatrix}, \quad \bar{R}_v = \begin{pmatrix} \bar{\rho}(\bar{u}\cdot\nabla)\bar{v} \\ \bar{v}\cdot\nabla\bar{\rho} \\ \nabla\cdot\bar{v} \end{pmatrix}$$

$$\bar{R}_\rho = \begin{pmatrix} \eta(\bar{u}\cdot\nabla)\bar{u} \\ \bar{u}\cdot\nabla\eta - \mathbf{1}\cdot\nabla(c_{sep}(\rho_1-\bar{\rho})\eta) \\ 0 \end{pmatrix}, \quad \bar{R}_q = \begin{pmatrix} \nabla q \\ 0 \\ 0 \end{pmatrix}$$

This term is a natural extension to the stabilization introduced in (2.8). When the density is set to 1 and the second dimension corresponding to the density equation is omitted we receive the same stabilization term as in the uniform density case.

Shock-capturing terms

In addition shock capturing terms are added to the momentum and density equation residuals. These take the form

$$SC_m = \|u^n\|_1 \left(c_1 h^2 R_{SC} + c_2 h^{3/2}\right)(\nabla\bar{u}, \nabla v)$$

for the momentum equation and

$$SC_\rho = \|u^n\|_1 \left(c_1 h^2 R_{SC} + c_2 h^{3/2}\right)(\nabla\bar{\rho}, \nabla\eta)$$

for the density equation, with

$$R_{SC} = R_d(\bar{u}, \bar{\rho}) + \|R_m(\bar{u}, \bar{\rho}, p)\|_1$$

and the constants c_1 and c_2 of unit size.

The shock capturing terms induce numerical dissipation to the velocity and density. The extent of dissipation added depends on the norm of the strong density and momentum residuals and on the norm of the velocity. The larger these terms become the more dissipation is introduced. Through this mechanism the density and velocity solution are smoothed near the presence of shocks. This prevents spurious oscillations and instabilities in the solution.

Schur complement term

The equations shall be solved in a decoupled way as described in Section 2.2. Therefore a Schur complement term similar to (2.11) is added to the model. The term takes the form

$$\alpha\left(\nabla(p^i - p^{i-1}), \nabla q\right), \tag{3.3}$$

where the index i denotes the fixed point iteration index. This term corresponds to a Schur complement preconditioner with the preconditioning factor α. In the case of uniform and constant density this

value is chosen as $\alpha = k$ (cf. equation (2.11)). An appropriate value for α in the variable density model is yet to be determined and the object of this work. In Chapter 4 a preconditioner is suggested and its performance is studied in Chapter 5.

3.1.2 A parameter free turbulence model

In this work only high Reynolds number flows are considered. As the size of the boundary layer becomes exceedingly small for high Reynolds numbers, it is not computationally feasible to resolve vortices in the boundary layer. However, it has been observed that for large Reynolds numbers, the skin friction at boundaries becomes negligible [12]. Thus, flows with high Reynolds numbers can be described by an inviscid model with slip boundary conditions [14]. Turbulent dissipation is then introduced to the FEM through its stabilization method.

An inviscid model has another advantage for the variable density model. In Section 2.4.1 we established that each fluid is described by its density ρ and viscosity ν. The proposed variable density model, however, only views the density as a varying parameter. The viscosity is a constant over the whole domain. Using an inviscid model thus bypasses this problem.[1]

Note that standard level set methods also require the implementation of a surface tension. In a high Reynolds number flow this surface tension can be neglected.

3.2 Test cases

To assess the performance of the Schur preconditioner suggested in Chapter 4, tests are run with a 2D Python code on a local machine and with the 3D C++ code `unicorn_var_den_ale` [4] on the computing cluster Beskow at the PDC Center for High Performance Computing at

[1] Alternatively, an approach to define the variable viscosity described in [17] is to make it depend on the density. For two fluids with parameters (ρ_1, ν_1) and (ρ_2, ν_2) the viscosity can be prescribed as

$$\nu = \nu_1 + \frac{\rho - \rho_1}{\rho_2 - \rho_1} (\nu_2 - \nu_1) .$$

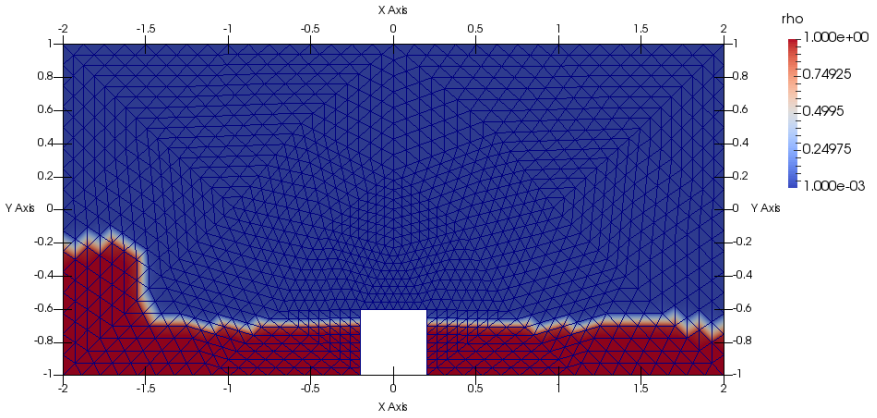


Figure 3.1: The initial condition of the density on the 2D test case on the coarse mesh with $h_{min} = 0.52$.

the KTH Royal Institute of Technology. The tested cases are presented in the following.

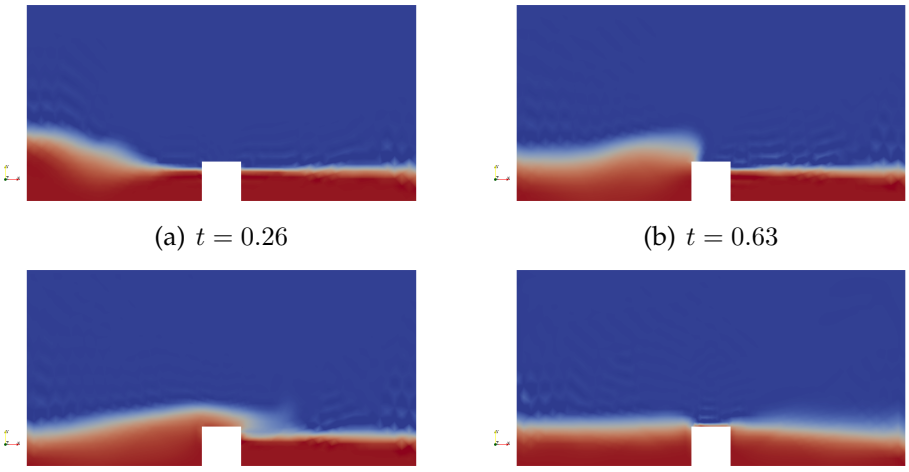
3.2.1 2D test case

The two dimensional test case simulates a water column hitting an obstacle mounted to the floor. The domain Ω is a rectangle of size 4×2 with a cube of side length 0.4 mounted on the floor in the centre. Simulations are run on two different tetrahedral meshes, the coarse one with mesh size $h = 0.52$ and the finer one with mesh size $h = 0.26$. The minimal density ρ_{min} is varied with respect to a maximal density of $\rho_{max} = 1$. In the standard case it is $\rho_{min} = 0.001$, the density of air with respect to water. Some tests are run with a larger minimal density $\rho_{min} = 0.01$ as this case speeds up the simulations. The initial condition of the density is presented in Figure 3.1.

The system is driven only through the gravitational force acting on the fluids. Slip boundary conditions are imposed for the velocity. On the upper boundary the density is set to ρ_{min} and the pressure to zero. The resulting interaction of water and air can be seen in Figure 3.2.

3.2.2 3D test case

For the 3D test case the FEniCS-HPC code `unicorn_var_den_ale` [4] is used, which simulates a floating cube. The densities of water, air and the cube are defined as $\rho_{water} = 1.0$, $\rho_{air} = 0.001$ and $\rho_{cube} =$



(a) $t = 0.26$ (b) $t = 0.63$

(c) $t = 1.05$ (d) $t = 2.63$

Figure 3.2: Simulation results of 2D test case on coarse mesh with $\rho_{min} = 0.001$.

0.5 respectively. The initial domain Ω is a large cube of side length 8 with a small cube of side length 0.5 in the center of the domain. The problem is again only driven through the gravitational force and fluid structure interaction. Boundary conditions are applied equivalent to the 2D case. For the initial condition the cube is half filled with water and half with air, as it is depicted in Figure 3.3. A cut through the z-plane is shown in Figure 3.4.

ALE method

In contrast to the 2D test case the position of the object is now changing over time. Hence the mesh is distorted with an ALE method as described in Section 2.3. The velocity of the cube's movement is restricted to the vertical direction. It is determined through Newton's second law of motion combined with an Euler forward method. Since only movement in the vertical direction is considered, the total force acting on the cube is the sum of the lift force L and the gravitational force $m_c g$, where m_c is the mass of the cube and g the gravitational constant. From Newton's second law we know that the force on the cube can also be expressed as $F = m_c \cdot \mathrm{d}v_c/\mathrm{d}t$. Applying Euler's method to $\mathrm{d}v_c/\mathrm{d}t = F/m_c$ gives the scheme

$$v_c^{n+1} = v_c^n + \frac{k}{m_c}\left(L + m_c g\right).$$

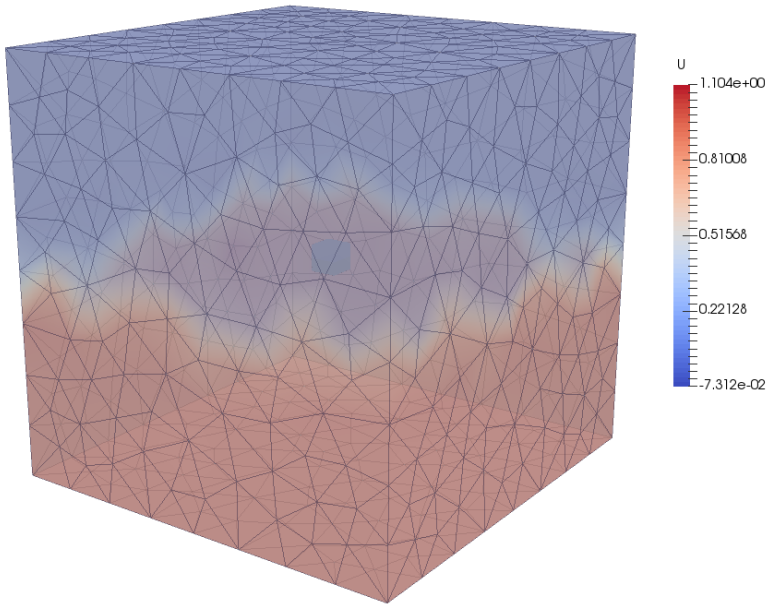


Figure 3.3: The initial condition of the density in the 3D test case.

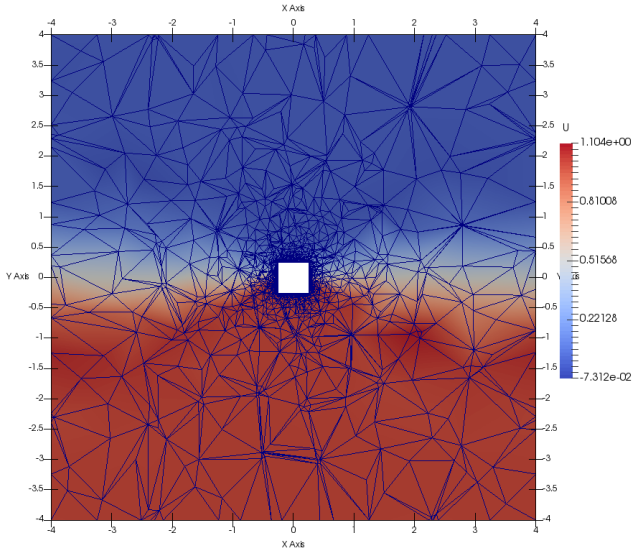


Figure 3.4: Slice through the 3D test case domain, depicting the initial condition of the density.

The cubes velocity defines the mesh velocity at the object boundary and a Laplacian smoother is applied to dispense the mesh distortion over the domain.

Chapter 4

A pressure segregation method for the variable density Navier-Stokes equations

In Section 2.2 the need for high performance computing to solve the Navier-Stokes equations is brought forward. To this end pressure segregation methods are introduced and two schemes, the incremental projection and predictor corrector method, are presented. It is also mentioned that it is constructive to study these methods on an algebraic level. This study is the objective for the present chapter.

A list of benefits that the algebraic approach to pressure segregation methods brings over the continuous point of view has been presented by Badia and Codina [24]. In the present project we take advantage of the fact that effective preconditioners can be derived for pressure segregation methods from an algebraic viewpoint.

To give an example, the predictor corrector method can be interpreted as a preconditioned Richardson iteration on the pressure Schur complement system corresponding to the Navier-Stokes equations. By the means of an algebraic approach a pressure corrector method for the non uniform density Navier-Stokes equations is thus motivated in Section 4.1.

With some additional considerations, this preconditioner is extended to the variable density model in Section 4.2. With the help of some heuristics we suggest a preconditioner for the variable density case. The proposed preconditioner coincides with a preconditioner that has already been put forward by Guermond [18] for an incremental pro-

jection scheme by the means of a quite different approach.

4.1 Algebraic approach to predictor corrector methods

To simplify an extension of the theory developed in this section to the variable density Navier-Stokes equations, we consider now a fluid with constant density that is not necessarily set to 1. That is, the Navier-Stokes equations (2.1)–(2.2) are augmented by a density parameter ρ and take the form

$$\begin{aligned} \rho\dot{u} + \rho u \cdot \nabla u - \mu\Delta u + \nabla p &= f \\ \nabla \cdot u &= 0, \end{aligned}$$

In the present section, pressure segregation methods for this system of equations are introduced form an algebraic point of view. An extensive review of these algebraic methods can be found in the books by Turek [29] and Elman [9]. In the following a preconditioned Richardson iteration for a pressure Schur complement system, that correspond to a predictor corrector method is developed, similar to the works by Houzeaux et al. [7, 6].

4.1.1 The discretized system

Let us first take a look at the discretized system given through the FEM for the Navier-Stokes equations. For simplicity the stabilization is neglected for now.

In Tureks book [29] the Navier-Stokes equations are time-discretized by a slightly different θ scheme than the one used in this work. As the problem discretized in Tureks way is convenient to demonstrate the form of the discretized system we shall nonetheless state it here:

Given (u^{n-1}, p^{n-1}) and the time step $k = t^n - t^{n-1}$, find the velocity and pressure (u^n, p^n) such that

$$\rho\frac{u^n - u^{n-1}}{k} + \theta\left(\rho u^n \cdot \nabla u^n - \mu\Delta u^n\right) + \nabla p^n = g^n \tag{4.1}$$

$$\nabla \cdot u = 0 \tag{4.2}$$

with the right hand side

$$g^n = \theta f^n + (1-\theta) f^{n-1} - (1-\theta)\left(\rho u^{n-1} \cdot \nabla u^{n-1} - \mu\Delta u^{n-1}\right). \tag{4.3}$$

It should be kept in mind that this is not the exact same time discretization as used in this project.[1]

After discretizing the system in space according to a finite element approach as described in Section 2.1.3, the space-discretized version of the problem can be expressed as a nonlinear algebraic system that reads as follows. Given (u^{n-1}, p^{n-1}) and the time step $k = t^n - t^{n-1}$, find the velocity and pressure (u^n, p^n) such that

$$A_\theta u^n + Bp^n = g, \tag{4.4}$$

$$-B^T u^n = 0, \tag{4.5}$$

with the right hand side

$$g = A_{\theta-1} u^{n-1} + \theta f^n - (1-\theta) f^{n-1}.$$

The matrix A_θ is defined as $A_\theta = \rho k^{-1} M + \rho N_\theta - \mu L_\theta$, where M is the mass matrix from the example in Section 2.1.4 and N_θ and L_θ correspond to the time discretized nonlinear convective term and the discrete Laplacian respectively.

The matrix B corresponds to a gradient operator ∇ and $-B^T$ to the divergence operator $\nabla\cdot$.

It can be convenient to express matrix A_θ in terms of an operator as Houzeaux et al. do in [7]. Note that A_θ corresponds to the operators in the momentum equation that are acting on the velocity. Using Tureks time splitting scheme, this operator can be denoted as

$$\mathcal{M} = \frac{\rho}{k} + \theta \left(\rho u^i \cdot \nabla - \mu \Delta \right). \tag{4.6}$$

4.1.2 Derivation of the Schur system

In the following the time indices are dropped to simplify the notation. Further, to include stabilization and other terms in the method, we generalize the system (4.4)-(4.5) to

$$\begin{pmatrix} A & B \\ -\bar{B}^T & C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} \tag{4.7}$$

[1] In fact the method in this project only differs from Tureks formulation in the nonlinear terms. Recall from equation (2.7) that the time discretization applied to the convective term takes the form $\bar{u}^n \cdot \nabla \bar{u}^n$ with $\bar{u}^n = \theta u^n + (1-\theta) u^{n-1}$ with $\theta \in [0, 1]$. Hence, using the approximation

$$\bar{u}^n \cdot \nabla \bar{u} \approx \theta u^n \cdot \nabla u^n + (1-\theta)\, u^{n-1} \cdot \nabla u^{n-1}$$

our method would lead to Tureks formulation (4.1)-(4.3).

as it is done in [7, 6]. This allows small contributions due to stabilization or other terms in the FEM formulation to be added. Nonetheless, the predominant contributions to A, B and $-\bar{B}^T$ still stem from the momentum operator $\mathcal{M}$, the gradient operator ∇ and the divergence operator $\nabla\cdot$ respectively.

Assuming that A is invertible the first equation can be rearranged to express the velocity

$$u = A^{-1}\left(g_1 - Bp\right). \tag{4.8}$$

This term is plugged in the second equation to obtain the Schur complement system

$$Sp = b, \tag{4.9}$$

where the Schur complement matrix and right hand side are defined as

$$S = C + \bar{B}^T A^{-1} B, \tag{4.10}$$

$$b = g_2 + \bar{B}^T A^{-1} g_1. \tag{4.11}$$

The problem is now reduced to a scalar problem in p only. Once the pressure is computed, the velocity can be obtained through relation (4.8). However, solving the Schur system (4.9) is still computationally expensive. The Schur complement matrix S is a full matrix which is neither trivial to construct nor to invert. More numerical techniques need to be exploited to solve the Schur system in a cheap way.

4.1.3 Preconditioned Richardson iteration

It is suggested in the literature to solve the Schur complement system (4.9) iteratively with a preconditioned Richardson iteration [2]. That is, at iteration $k+1$ the new pressure solution p^{k+1} is computed from p^k by

$$p^{k+1} = p^k + Q^{-1} r_k,$$

where Q^{-1} denotes the preconditioner and $r_k = b - Sp^k$ the residual of the Schur system at iteration k.

Constructing the residual r_k requires the exact computation of the Schur complement S and thereby the inversion of matrix A, which can become very costly. To avoid this explicit computation, note that with the relations (4.8), (4.10) and (4.11) it follows that

$$Qp^k + r_k = Qp^k + b - Sp^k = \left(Q - C\right) p^k + g_2 + \bar{B}^T u^{k+1}$$

Hence, instead of solving the system

$$\begin{aligned} Au^{k+1} &= g_1 - Bp^k \\ p^{k+1} &= p^k + Q^{-1} r_k \end{aligned}$$

one can instead solve

$$Au^{k+1} = g_1 - Bp^k \tag{4.12}$$
$$Qp^{k+1} = g_2 + \bar{B}^T u^{k+1} + (Q - C)\, p^k. \tag{4.13}$$

Thus, the Schur complement matrix S does not need to be constructed explicitly.

At this point we can verify that the solution to this system converges to the solution of the monolithic scheme. For the converged solution it holds $u^k \approx u^{k+1}$ and $p^k \approx p^{k+1}$. Plugging this in the system it only remains

$$\begin{aligned} Au^k + Bp^k &= g_1 \\ -\bar{B}^T u^k + Cp^k &= g_2, \end{aligned}$$

the monolithic system(4.7). Hence, the solution of the Schur system solved with a Richardson iteration is the same as of the monolithic scheme.

4.1.4 A preconditioner leading to a predictor corrector scheme

We will now see that a certain choice of a preconditioner leads to a predictor corrector scheme. The preconditioner proposed in the literature [7, 6, 9] is an approximation of the inverse Schur complement matrix S. To motivate this choice of a preconditioner consider the sequence of errors $e^k := p^k - p^*$, where p^* denotes the exact solution [2]. For the error it holds that

$$e^{k+1} = p^{k+1} - p^* = \left(I - Q^{-1}S\right) e^k.$$

Hence, the error converges if $\|I - Q^{-1}S\| < 1$ and it converges faster, the better $Q^{-1}S$ approximates the identity matrix in a certain sense. This motivates the choice of $Q \approx S^{-1}$ as a preconditioner.

As in the system (4.12)-(4.13) not the preconditioner Q^{-1} but its inverse Q appears, we refer to both Q^{-1} and Q as preconditioner.

The preconditioner Q can be split up in

$$Q = C + P,$$

where

$$P \approx B^T A^{-1} B.$$

As stated before the construction of the exact product $B^T A^{-1} B$ is computationally expensive. Again we need to find a way to avoid its explicit assembly.

Preconditioner for system without stabilization

Let us first again consider a system of the form (4.4)-(4.5). That is, we ignore additional terms such as stabilization so that the matrix C and vector g_2 vanish.

The Schur system (4.9) for this case simplifies to

$$Sp = b,$$

with

$$S = B^T A^{-1} B,$$
$$b = B^T A^{-1} g.$$

An appropriate preconditioner for the Schur complement matrix S according to Turek [29] and sources therein, is constructed as

$$P = B^T \tilde{A}^{-1} B,$$

where $\tilde{A}$ is typically chosen as a diagonal matrix, for example $\tilde{A} = \text{diag}(A)$.

Recall from Section 4.1.1 that matrix A can be expressed as a sum of reactive, convective and diffusive parts. Dropping the θ index the sum is written as

$$A = \frac{\rho}{k} M + \rho N - \mu L$$

Turek describes an additive approach to derive a global preconditioner P. By finding preconditioners for the reactive, convective and diffusive parts in A, the final operator can be expressed as a sum of these preconditioners. That is the global Schur system preconditioner is approximated as

$$\left(B^T A^{-1} B\right)^{-1} \approx k^{-1} \left(B^T M^{-1} B\right)^{-1} + \left(B^T N^{-1} B\right)^{-1} - \mu \left(B^T L^{-1} B\right)^{-1}.$$

We note that for unsteady flows and sufficiently small time steps k, the reactive part of the matrix A dominates the convective and diffusive parts. That is,

$$A = \frac{\rho}{k} M + \rho N + \nu L \;\approx\; \frac{\rho}{k} M, \;\text{ as } k \text{ becomes small.}$$

Hence, the preconditioner $P^{-1} = \rho k^{-1} \left(B^T M_l^{-1} B\right)^{-1}$ is suggested. The matrix M_l is a lumped version of M, for example $M_l = \text{diag}(M)$. This choice of a Schur complement matrix behaves like a discrete Poisson operator. As M_l is a diagonal matrix it is commutative, thus

$$P = \frac{k}{\rho} M_l B^T B = \frac{k}{\rho} M_l L, \tag{4.14}$$

where L denotes the discrete Laplacian as before.

Houzeaux et al. [7] derive the same preconditioner from an operator point of view. They consider the operator $\mathcal{M}$ defined in (4.6) which corresponds to matrix A. Hence, the matrix product $P = B^T A^{-1} B$ can be computed from the weak form of Uzawa's operator $-\nabla \cdot \mathcal{M}^{-1} \nabla$. For sufficiently small time steps it is proposed to approximate the momentum operator element wise by its first term $\mathcal{M} \approx \frac{\rho}{k}$. Integrating by parts and omitting the boundary terms gives the following expression to compute P

$$-\int_\Omega \left(\nabla \cdot \mathcal{M}^{-1} \nabla p\right) q \,\mathrm{d}\Omega \;\approx\; \int_\Omega \mathcal{M}^{-1} \nabla p \cdot \nabla q \,\mathrm{d}\Omega \;\approx\; \int_\Omega \frac{k}{\rho} \nabla p \cdot \nabla q \,\mathrm{d}\Omega.$$

Hence, the preconditioner P in operator form is $\mathcal{P} = -\frac{k}{\rho}\Delta$, which is also the operator corresponding to the algebraically derived preconditioner (4.14).

Preconditioner for system with stabilization

Let us now come back to the generalized system (4.7) with stabilization terms. As mentioned before the proposed preconditioner is $Q = (C + P)$, which approximates the Schur complement matrix S. Neglecting small perturbations in B and $-\bar{B}^T$ these matrices can still be understood as discrete counterparts to the gradient and divergence

operator. The approximation found for P in the previous section can then be directly applied for the generalized case. The full scheme (4.12)-(4.13) reads then

$$\begin{aligned} Au^{k+1} &= g_1 - Bp^k \\ (C+P)\,p^{k+1} &= g_2 + \bar{B}^T u^{k+1} + Pp^k. \end{aligned}$$

That is, the stabilization C is only applied to the updated pressure p^{k+1} but not to the previous pressure p^k. One can therefore regard the matrix C as part of the scheme and only P as the preconditioner. This point of view is followed in the numerical analysis of the preconditioner in Chapter 5.

4.1.5 Predictor corrector scheme

It is left to show that the Richardson iteration on the Schur complement system with the suggested preconditioner leads to a predictor corrector method as introduced in Section 2.2.2. We only consider the non stabilized version, that is the algebraic scheme reads

$$\begin{aligned} Au^{k+1} &= g - Bp^k \\ Pp^{k+1} &= B^T u^{k+1} + Pp^k, \end{aligned}$$

where A corresponds to the momentum operator $\mathcal{M}$, B to the gradient operator and $-B^T$ to the divergence operator. The matrix P corresponds to the Laplacian operator multiplied by a factor $-k/\rho$. Hence, the scheme in operator form reads

$$\begin{aligned} &\rho\frac{u^{n+1}-u^n}{k} + \rho(u^n\cdot\nabla)u^{n+1} - \mu\Delta u^{n+1} = f - \nabla p^n \\ &\frac{k}{\rho}\Delta p^{n+1} = \nabla\cdot u^{n+1} + \frac{k}{\rho}\Delta p^n. \end{aligned}$$

With the density set to 1 this scheme is identical to the predictor corrector scheme for uniform density presented in Section 2.2.2. Moreover, we found a natural extension for the predictor corrector method with nonuniform density, namely to replace the preconditioning factor $\alpha = k$ by $\alpha = k/\rho$. In fact, the incremental projection method can be modified in the same way for the non uniform density model. We can interpret this scheme from an algebraic viewpoint as a Schur complement method that applies only one Richardson iteration per time step.

4.2 Extension of the preconditioner to the variable density model

The derivations in Section 4.1 are based on the fact that the density ρ is constant. Let us now consider the case of variable density. The system is therefore augmented by an additional parameter ρ. Similar to (4.7) the variable density system shall be expressed as a matrix multiplication. From the variable density FEM (3.2) without any stabilization, phase separation or shock capturing terms the following equivalent to (4.7) is constructed.

$$\begin{pmatrix} A & 0 & B \\ 0 & D & 0 \\ -B^T & 0 & 0 \end{pmatrix} \begin{pmatrix} u \\ \rho \\ p \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ 0 \end{pmatrix} \tag{4.15}$$

The matrices A and B are defined as before, that is A corresponds to the momentum operator $\mathcal{M}$ and B and $-B^T$ correspond to the gradient and divergence operator respectively. The matrix D is associated with the transport equation for the density. Its operator form is

$$\mathcal{D} = \frac{1}{k} + u^i \cdot \nabla.$$

In the system (4.15) one can see that neither the momentum equation nor the pressure equation are acting on the density ρ. Moreover, the density equation is acting neither on the velocity u nor on the pressure p. Although the density is part of the momentum operator $\mathcal{M}$ and the velocity appears in the transport operator $\mathcal{D}$, there is a certain decoupling between the density equation and the momentum and pressure equation. This leads to the assumption that a pressure Schur complement similar to the constant density case can be applied. However, since the density is variable the definition of a suitable preconditioner is not straightforward.

We presume that the preconditioner P is still associated with a discrete Laplacian. That is, it corresponds to the operator

$$\mathcal{P} = -\alpha\Delta.$$

In the following we shall refer to only the constant α as preconditioner. Recall that in the constant density case we derived the preconditioner

$\alpha = k/\rho$. For a two-phase flow this approach would suggest to use two different preconditioners for the two different phases, such that

$$\alpha = \begin{cases} k/\rho_{max}, & \text{if } \rho \approx \rho_{max} \\ k/\rho_{min}, & \text{if } \rho \approx \rho_{min} \end{cases} \tag{4.16}$$

However, the interface between the two phases is not tracked explicitly. Defining α in this way is thus not possible. Hence, we decide to use the more conservative choice for α. The preconditioner k/ρ_{max} is by a factor 10^3 smaller than k/ρ_{min}. One can therefore not expect it to perform well in the domains where $\rho \approx \rho_{min}$. On the other hand, using k/ρ_{min} in the domains of $\rho \approx \rho_{max}$ is not expected to imply any difficulties. The preconditioner we suggest is therefore

$$\alpha = \frac{k}{\rho_{min}}.$$

The same preconditioner has already been proposed by Guermond [18] for an incremental projection scheme. Guermonds argumentation follows a quite different approach from our algebraic considerations. The term $k\rho_{min}^{-1}\Delta p$ is not regarded as a pressure Schur complement term, but as a perturbation to the Navier-Stokes equations that acts as a penalty on the divergence of velocity. The term

$$\frac{k}{\rho_{min}} (\nabla p, \nabla q)$$

in the weak residual results from both approaches.

Chapter 5

Results of numerical experimentation

In Chapter 4 a predictor corrector method is proposed for the variable density incompressible Navier-Stokes equations. Formally, the method is implemented by adding the term

$$\alpha \left(\nabla p, \nabla q \right)$$

to the weak residual. The suggested preconditioning factor is

$$\alpha = \frac{k}{\rho_{min}}.$$

In the following we refer to α as the preconditioner. Its performance is investigated in this chapter.

In the constant density model the Schur preconditioning term allows for a larger time step size k. In Section 5.1 it is shown that the Schur preconditioning term has the same effect on the variable density model.

In Section 5.2 the convergence of the fixed point iteration is studied for the method without and with preconditioning terms. The preconditioner α is varied to examine the effect it has on the convergence.

One way to measure the performance of the preconditioner is by counting the total amount of required computations for the method. In Section 5.3 we quantify this through the accumulated number of fixed point iterations required for simulations up to a certain time. The influence of different time step sizes k and preconditioners α is investigated.

In Section 5.4 the accumulated number of fixed point iterations is studied for the 3D test case. Quantifying the performance through this feature ignores the internal iterations of Newton's method and the linear solver in each fixed point iteration loop. Therefore, we also study the total computation time spent in the fixed point iteration, which is the feature the user ultimately aims to minimize.

5.1 Maximal time step size k

As pointed out in Section 2.2, the step size in the method for the constant density incompressible Navier-Stokes equations is restricted by a CFL type condition

$$k \leq C_{CFL} \frac{h}{|U|}. \tag{5.1}$$

Introducing a preconditioner to the fixed point iteration allows for a larger constant C_{CFL}. The quality of the preconditioner can thus be characterized by the largest possible step size k_{max}.

In a few first tests with the variable density model, the size of k_{max} and therefore C_{CFL} is determined experimentally for the method with and without preconditioner. This is done for the two dimensional test case on both the coarse and fine mesh. The time step size k is kept constant for the whole simulation. Table 5.1 and Table 5.2 show the results for the case where the minimal density is $\rho_{min} = 0.001$ and $\rho_{min} = 0.01$ respectively.

As it is known from the constant density case, we can confirm for the variable density model that it is possible to increase the CFL constant significantly when applying a Schur preconditioner. The time step can be increased by a factor of around 50. On the fine mesh with lower minimal density the factor is slightly lower and on the coarse mesh a little larger.

Further, comparing the values in Table 5.1 and 5.2 it is visible that when the minimal density ρ_{min} is chosen a factor 10 larger, the CFL constant can be increased by the same factor. This becomes especially apparent when comparing the values without preconditioner terms, but can also be seen from the result with preconditioner.

In the method without preconditioning term, the CFL constant does not appear to be influenced by the mesh size. When preconditioning terms are applied, however, the CFL number seems to be smaller on

	coarse mesh, h=0.52		**fine mesh, h=0.26**	
	C_{CFL}	k_{max}	C_{CFL}	k_{max}
without preconditioning	0.0004	$2 \cdot 10^{-5}$	0.0004	$1 \cdot 10^{-5}$
with preconditioning	0.025	$131 \cdot 10^{-5}$	0.015	$39 \cdot 10^{-5}$
factor with/without preconditioning	62.5		37.5	

Table 5.1: Maximal step size k_{max} with minimal density $\rho_{min} = 0.001$, with and without Schur preconditioner $\alpha = k/\rho_{min}$ applied.

	coarse mesh, h=0.52		**fine mesh, h=0.26**	
	C_{CFL}	k_{max}	C_{CFL}	k_{max}
without preconditioning	0.004	$2 \cdot 10^{-4}$	0.004	$1 \cdot 10^{-4}$
with preconditioning	0.25	$131 \cdot 10^{-4}$	0.2	$53 \cdot 10^{-4}$
factor with/without preconditioning	62.5		50	

Table 5.2: Maximal step size k_{max} with minimal density $\rho_{min} = 0.01$, with and without Schur preconditioner $\alpha = k/\rho_{min}$ applied.

the finer meshes. However, due to the small number of numerical experiments it is possible that this observance is not a general rule. According to the restriction (5.1), the step size k is already linearly dependent on the mesh size. The question whether the CFL constant in the variable density model depends on the mesh size stronger than linearly thus remains open for further investigation.

5.2 Convergence of Fixed point iteration

We now take a look at the convergence of the fixed point iteration. To investigate this, the iteration is solved until the relative difference

$$\frac{\|u_i - u_{i-1}\|_{L^2(\Omega)}}{\|u_i\|_{L^2(\Omega)}} \tag{5.2}$$

of two solutions in the fixed point iteration becomes smaller than a tolerance of 10^{-8}. In practice such a small tolerance is usually not required. However, it allows us to compare the convergence with and without precondtioning term. The preconditioner $\alpha = k/\rho_{min}$ is varied to assess its influence on the performance of the method.

All tests in this section are run on the coarse 2D mesh.

5.2.1 Convergence without Schur preconditioning term

First, the convergence of the fixed point iteration without preconditioning term is analyzed. As we saw in Section 5.1, the time step restriction is very strict if no preconditioning term is applied. To limit the amount of computations, only the case with the larger minimal density $\rho_{min} = 0.01$ is considered. The time step sizes investigated are $k = 1 \cdot 10^{-5} h_{min}$, $k = 2 \cdot 10^{-5} h_{min}$ and $k = 3 \cdot 10^{-5} h_{min}$ at three different simulation times t.

The size of the relative difference (5.2) as a function of the iteration count for a number of values for k and t are collected in Figure 5.1. As a reference a line for quadratic convergence and for the convergence of order four is included in every plot. The observed convergence of the fixed point iteration is around order four. Especially for the small step sizes (cf. left column) the convergence order is even larger. For the larger step size, on the other hand, (cf. right column) the order of convergence is lower than four.

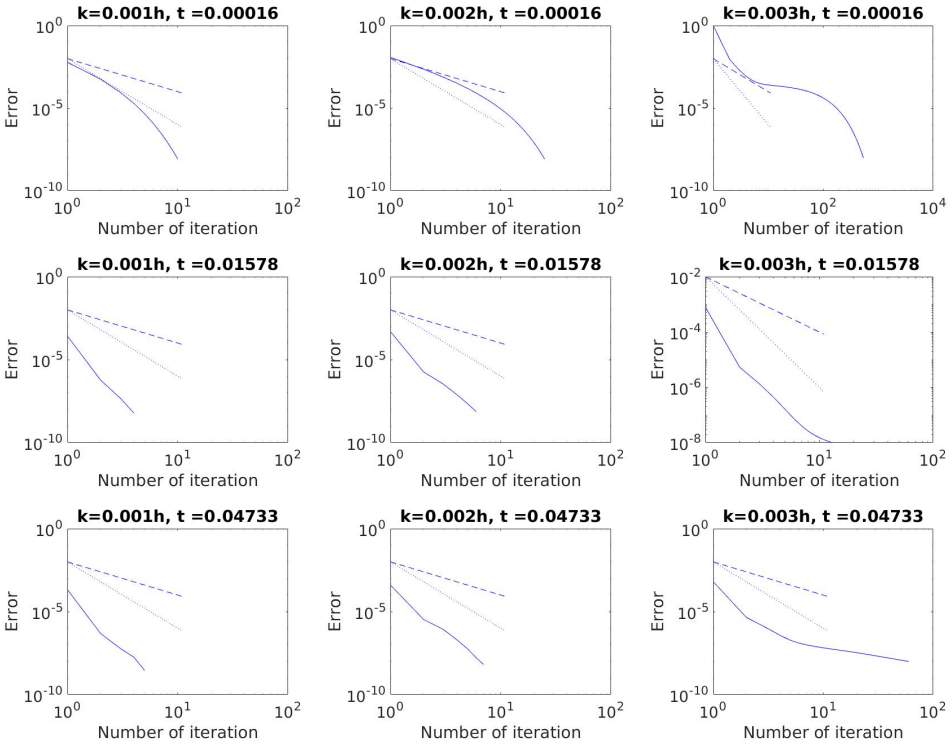


Figure 5.1: Convergence of the fixed point iteration for various time steps k at several times t without preconditioning. The dashed line is a reference line for quadratic convergence and the dotted line for convergence of order 4. Simulations are run on the coarse 2D mesh with $\rho_{min} = 0.01$.

After a few time steps the relative difference is already very small within the first fixed point iteration. At time $t = 0.015$ the relative difference after one iteration is only of order $10^{-3} - 10^{-4}$. It is smaller, the smaller the time step size is chosen.

Hence, the number of fixed point iteration required to reach convergence is smallest for the small time step. Convergence is reached in about 100 iterations in the simulations beginning and about 40 iterations at later times. When the step size is doubled, the number of iterations is increased as well, however not doubled. Thus, we expect the total amount of required fixed point iterations up to a certain simulation time t to be smaller for the larger time steps k.

5.2.2 Convergence with Schur preconditioning term

The convergence of the fixed point iteration is investigated in a similar way for the method including a Schur preconditioning term. The preconditioner $\alpha = k/\rho_{min}$ is also varied to $\alpha_2 = 2k/\rho_{min}$ and $\alpha_{1/2} = k/2\rho_{min}$. In this way the Schur preconditioning term is once doubled and once halved. This allows us to further examine the effect the Schur preconditioning term has on the convergence of the fixed point iteration. In the same sense our suggested preconditioner α is also denoted as α_1.

In Figure 5.2 the relative difference (5.2) as a function of the iteration count is plotted for the method with the small minimal density $\rho_{min} = 0.001$. Results are compared for the time step sizes $k = 0.005h_{min}$, $k = 0.01h_{min}$ and $k = 0.02h_{min}$ for the preconditioners α_1, α_2 and $\alpha_{1/2}$ at several times t.

First of all, we note that the convergence is approximately quadratic. Without Schur preconditioning terms the convergence was observed to be of order four. It is thus inhibited through the Schur term. However, as we noted that the time step k can be increased significantly this slower convergence is not expected to be constrictive.

Further, one can see that for later simulation times t the initial guess is already quite close to the solution, such that the relative difference is of order 10^{-2} after the first fixed point iteration. For small time steps k it is even lower, of approximate order 10^{-3}.

Moreover, the tolerance 10^{-8} is reached significantly faster, the smaller the step size k is chosen. However, doubling the step size k, the number of iterations until convergence is not doubled. Hence, we still ex-

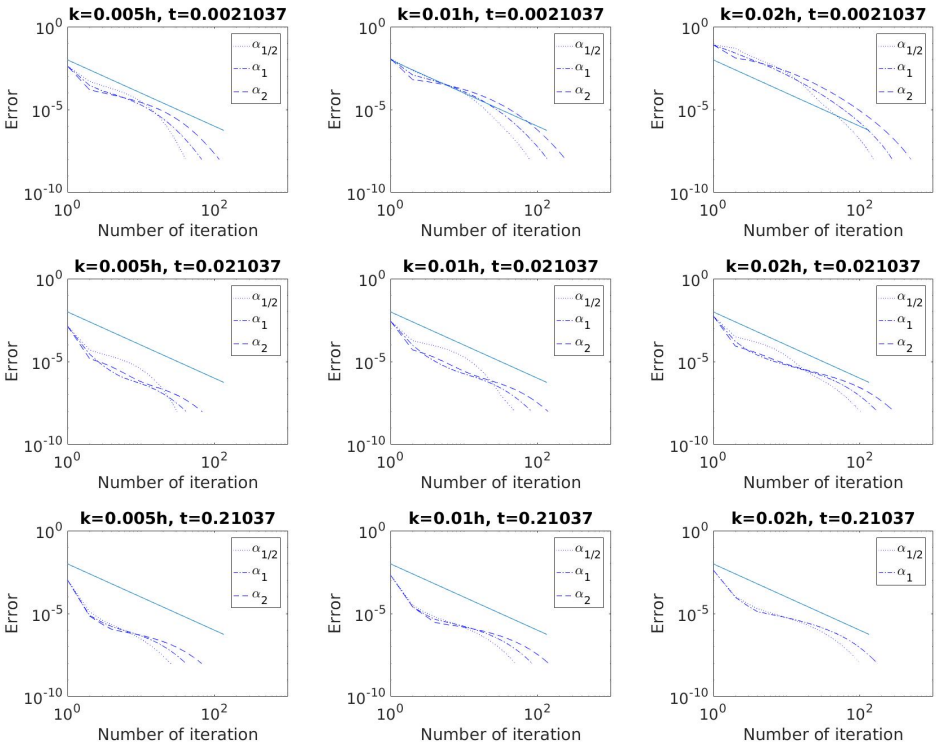


Figure 5.2: Convergence of the fixed point iteration for various time steps k at several times t. The full line is a reference line for quadratic convergence. Simulations are run on the coarse 2D mesh with $\rho_{min} = 0.001$.

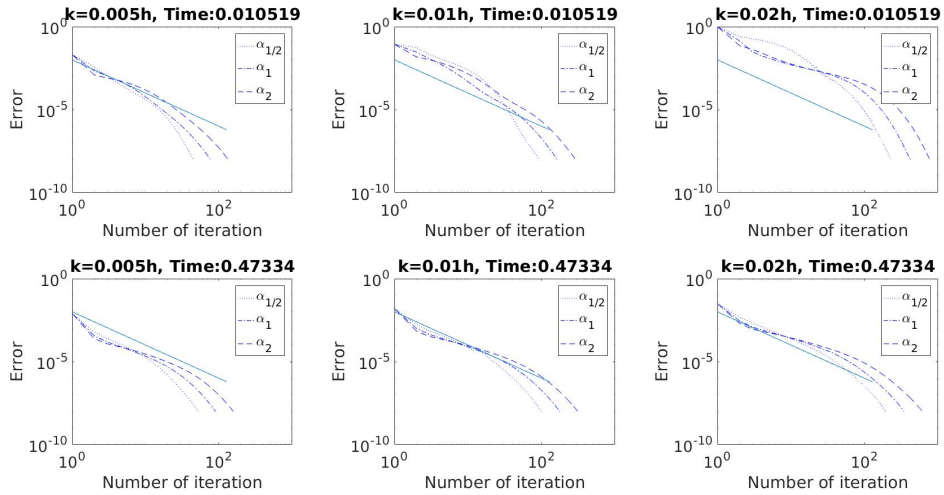


Figure 5.3: Convergence of the fixed point iteration for various time steps k at several times t. The full line is a reference line for quadratic convergence. Simulations are run on the coarse 2D mesh with $\rho_{min} = 0.01$.

pect the overall number of computations to be smaller for larger k. This is investigated further in Section 5.3.

Comparing the preconditioners α_1, α_2 and $\alpha_{1/2}$ one can observe that using the smaller preconditioner $\alpha_{1/2}$, convergence to tolerance 10^{-8} is reached fastest. However, one can also see that for the first iteration steps the other two preconditioners perform better. The small preconditioner becomes only best when the relative difference is quite small, of order $10^{-5} - 10^{-6}$. The large preconditioner α_2 on the other hand, converges last to tolerance 10^{-8}, but performs slightly better than α in the very beginning. This better performance is however only observable for a small range of tolerances and makes a difference of very few fixed point iterations. Overall, the suggested preconditioner α_1 gives good results for the convergence compared to the other two, especially when an extremely low tolerance of 10^{-8} is not required.

To confirm the observances made for this test case, the same kind of simulations are conducted with a larger minimal density $\rho_{min} = 0.01$. The tested time step sizes are $k = 0.05h_{min}$, $k = 0.1h_{min}$ and $k = 0.2h_{min}$ and results collected in Figure 5.3. The convergence of the relative difference in the fixed point iteration behaves very similar to the case with lower minimal density.

Convergence of the fixed point iteration to a tolerance of 10^{-8} is

usually not desirable. From Figures 5.2 and 5.3 we can see that convergence to a tolerance of 10^{-2} is often reached within the first iterations. Only in the initial phase this tolerance is not reached after one iteration. Even a tolerance of order 10^{-4} is attained very quickly. Moreover, the choice of the preconditioner does not have a large influence on the number of iterations required. Especially α_1 and α_2 perform very similarly for convergence until these tolerances.

However, it should be noted that when the fixed point iteration is solved to a larger tolerance the initial guess in the fixed point iteration for the next time step is less exact. This could alter the behavior of convergence. Hence we cannot directly infer from the observations in this section to how the iteration converges if the tolerance is chosen differently. These investigations merely give hints on the convergence.

5.3 Accumulated number of fixed point iterations

In practice the fixed point iteration is not solved to a tolerance of the order 10^{-8}. From Figures 5.2 and 5.3 it became clear that after a few initial time steps a tolerance of order 10^{-2} is usually reached within one fixed point iteration. Therefore the total number of computations to simulate the system until a certain time t is expected to be smallest for the largest possible time step k.

We shall now take a closer look at the accumulated number of fixed point iterations up to a simulation time t. Therefore, we are ignoring the possibly varying number of iterations of Newton's method and the linear solver. Then the accumulated number of fixed point iterations serves as a measure for the total number of computations needed.

Again the modified preconditioners $\alpha_{1/2}$ and α_2 are compared to α_1. We investigate for which choice of preconditioner and time step the simulation is stable and for which the least amount of iterations are needed.

Figures 5.4 and 5.5 show results with low minimal density $\rho_{min} = 0.001$ on the coarser grid. In Figure 5.4 the accumulated number of fixed point iterations is plotted over the simulation time t for various values of k and the three preconditioners α_1, $\alpha_{1/2}$ and α_2. Since the fixed point iteration seldom requires more than one iteration to converge, the slope is almost constant in all cases. One can clearly see that

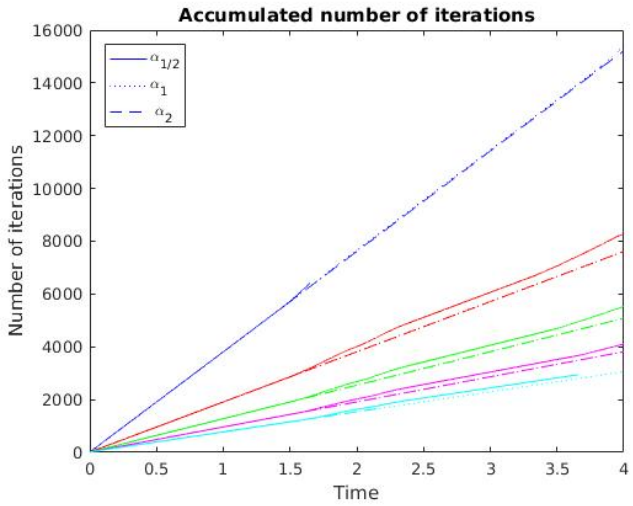


Figure 5.4: Accumulated number of fixed point iterations as a function of time for several preconditioners. Simulations are run on the coarse 2D mesh with $\rho_{min} = 0.001$ for the time step sizes $k = 0.005h$ (blue), $k = 0.01h$ (red), $k = 0.015h$ (green), $k = 0.02h$ (magenta), $k = 0.025h$ (cyan).

the total number of iterations required is smaller the larger the time step k is chosen. Moreover, the precondtitioners α_1 and α_2 perform slightly better than $\alpha_{1/2}$. Not all simulations with the small preconditioner $\alpha_{1/2}$ are even stable until simulation time $t = 4$, for example when the time step is chosen relatively small or large.

In Figure 5.5, the accumulated number of fixed point iterations up to time $T = 4$ is plotted as a function of the step size k for the three preconditioners. This illustrates again what has been observed in Figure 5.4. There is a clear trend towards a smaller number of required iterations for larger time steps k. Moreover, we can see that α_2 and α_1 perform similarly, while the method with $\alpha_{1/2}$ generally requires more iterations.

The equivalent tests have been performed on the same mesh, but with a larger minimal density $\rho_{min} = 0.01$, which allows the time step size k to be chosen of order 10 larger. Simulations can therefore be executed for longer times t. The results are depicted in Figures 5.6 and 5.7. The accumulated number of iterations plotted over simulation time t in Figure 5.6 shows that, as before, after an initial phase the slope of the graphs becomes nearly constant, due to the convergence within

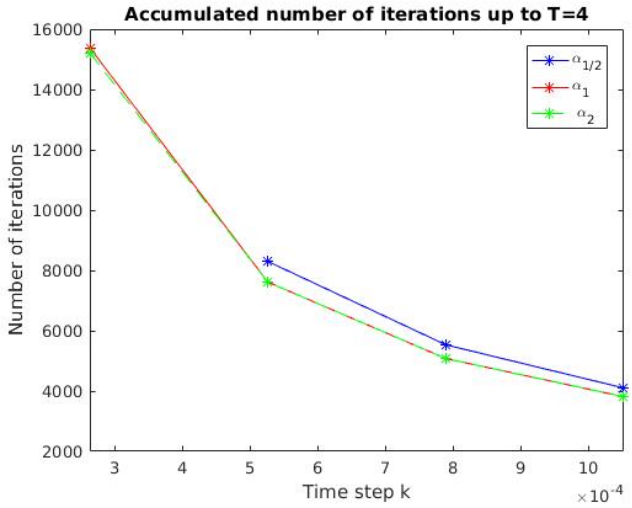


Figure 5.5: Accumulated number of fixed point iterations up to time $T = 4$ as a function of time step size k for several preconditioners. Simulations are run on the coarse 2D mesh with $\rho_{min} = 0.001$.

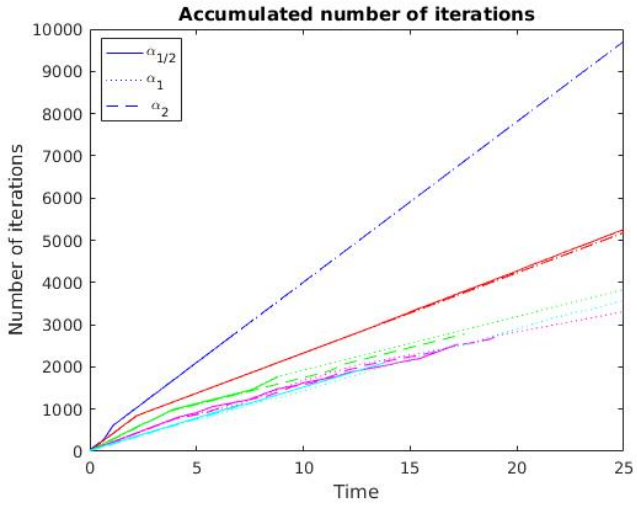
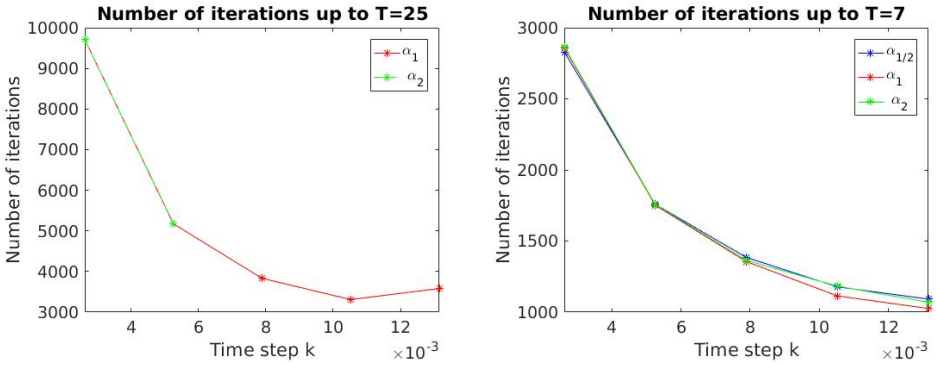


Figure 5.6: Accumulated number of fixed point iterations as a function of time for several preconditioners. Simulations are run on the coarse 2D mesh with $\rho_{min} = 0.01$ for the time step sizes $k = 0.05h$ (blue), $k = 0.1h$ (red), $k = 0.15h$ (green), $k = 0.2h$ (magenta), $k = 0.25h$ (cyan).



(a) (b)

Figure 5.7: Accumulated number of fixed point iterations up to time $T = 25$ (a) and $T = 7$ (b) as a function of time step size k for several preconditioners. Simulations are run on the coarse 2D mesh with $\rho_{min} = 0.01$.

one fixed point iteration. Using $\alpha_{1/2}$ as preconditioner, the method is unstable for most tested time steps. It is only stable until $T = 25$ for the step size $k = 0.01h_{min}$. As previously, the graphs for α_2 and α lie very close together. In a few cases, for example using $k = 0.015h_{min}$, the method with preconditioner α_2 requires a few iterations less than with α_1. However, for large time steps k the method is only stable until $t = 25$ if the preconditioner α_1 is applied. This clearly depicts the superiority of the derived preconditioner compared to the two modified ones.

In Figure 5.7(a) the accumulated number of fixed point iterations up to time $T = 25$ is shown as a function of the time step size k. For small step sizes, α_1 and α_2 lie very close to each other. For larger step sizes the method is only stable with α_1. As in the previous example, the total number of iteration first decreases with increasing step size k. However, for the largest tested step size $k = 0.25h_{min}$, the number of iterations slightly increases again. This increase is however quite small (from 3310 for $k = 0.2h_{min}$ up to 3581 for $k = 0.25h_{min}$).

Figure 5.7(b) shows the accumulated number of fixed point iterations up to time $T = 7$ as a function of k. Until this time all simulations are stable, so that we can compare all three preconditioners. For small step sizes k the choice of preconditioner does not have a strong influence. For the larger step sizes, however, the preconditioner α_1 does perform better than $\alpha_{1/2}$ and α_2.

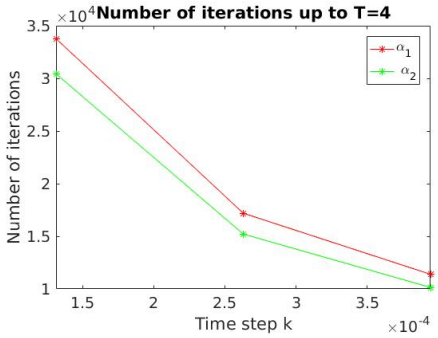


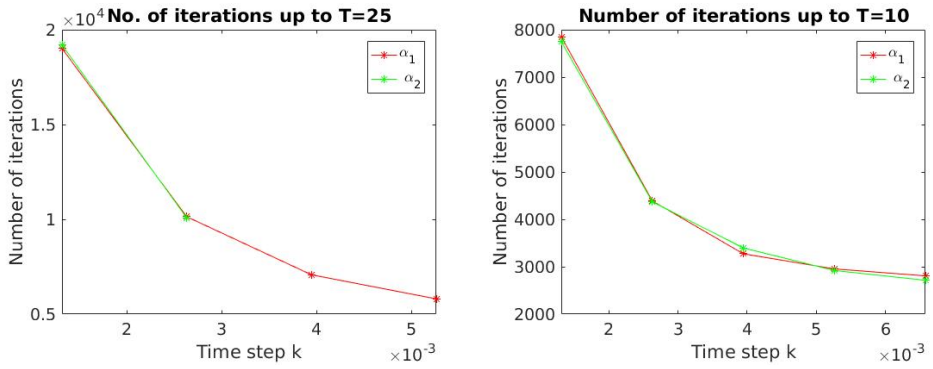
Figure 5.8: Accumulated number of fixed point iterations up to time $T = 4$ as a function of time step size k for several preconditioners. Simulations are run on the fine 2D mesh with $\rho_{min} = 0.001$.

The same tests are repeated on the finer 2D mesh. Using the small preconditioner $\alpha_{1/2}$, the simulations are barely stable. We are therefore concentrating on the preconditioners α_1 and α_2.

Figure 5.8 shows the result with a lower minimal density ρ_{min}. As before, one can see that the number of iterations required until simulation time $t = 4$ decreases as the time step sizes k is increased. However, in contradiction to the previous observations, the larger preconditioner α_2 performs better than the suggested preconditioner α_1.

In Figure 5.9 the outcome for the tests with larger minimal density $\rho_{min} = 0.01$ are presented. With the larger step sizes k the simulation is only stable until simulation time $t = 25$ with the proposed preconditioner α_1 (cf. Figure 5.9(a)). This result is supporting the observation from the simulations on the coarse mesh. The preconditioner α_1 is clearly predominant in this test case.

To compare it to the larger preconditioner in Figure 5.9(b) the accumulated number of fixed point iterations are measured up to a simulation time $t = 10$ where both methods are stable. Both preconditioners α_1 and α_2 perform very similarly up to this time. For a step size of $k = 0.15h_{min}$ the smaller preconditioner α_1 performs slightly better, while for $k = 0.25h_{min}$ the doubled preconditioner α_2 performs a little better. The difference between α_1 and α_2 is however not significant.



(a) (b)

Figure 5.9: Accumulated number of fixed point iterations up to time $T = 25$ (a) and $T = 10$ (b) as a function of time step size k for several preconditioners. Simulations are run on the fine 2D mesh with $\rho_{min} = 0.01$.

5.4 Comparison with 3D simulations

To confirm the observations made with the 2D test case, a number of additional experiments are performed on the 3D test case presented in Section 3.2.2.

In Section 5.3 the accumulated number of fixed point iterations is used as a measure for the number of computations required to simulate the system until a certain time t. Thereby, the computations in the internal iterations of Newton's method and the linear solver inside every fixed point iteration loop are neglected. However, the number of these internal iterations could possibly vary between different preconditioners α and time step sizes k. Therefore, we now also measure the computation time spent in every outer iteration loop. This time is the feature we are ultimately interested in.

As for the 2D test case the tolerance chosen for the fixed point iteration is 10^{-2}. The performance of the methods using the modified preconditioners $\alpha_{1/2}$ and α_2 are compared to the method with the proposed preconditioner α_1.

Figure 5.10 shows the accumulated number of fixed point iterations as a function of time for the three preconditioners and several time step sizes k. The result resembles the figures in Section 5.3. After an initial phase all graphs become relatively straight lines which is due to the fact that the fixed point iteration converges in the first iteration

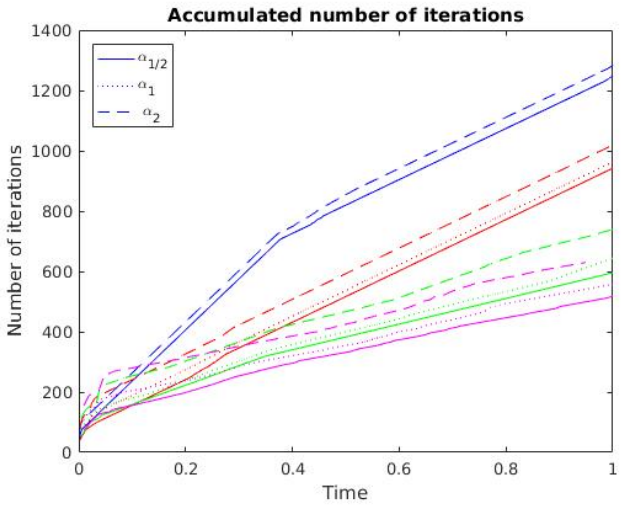


Figure 5.10: Accumulated number of fixed point iterations as a function of time for several preconditioners. Simulations are run on the 3D mesh with $\rho_{min} = 0.001$ for the time step sizes $k = 0.1h$ (blue), $k = 0.2h$ (red), $k = 0.4h$ (green), $k = 0.8h$ (magenta)

step. It is notable that this transition to a constant (and minimal) slope happens earlier the larger the time step k is chosen.

As in the 2D case the trend is that the accumulated number of iterations is smaller the larger the time step k. Only in the short initial stage this cannot be confirmed clearly.

In Section 5.3 the preconditioner α_1 generally performed best, with only the preconditioner α_2 performing similarly well in some cases. This cannot be detected in Figure 5.10. Quite distinctly the preconditioner α_2 performs worst for all step sizes k. For the largest tested step size $k = 0.8h$ it does not even give a stable solution until the end of simulation at $t = 1$. The preconditioners α_1 and $\alpha_{1/2}$ perform more similarly. However, for larger time steps the method with $\alpha_{1/2}$ requires clearly the least amount of fixed point iterations. This observation is quite antithetic to the observations from the 2D test case.[1]

Let us move on to a study of the computation time spent in the fixed point iteration. Figure 5.11 shows the accumulated time for the fixed point iteration as a function of the simulation time for the same

[1]The reader might suspect a smaller preconditioner, say $\alpha_{1/4} = k/4\rho_{min}$ to perform even better. However, none of the experiments executed with this preconditioner resulted in a stable solution.

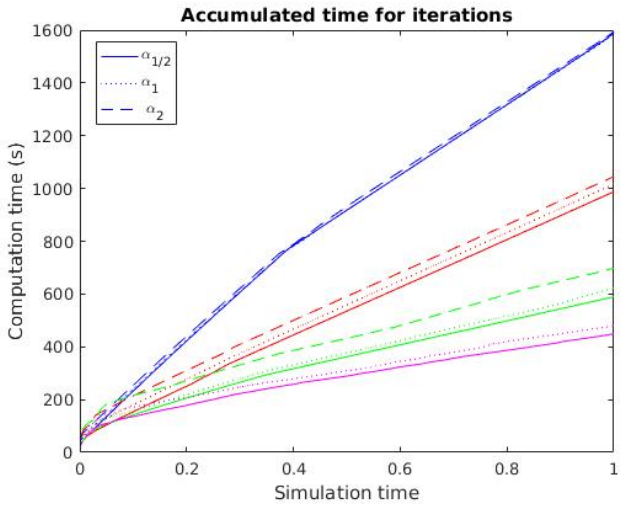


Figure 5.11: Accumulated time for fixed point iterations as a function of time for several preconditioners. Simulations are run on the 3D mesh with $\rho_{min} = 0.001$ for the time step sizes $k = 0.1h$ (blue), $k = 0.2h$ (red), $k = 0.4h$ (green), $k = 0.8h$ (magenta)

preconditioners and time step sizes as in Figure 5.10. As expected, the result does not differ substantially from the plot of the accumulated number of iterations. The larger the time step k and the smaller the preconditioner α, the faster is the overall computation until $t = 1$. However, comparing the different preconditioners for each time step k it seems that the choice of preconditioner has a smaller influence on the computation time as it does on the number of iterations. This would mean that for α_2 and α_1 on average a single fixed point iteration loop takes less time compared to the loop with $\alpha_{1/2}$.

In Figure 5.12 the accumulated number of fixed point iterations and the accumulated computation time spent on these are presented as a function of the time step k. The figures demonstrate again that both features become smaller with increasing time step and decreasing preconditioner. Nonetheless, the difference seems larger in the number of iterations than in the required computation time.

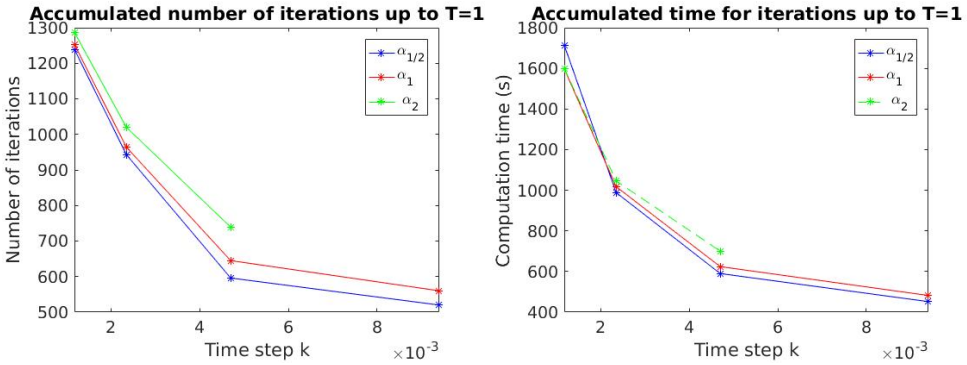
To investigate this further we compare the raw data. Table 5.3 shows the accumulated number of fixed point iteration for simulations with the preconditioners $\alpha_{1/2}$ and α_1. The relative difference when decreasing the preconditioner from α_1 to $\alpha_{1/2}$ is listed for each time step size k. The equivalent data for the accumulated computation

Time step size k	$\alpha_{1/2}$	α_1	relative difference in %
$0.1h$	1239	1253	1.12
$0.2h$	943	964	2.18
$0.4h$	596	645	7.60
$0.8h$	520	560	7.14

Table 5.3: Accumulated number of fixed point iterations for the simulation with preconditioners $\alpha_{1/2}$ and α_1 for each time step size k and their relative difference.

Time step size k	$\alpha_{1/2}$	α_1	relative difference in %
$0.1h$	1714s	1598s	-7.28
$0.2h$	989s	1018s	2.85
$0.4h$	590s	624s	5.47
$0.8h$	452s	482s	6.28

Table 5.4: Accumulated computation time for the simulation with preconditioners $\alpha_{1/2}$ and α_1 for each time step size k and their relative difference.



(a) (b)

Figure 5.12: Accumulated number of fixed point iterations (a) and accumulated time for fixed point iterations (b) up to time $T = 1$ as a function of time step size k for several preconditioners. Simulations are run on the 3D mesh with $\rho_{min} = 0.001$.

time spent in the fixed point iterations is collected in Table 5.4. Indeed, the relative difference in computation time is generally smaller than the relative difference in number of fixed point iterations. With a time step of $k = 0.4h$ the relative difference in number of iterations is 7.60%, while it is only 5.47% for the computation time. For a small time step of $k = 0.1h$ the method with preconditioner α_1 is even faster than with $\alpha_{1/2}$ although more fixed point iterations are performed with α_1. Hence, on average the time spent in one iteration loop must be larger for the small preconditioner $\alpha_{1/2}$.

We compute the average time required for one fixed point iteration loop by dividing the accumulated computation time by the accumulated number of fixed point iterations. In Table 5.5 the results are listed for all three preconditioners and the four time step sizes. One can observe two tendencies. Firstly, the average time spent in one iteration loop is smaller for large time steps and secondly it decreases for large preconditioners.

The larger the step size k and the preconditioner α are chosen, the more iterations are required in each time step. For the later fixed point iterations one can expect the starting guess for Newton's method to be closer to the solution. Hence, more fixed point iterations being performed in one time step brings down the average time spent in one iteration loop. Thus, the decreasing average computation time for larger time step sizes and preconditioners can be explained.

Time step size	**Computation time / Iteration loop**		
k	$\alpha_{1/2}$	α_1	α_2
$0.1h$	1.3833s	1.2750s	1.2430s
$0.2h$	1.0485s	1.0557s	1.0254s
$0.4h$	0.9900s	0.9677s	0.9476s
$0.8h$	0.8688	0.8608	–

Table 5.5: Average time spent on one iteration loop for several preconditioners α and time step sizes k.

This shows us that although the accumulated number of fixed point iterations gives a hint on the computational expense of the method, we should accept this information with caution. In the tested cases the best performing preconditioner regarding both features coincides. The relation between the performance of different preconditioners with a fixed time step size is represented through the accumulated number of fixed point iterations.

However, as most of the computational expense is expected to occur in the first fixed point iteration measuring the number of outer iterations might give a misleading idea. It is therefore favorable to also measure the number of internal iterations in Newton's method and the linear solver or the computational time directly.

Chapter 6

Discussion

To conclude this master thesis we give a short summary of the work and the main results. Lastly, some open questions and inspirations for future work are given.

6.1 Summary

In this master thesis a preconditioner for a Finite Element Method for the variable density incompressible Navier-Stokes equations was suggested. This preconditioner can be induced to the FEM by adding the term

$$(\alpha \nabla p, \nabla q)$$

to the weak residual and solving the momentum, density and pressure equations separately in the form of a fixed point iteration. The factor α is also referred to as the preconditioner. Our suggested preconditioner for the variable density model is

$$\alpha = \frac{k}{\rho_{min}}.$$

The preconditioner's performance was experimentally investigated for a number of test cases. It was examined if the introduction of the preconditioner term allowed the time step size k to be chosen larger and if larger k speed up the simulations. Moreover, the suggested preconditioner was slightly varied to $\alpha_{1/2} = 1/2\alpha$ and $\alpha_2 = 2\alpha$ in order to investigate whether it is indeed optimal.

6.2 Conclusions

The proposed preconditioner has been shown to perform well. We conclude the following observations.

- Introducing the preconditioning term to the method allows us in the tested cases to increase the time step size k by a factor of about 50, which means that the computational expense decreases by about the same factor.
- The fixed point iteration converges with order two. For the 2D test cases, a tolerance of 10^{-8} is first reached when applying the preconditioner $\alpha_{1/2}$. For higher tolerances (as are usually used in practice) the preconditioners α and α_2 perform best.
- Measuring the accumulated number of fixed point iterations until the end of simulation, the number is lower the larger the time step size k is chosen.
- For the 2D test case the preconditioner α was shown to perform best in terms of accumulated number of fixed point iterations and stability in most of the performed experiments.
- For the experiment performed on the 3D test case the method with preconditioner $\alpha_{1/2}$ resulted in the lowest accumulated number of fixed point iterations and computational time spent on these. However, compared to the preconditioner α, the difference in computational time was shown to be smaller than the difference in the number of iterations.

6.3 Open questions and future work

Through this master thesis a number of questions appeared that could not yet be answered to full extend. Some stimulus for further investigation is given here.

- In the study of the maximal possible time step k it seemed that the dependence of the step size on the mesh size was stronger than linear. It would thus not be determined by a CFL type condition. However, this dependence might be only a small irregularity. The convergence criteria for the fixed point iteration could still be studied more rigorously.

- On the 2D test case the accumulated number of iterations was used as a measure for the computational expense. However, as we discussed in Section 5.4 the computational expense is only represented to a limited extend through the accumulated number of iterations. For a more detailed picture it is suggested to measure the internal iterations or the computation time. However, as was seen in Figure 5.6 the method with preconditioner α tends to be stable for the largest time step sizes k. One can therefore still expect this preconditioner to perform best regarding the suggested features.

- 3D experiments have only been executed for one test case and for a relatively short simulation time. The small preconditioner $\alpha_{1/2}$ that performed best for these experiments was the worst performing preconditioner in the 2D test cases. It is questionable whether the small preconditioner would withstand further testing. In the 2D test cases the method using the small preconditioner was observed to be most prone to instabilities.

Bibliography

[1] G. N. Wells et al. A. Logg, K.-A. Mardal. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[2] Å. Björck. *Numerical Methods in Matrix Computations*. Springer, 2015.

[3] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation, Vol. 22*, 1968.

[4] C. Niyazi D. Castanon. Fenics-hpc code `var_den_ale`; https://bitbucket.org/fenics-hpc/unicorn/branch/unicorn_var_den_ale.

[5] A. Tourin E. Rouy. A viscosity solutions approach to shape-from-shading. *SIAM J. Numer. Anal., 29(3), 867–884*, 1992.

[6] M. Vázquez G. Houzeaux, R. Aubry. Extension of fractional step techniques for incompressible flows: The preconditioned orthomin(1) for the pressure Schur complement. *Computers & Fluids 44, Pages 297-313*, 2011.

[7] R. Aubry J.M. Cela G. Houzeaux, M. Vázquez. A massively parallel fractional step solver for incompressible flows. *Journal of Computational Physics Volume 228, Issue 17, Pages 6316-6332*, 20 September 2009.

[8] P. M. Gresho. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part 1: Theory. *Int. J. Numer. Meth. Fluids, 11, Pages 621–659*, 1990.

[9] A. Wathen H. Elman, D. Silvester. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, 2005.

[10] J.Ph. Ponthot A. Rodriguez Ferran J. Donea, A. Huerta. Arbitrary Lagrangian–Eulerian methods. *Encyclopedia of Computational Mechanics, Vol. 1: Fundamentals*, 2004.

[11] C. Johnson J. Hoffman. *Computational Turbulent Incompressible Flow, Applied Mathematics: Body and Soul 4*. Springer, 2007.

[12] N. Jansson J. Hoffman. A computational study of turbulent flow separation for a circular cylinder using skin friction boundary conditions. In *Salvetti M., Geurts B., Meyers J., Sagaut P. (eds) Quality and Reliability of Large-Eddy Simulations II*, 2001.

[13] N. Jansson J. Hoffman, J. Jansson. Fenics-hpc: Automated predictive high-performance finite element computing with applications in aerodynamics. *Proceedings of the 11th International Conference on Parallel Processing and Applied Mathematics*, 2015.

[14] N. Jansson R. V. De Abreu J. Hoffman, J. Jansson. Towards a parameter-free method for high Reynolds number turbulent flow simulation based on adaptive finite element approximation. *Computer Methods in Applied Mechanics and Engineering, Vol. 288*, 2015.

[15] N. Jansson R. Vilela De Abreu C. Johnson J. Hoffman, J. Jansson. Computability and adaptivity in cfd. *Encyclopedia of Computational Mechanics*, 2016.

[16] Stöckli J. Hoffman, J. Jansson. Unified continuum modeling of fluid-structure interaction,. *Math. Models Methods Appl. Sci. 21, 491*, 2011.

[17] M. M. Ginard D. C. Quiroz L. Saavedra E. Krishnasamy A. Goude J. Hoffman J. Jansson, V. D. Nguyen. Direct finite element simulation of turbulent flow for marine based renewable energy. *Preprint*, 2017.

[18] A. Salgado J.-L. Guermond. A splitting method for incompressible flows with variable density based on a pressure poisson equation. *Journal of Computational Physics, Volume 228, Issue 8, Pages 2834-2846*, 2009.

[19] J. Shen J. L. Guermond, P. Minev. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering, Vol. 195, Pages 6011-6045*, 2005.

[20] J. Jansson. Fenics-hpc: Automated adaptive high-performance finite element computing with applications in turbulent flow and fluid-structure interaction. Lecture Presentation, HPFEM : High performance finite element modeling; CST, CSC, KTH, 2016.

[21] P. Hansbo C. Johnson K. Eriksson, D. Estep. *Computational Differential Equations*. Cambridge University Press, 2009.

[22] T. Klock M. Jahn. A level set toolbox including reinitialization and mass correction algorithms for fenics. *Berichte aus der Technomathematik, Report 16–01,*, 2016.

[23] S. Osher M.Sussman, P. Smereka. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics, Volume 114 Issue 1, Pages 146 - 159*, 1994.

[24] R. Codina S. Badia. Algebraic pressure segregation methods for the incompressible Navier-Stokes equations. *Archives of Computational Methods in Engineering, Volume 15, Issue 3, pp 343–369*, 2008.

[25] A. Reusken S. Gross. *Numerical Methods for Two-phase Incompressible Flows*. Berlin/Heidelberg : Springer, 2011.

[26] J. A. Sethian S. Osher. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-jacobi formulations. *Journal of Computational Physics, Volume 79, Issue 1, November 1988, Pages 12-49*, 1988.

[27] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States, Vol.93(4), p.1591(5)*, 1996.

[28] J. Shen. On error estimates of some higher order projection and penalty-projection methods for Navier-Stokes equations. *Numerische Mathematik, Vol. 62, Issue 1, pp 49–73*, 1992.

[29] S. Turek. *Efficient Solvers for Incompressible Flow Problems*. Springer-Verlag Berlin Heidelberg, 1999.

[30] R. Témam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (ii). *Archive for Rational Mechanics and Analysis, Vol.33(5), page 115-152*, 1969.

List of Figures

List of Tables

TRITA -MAT-E 2017:64
ISRN -KTH/MAT/E--17/64--SE