Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Fast iterative methods for solving the incompressible Navier-Stokes equations

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

Carlos Echeverría Serur

Delft, The Netherlands
August 20, 2013

TU Delft Delft University of Technology

MSc THESIS APPLIED MATHEMATICS

"Fast iterative methods for solving the incompressible Navier-Stokes equations"

Carlos Echeverría Serur

Delft University of Technology

Daily supervisor	**Responsible professor**
Prof. dr. ir. C. Vuik	Prof. dr. ir. C. Vuik

Committee members

Prof.dr.ir. C. Vuik	Prof.dr.ir. A. Heemink
Dr.ir. M.B. van Gijzen	Dr.ir. W.T. van Horssen

August 20, 2013 Delft, The Netherlands

Acknowledgements

This master thesis has been written within the Erasmus Mundus Master's program *Computer Simulations for Science and Engineering (COSSE)*[1]. It represents the culmination of my efforts spent as a master student studying one year at the Technische Universität Berlin, Germany, and one year at the Technische Universiteit Delft, The Netherlands.

At the beginning of this list, I would like to thank my supervisor Prof. Dr. Ir. Kees Vuik, whom without his thoughtful comments I would not have been able to achieve the goals set out for this thesis. His ever-welcoming attitude provided me with the confidence and support that allowed me to go through with this project, and his professionalism as an academic is a great example to follow.

Equally special was the help, support, and inspiration provided to me by Prof. Carlos Málaga, Prof. Reinhard Nabben, and Prof. Jörg Liesen. Each one of whom showed me great characteristics of being a modern academic and represent, each on its own way, great examples of exceptional individuals.

On the second place of this list, I would like to thank my family. Raquel, my mother, always showed me the bright side of life. Her love and guidance accompanied me every day during these years and the thanks I can give her are only infinite. Alberto, my brother, has always been a source of inspiration for me. His passion for the natural world and the life within it makes me remember how a true scientist should work.

Next, I would like to thank my dear Evelyn. Her love and tenderness is present with me ever since the first day we met. I don't know where I would be without your love.

Lastly, I would like to thank all of my friends. I will always remember the good times spent with my colleagues of the master programme: Firat, Luis, Manuel, Martin, Matthew, Xiao, and Jeroen - working with you guys taught me things that no lecture can teach, I

[1] For more information, please visit: `http://www.kth.se/en/studies/programmes/em/cosse`

will never forget you guys. Lastly, I would like to mention my dear and true friends from the heart: Dominik, Max, Maxi, David and Katharina - my family out of home, the idea of getting back together always made me smile.

During these two years, I had the opportunity to meet and share time with people that are, or became, of extremely high value to me. From cousins to students to professors to complete strangers, I would like to thank anybody who is not mentioned in this list. Thanks for making my master's degree such a rich and diverse experience.

Contents

Chapter 1

Introduction

1.1 Motivation

The numerical solution of the incompressible Navier-Stokes (N-S) equations is an area of much importance in contemporary scientific research. Except for some simple cases, the analytical solution of the N-S equations is impossible. Therefore, in order to solve these equations, it is necessary to apply numerical techniques (See [34], [39], [40], [48]). The most commonly used numerical discretization techniques include finite difference methods (FDM), finite volume methods (FVM) and finite element methods (FEM). The discretization approach followed throughout this thesis is done by the FEM. Due to the nonlinear character in the behavior of fluids, the solution of the N-S system of equations requires a suitable linearization of the algebraic system arising from the FEM discretization of the original equations. The resulting linear system of equations gives rise to a so called saddle-point problem, an algebraic system which is non-symmetric, indefinite, and typically ill conditioned. This process can be summarized in the next scheme, where in the left-hand side of the arrow we have the original N-S system of equations and on the right-hand side we have a linear system with A nonsymetric, indefinite, and ill-conditioned:

$$\left.\begin{array}{rl} -\nu\nabla^2\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \\ \nabla\cdot\mathbf{u} = 0 & \text{in } \Omega \end{array}\right\} \longrightarrow Ax = b$$

Saddle-point problems also arise in electrical circuit problems, linear elasticity, constraint optimization and many other fields. A survey of saddle-point problems is given by Benzi, Golub and Liesen in [3]. The efficient solution of this type of linear algebraic problem is a challenge (See [3], [20], [28]). The primary interest in these types of problems is due to the fact that most of the computing time and memory of a computational implementation is consumed by the solution of this system of equations.

The fastest way of solving such systems is done by adopting an iterative approach in the solution process of the algebraic system, mainly by the use of a Krylov subspace method

combined with a preconditioned linear system of equations (See [33]). In the case of the Navier-Stokes problem, the most used type of preconditioners belong to a branch of block preconditioners known as SIMPLE-type preconditioners (Semi Implicit Pressure Linked Equations) in literature (See [23], [24], [43], [46]). These methods decouple the system and solve separate subsystems of the velocity and pressure. The pressure subsystem arises from an appropriate approximation of the Schur complement of the system. The use of these preconditioners has enhanced the solution process of the N-S system and provides an implementation which reduces the computational time needed to solve the system when compared to the time needed to solve the un-preconditioned system.

Nevertheless, it has been found that there still exists room for improving the SIMPLE-type preconditioners. As published in [43], we can see the existence of a stagnation behavior in the solution process of the linearized system. This stagnation behavior is problem dependent and is usually found when stretched grids are used in the discretization of the system, or whenever the conditions of the problem lead to an ill conditioned spectrum of the linearized matrix. The main effort of this thesis is to investigate and, if possible, to eliminate this stagnation behavior, thus providing a more efficient way of solving the N-S system of equations.

1.2 Research direction

In order to achieve the desired goal of this thesis, we present a study of the available preconditioners used to achieve a fast solution of the saddle point problem arising from the linearization of the N-S system of equations. We will focus our attention on the SIMPLER preconditioner and we will present problems for which the stagnation behavior is present. Once we have implemented the solution method and have reproduced this type of behavior, we are interested in approaching the following questions:

- Why is there a stagnation phase in the iterative solution of the SIMPLE-preconditioned Navier-Stokes algebraic system?
- Why does the number of iterations increase for stretched grids?

In order to address the first of these questions, an investigation of the stagnation behavior of the initial phase of the SIMPLER preconditioner is carried out. It was found that a detailed study of the spectrum of the SIMPLER preconditioned matrix provides insight in the problem of identifying the reasons of such stagnation. An eigenvalue deflation approach is followed after the identification of some "ill conditioned" eigenvectors in order to eliminate the stagnation behavior. The approach is successful, reducing the stagnation phase and the total number of iterations of the preconditioned `GMRES` method.

Regarding the increase of iteration count for stretched grids, it was found that the reason for this increment in the number of iterations comes as a consequence of the degeneration of the so called inf-sup condition for finite elements. It is expected that a macroelement construction (See [11], [26]) of the finite elements may lead to a stable computational model for any conceivable grid. The proof of stability of this construction is included in this thesis.

1.3 Chapter outline

In Chapter 2, the Navier Stokes equations are introduced and discretized via the Finite Element Method. The resulting algebraic system is then linearized via the Newton or Picard method. Lastly, a comment on finite element selection is given, guided by a discussion of inf-sup stability. In Chapter 3, the main concepts of linear algebra which are used in this thesis are presented. The theory behind Krylov subspace methods is discussed and the general theory of preconditioners is presented. Also, a general approach to eigenvector deflation is explained. In Chapter 4, specific block-type preconditioners for the Navier-Stokes equations are studied including preconditioners based on approximate commutators and SIMPLE-type preconditioners. In Chapter 5, the numerical results of the investigations are presented. This section includes a set of reference problems (benchmarks) that make use of the theory presented in the previous sections. The two-dimensional Poiseuille flow as well as the driven cavity flow and the backward facing step problems are shown. The problems are studied by the use of the `MATLAB` toolbox `IFISS` [1] (Incompressible Flow Iterative Solution Software) developed in the University of Manchester, UK. In Chapter 6, a discussion of the results obtained in the numerical experiments is provided. Finally, in Chapter 7, the conclusions of this master thesis are presented, together with ideas for future research in this topic.

[1] http://www.manchester.ac.uk/ifiss

Chapter 2

Discretization and linearization of the Navier-Stokes equations

Partial differential equations in general, or the governing equations in fluid dynamics in particular, are classified into three categories: (1) elliptic, (2) parabolic, and (3) hyperbolic. The physical situations these types of equations represent can be illustrated by the flow velocity relative to the speed of sound. Consider that the flow velocity u is the velocity of a body moving in a fluid at rest. The movement of this body disturbs the fluid particles ahead of the body, setting off the propagation velocity equal to the speed of sound c. The ratio of these two competing speeds is defined as Mach number:

$$M = \frac{u}{c}$$

For subsonic speed, $M < 1$, as time t increases, the body moves a distance, ut, which is always shorter than the distance at the sound wave. If, on the other hand, the body travels at the speed of sound, $M = 1$, then the observer does not hear the body approaching him prior to the arrival of the body, as these two actions are simultaneous. For supersonic speed, $M > 1$, the velocity of the body is faster than the speed of sound. The governing equations for subsonic flow, transonic flow, and supersonic flow are classified as elliptic, parabolic, and hyperbolic, respectively.

2.1 Discretization

In continuum mechanics, incompressible flow refers to a flow in which the material density is constant within an infinitesimal volume that moves with the velocity of the fluid. The Mach number can be used to determine if a flow can be treated as an incompressible flow. If $M \ll 1$ and the flow is quasi-steady and isothermal, compressibility effects will be small and a simplified incompressible flow model can be used. The incompressible flow of a

Newtonian fluid is governed by the behavior defined by the set of equations

$$-\nu\nabla^2\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega \tag{2.1}$$

$$\nabla\cdot\mathbf{u} = 0 \text{ in } \Omega \tag{2.2}$$

Equation (2.1) represents the conservation of momentum, while equation (2.2) enforces conservation of mass. The second equation is also referred to as the incompressibility constraint. The boundary value problem that is considered is the system composed of equations (2.1) and (2.2) posed on a two or three dimensional domain Ω, together with boundary conditions on $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ given by

$$\mathbf{u} = \mathbf{w} \text{ on } \partial\Omega_D, \quad \nu\frac{\partial\mathbf{u}}{\partial n} - \mathbf{n}p = \mathbf{s} \text{ on } \partial\Omega_N. \tag{2.3}$$

The variable $\mathbf{u}$ is a vector-valued function representing the velocity of the fluid, and the scalar function p represents the pressure. Note that the Laplacian of a vector function is simply the vector obtained by taking the Laplacian of each component in turn. Moreover, there is little loss of generality if $\mathbf{f}$ is set to zero. A conservative body force (e.g. gravity) is the gradient of a scalar field, that is, $\mathbf{f} = -\nabla\Phi$, and thus it can be incorporated into the system by redefining the pressure ($p \leftarrow p + \Phi$).
To derive a weak formulation of the Navier-Stokes equations we require that for an appropriate set of test functions $\mathbf{v}$ and q,

$$\nu\int_\Omega \mathbf{v}\cdot(-\nabla^2\mathbf{u}) + \int_\Omega \mathbf{v}\cdot(\mathbf{u}\cdot\nabla\mathbf{u}) + \int_\Omega \mathbf{v}\cdot\nabla p = 0, \tag{2.4}$$

$$\int_\Omega q(\nabla\cdot\mathbf{u}) = 0, \tag{2.5}$$

for all $\mathbf{v}$ and q in suitably chosen spaces of test functions. The continuity requirements on the weak solution $(\mathbf{u}, p)$ can be reduced by "transferring" derivatives onto the test functions $\mathbf{v}$ and q:

$$\begin{aligned}
-\int_\Omega \mathbf{v}\cdot\nabla^2\mathbf{u} &= \int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} - \int_\Omega \nabla\cdot(\nabla\mathbf{u}\cdot\mathbf{v}) \\
&= \int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} - \int_{\partial\Omega}(\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{v}) \\
\int_\Omega \mathbf{v}\cdot\nabla p &= -\int_\Omega p\nabla\cdot\mathbf{v} + \int_\Omega \nabla\cdot(p\mathbf{v}) \\
&= -\int_\Omega p\nabla\cdot\mathbf{v} + \int_{\partial\Omega} p\mathbf{n}\cdot\mathbf{v}.
\end{aligned}$$

Here $\nabla\mathbf{u} : \nabla\mathbf{v}$ represents the componentwise scalar product, for example, in two dimensions $\nabla u_x \cdot \nabla u_y + \nabla v_x \cdot \nabla v_y$. Combining these results and substituting them in (2.4) we obtain

$$\nu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} + \int_\Omega \mathbf{v}\cdot(\mathbf{u}\cdot\nabla\mathbf{u}) - \int_\Omega p\nabla\cdot\mathbf{v} - \int_{\partial\Omega}(\nu\frac{\partial\mathbf{u}}{\partial n} - \mathbf{n}p)\cdot\mathbf{v} = 0,$$

for all $\mathbf{v}$ in a suitable set of test functions. We can see that the boundary term matches the Neumann condition in (2.3). These facts lead to the following velocity and test spaces:

$$\mathbf{H}^1_E := \{\mathbf{u} \in \mathcal{H}^1(\Omega)^d | \mathbf{u} = \mathbf{w} \quad \text{on} \quad \partial\Omega_D\},$$
$$\mathbf{H}^1_{E_0} := \{\mathbf{v} \in \mathcal{H}^1(\Omega)^d | \mathbf{v} = 0 \quad \text{on} \quad \partial\Omega_D\},$$

where $\mathcal{H}^1(\Omega)^d$ denotes the Sobolev space of functions whose generalized derivatives are in $L_2(\Omega)$ and where $d = 2$ or $d = 3$ is the spatial dimension. The fact that there are no pressure derivatives means that $L_2(\Omega)$ is the appropriate space for p. Moreover, choosing the pressure test function q from $L_2(\Omega)$ ensures that the left hand side of (2.5) is finite. The end product is thus the following weak formulation:

Find $\mathbf{u} \in \mathbf{H}^1_E(\Omega)$ and $p \in L_2(\Omega)$ such that

$$\nu \int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} + \int_\Omega (\mathbf{u} \cdot \nabla\mathbf{u}) \cdot \mathbf{v} - \int_\Omega p(\nabla \cdot \mathbf{v}) = \int_{\partial\Omega_N} \mathbf{s} \cdot \mathbf{v} \text{ for all } \mathbf{v} \in \mathbf{H}^1_{E_0} \tag{2.6}$$

$$\int_\Omega q(\nabla \cdot \mathbf{u}) = 0 \text{ for all } q \in L_2(\Omega) \tag{2.7}$$

A discrete weak formulation is defined using finite dimensional spaces $\mathbf{X}^h_0 \subset \mathbf{H}^1_{E_0}$ and $M^h \subset L_2(\Omega)$. Specifically, given a velocity solution space $\mathbf{X}^h_E$, the discrete version of (2.6) and (2.7) is:

Find $\mathbf{u}_h \in \mathbf{X}^h_E$ and $p_h \in M^h$ such that:

$$\nu \int_\Omega \nabla\mathbf{u}_h : \nabla\mathbf{v}_h + \int_\Omega (\mathbf{u}_h \cdot \nabla\mathbf{u}_h) \cdot \mathbf{v}_h - \int_\Omega p_h(\nabla \cdot \mathbf{v}_h) = \int_{\partial\Omega_N} \mathbf{s} \cdot \mathbf{v}_h \text{ for all } \mathbf{v}_h \in \mathbf{X}^h_0 \tag{2.8}$$

$$\int_\Omega q_h(\nabla \cdot \mathbf{u}_h) = 0 \text{ for all } q_h \in M^h. \tag{2.9}$$

Following the steps of the Galerkin method we define two types of basis functions, $\psi_i(x)$ for the pressure and $\phi_i(x)$ for the velocity. So the approximation for $\mathbf{u}_h$ and p_h is defined as

$$p_h = \sum_{j=1}^{n_p} p_j \psi_j(x), \quad n_p \text{ is the number of pressure unknowns} \tag{2.10}$$

and

$$\mathbf{u}_h = \sum_{j=1}^{\frac{n_u}{2}} u_{1j}\phi_{j1}(x) + u_{2j}\phi_{j2}(x) = \sum_{j=1}^{n_u} u_j \phi_j(x) \tag{2.11}$$

where n_u is the number of velocity unknowns, u_j is defined by $u_j = u_{1j}$, for $j = 1, .. \frac{n_u}{2}$, $u_{j+\frac{n_u}{2}} = u_{2j}$, for $j = 1, .. \frac{n_u}{2}$ and ϕ_j in the same way. If we make the substitution $\mathbf{v} = \phi_i(x)$, $q = \psi_i(x)$, we get the standard Galerkin formulation:

Find p_h and $\mathbf{u}_h$, such that

$$\nu \int_\Omega \nabla \mathbf{u}_h : \nabla \phi_i + \int_\Omega (\mathbf{u}_h \cdot \nabla \mathbf{u}_h) \cdot \phi_i - \int_\Omega p_h (\nabla \cdot \phi_i) = \int_{\partial \Omega_N} \mathbf{s} \cdot \phi_i \text{ for all } i = 1, ..n_u, \tag{2.12}$$

$$\int_\Omega \psi_i (\nabla \cdot \mathbf{u}_h) = 0 \text{ for all } i = 1, ...n_p. \tag{2.13}$$

This system of equations can be represented in matrix form as

$$A_d u + N(u) + B^T p = f, \tag{2.14}$$

$$Bu = g, \tag{2.15}$$

where u denotes the vector of unknowns u_{1i} and u_{2i}, and p denotes the vector of unknowns p_i. The term $A_d u$ is the discretization of the viscous term and $N(u)$ is the discretization of the nonlinear convective term, Bu denotes the discretization of the negative divergence of u and $B^T p$ is the discretization of the gradient of p. The right-hand side vectors f and g contain all contributions of the source term, the boundary integral as well as the contribution of the prescribed boundary conditions.

2.2 Linearization

As we can see, the Navier Stokes equations are nonlinear because of the existence of the convective term. The usual approach in order to solve these equations is to solve a linearized version of the equations at each time step. The linearization can be done by Picard or Newton iteration schemes. The *Picard iteration method* gives rise to the so-called Oseen problem:

$$-\nu \Delta u^{k+1} + (u^k \cdot \nabla) u^{k+1} + \nabla p^{k+1} = f \text{ in } \Omega, \tag{2.16}$$

$$\nabla \cdot u^{k+1} = 0 \text{ in } \Omega, \tag{2.17}$$

In this approach, the nonlinear term is substituted by an approximation including the velocity vector calculated at distinct iterations, that is, the convective term at the new time step is defined by

$$u^{k+1} \cdot \nabla u^{k+1} \approx u^k \cdot \nabla u^{k+1}.$$

We have to use an initial guess u^0 for the velocity field in order to construct the approximate solutions (u^{k+1}, p^{k+1}). If we use $u^0 = 0$ we obtain the Stokes problem in the first iteration.

Another approach is the *Newton linearization scheme* which is characterized by assuming that the velocity field at the new iteration is the sum of the velocity field at the previous iteration plus a correction, that is:

$$u^{k+1} = u^k + \delta u^k.$$

If we neglect quadratic terms in δu arising in the convective term, we obtain the *Newton Linearization of the Navier-Stokes equations*:

$$-\nu\Delta u^k + u^k \cdot \nabla u^{k-1} + u^{k-1} \cdot \nabla u^k + \nabla p^k = f - u^{k-1} \cdot \nabla u^{k-1} \text{ in } \Omega, \tag{2.18}$$

$$\nabla \cdot u^k = 0 \text{ in } \Omega. \tag{2.19}$$

After using any type of linearization, the Navier-Stokes system of equations can be written as a linear algebraic system of equations:

$$Fu + B^T p = f, \tag{2.20}$$

$$Bu = g, \tag{2.21}$$

where $F = A_d + N(u^k)$ is the linearized operator and u^k is the solution of the velocity one iteration before. In matrix notation, this system is written as:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{2.22}$$

In general optimization theory, systems of this type arise as the first-order optimality conditions for the following equality-constrained quadratic programming problem:

$$\min : J(x) = \frac{1}{2} u^T A u - f^T u$$

$$\text{subject to: } Bu = g.$$

In this case the variable p represents the vector of Lagrange multipliers. Any solution (u^*, p^*) of (2.22) is a saddle point for the Lagrangian

$$\mathcal{L}(u, p) = \frac{1}{2} u^T A u - f^T u + (Bu - g)^T p,$$

hence the name saddle-point problem given to (2.22). The zero block reflects the absence of the pressure in the continuity equation. As a consequence the system of equations may be under determined for an arbitrary combination of pressure and velocity unknowns [3].

2.3 Finite element selection

The continuity equation, discretized as $Bu = g$, does contain only velocity unknowns. However, the number of rows in this equation is completely determined by the number of pressure unknowns. Suppose that there are more pressure unknowns than velocity unknowns. In that case equations (2.20) and (2.21) contain more rows than columns and we end up with an inconsistent system of equations, that is, the matrix to be solved is singular. Therefore, we have to demand that the number of pressure unknowns never exceeds the number of velocity unknowns. Since we want to solve the Navier-Stokes equations by finite element methods for various grid sizes, this demand should be valid independently

of the number of elements. This demand restricts the number of applicable elements considerably.

In order to satisfy this criterion, a general accepted rule is that the order of approximation of the pressure must be one lower than the order of approximation of the velocity. So if the velocity is approximated by a linear polynomial, then the pressure is approximated by a constant per element and so on. Unfortunately this rule is not sufficient to guarantee that the number of pressure unknowns is not larger than the number of velocity unknowns independently of the number of elements. In the literature, an exact admissibility condition that the finite elements must satisfy is known as the Ladyženskaja-Babuška-Brezzi (LBB) condition. This condition states that, for BB^T in (2.22) to be invertible it is necessary that $\ker(B^T) = 0$, where B^T is a $n_u \times n_p$ matrix. The condition $\ker(B^T) = 0$ means that B^T has rank n_p, and is equivalent to requiring that

$$\max_{\mathbf{v}}(B\mathbf{v}, p) = \max_{\mathbf{v}}(\mathbf{v}, B^T p) > 0, \forall p.$$

This relation in the framework of the finite element method is cast as:

$$\max_{\mathbf{v}\neq\mathbf{0}} \frac{|(q, \nabla \cdot \mathbf{v})|}{\|\mathbf{v}\|_{1,\Omega}\|q\|_{0,\Omega}} > 0 \tag{2.23}$$

Nevertheless, this condition allows the family of matrices to degenerate towards a singular system as $h \to 0$. This calls for a stricter version of the LBB condition which takes the form,

$$\inf_{q\neq\text{constant}} \sup_{\mathbf{v}\neq\mathbf{0}} \frac{|(q, \nabla \cdot \mathbf{v})|}{\|\mathbf{v}\|_{1,\Omega}\|q\|_{0,\Omega}} \geq \gamma > 0. \tag{2.24}$$

However, the LBB condition is rather abstract and in practice it is very difficult to verify whether it is satisfied or not. Fortin in [17] has given a simple method to check the LBB condition on a number of elements based on the following statement:

An element satisfies the BB condition, whenever, given a continuous differentiable vector field u, one can explicitly build a discrete vector field $\hat{u}$ such that:

$$\int_\Omega \Psi_i(\nabla \cdot \hat{u})\, d\Omega = \int_\Omega \Psi_i(\nabla \cdot u)\, d\Omega \quad \text{for all basis functions } \Psi_i.$$

With respect to the types of elements that are applied we make a subdivision into two groups: elements with continuous pressure known as *The Taylor-Hood family* and elements with discontinuous pressure which form *The Crouzeix-Raviart family.*

Chapter 3

Projection techniques, Krylov subspace methods, Preconditioners, and Deflation

The discretization of the Navier-Stokes equations by the Finite Element Method leads to a nonlinear system of equations. The solution process of these equations therefore involves the linearization of such a nonlinear system. Once this linearization is performed, the most general form of the resulting saddle-point problem (2.22) is a matrix equation of the form:

$$Ax = b, \tag{3.1}$$

where A is an $n \times n$, real, nonsymetric, indefinite and typically ill-conditioned matrix. In the present chapter, we present a collection notions in linear algebra which we need to have present while solving such systems.

3.1 Projection techniques

The idea of *projection techniques* is to extract an approximate solution to the above problem from a subspace of $\mathbb{R}^n$. If $\mathcal{K}_m$ is this subspace, usually referred to as the *search subspace* or *ansatz subspace* with dimension m, then, in general, m constraints must be imposed to be able to extract such an approximation. Usually, these constraints are m independent orthogonality conditions. Specifically, the residual vector $b-Ax$ is constrained to be orthogonal to m linearly independent vectors. This defines the so-called *constraints space* $\mathcal{L}_m$ of dimension m. This framework is known as the Petrov-Galerkin conditions [33]. A projection technique onto the subspace $\mathcal{K}_m$ and orthogonal to $\mathcal{L}_m$ is a process that finds an approximate solution $\tilde{x}$ to (3.1) by imposing the conditions that $\tilde{x}$ belong to $\mathcal{K}_m$ and that the new residual vector be orthogonal to $\mathcal{L}_m$. If we can exploit the knowledge of an initial guess x_0 to the solution, then the approximation must be in the affine space

$x_0 + \mathcal{K}_m$ instead of the homogeneous vector space $\mathcal{K}_m$:

$$\text{Find } \tilde{x} \in x_0 + \mathcal{K}_m \text{ such that } b - A\tilde{x} \perp \mathcal{L}_m.$$

Note that if $\tilde{x}$ is written in the form $\tilde{x} = x_0 + \delta$ and the initial residual vector r_0 is defined as $r_0 = b - Ax_0$, then the projection process can be defines as:

$$\tilde{x} = x_0 + \delta, \quad \delta \in \mathcal{K}_m, \tag{3.2}$$

$$(r_0 - A\delta, w) = 0 \quad \forall w \in \mathcal{L}_m. \tag{3.3}$$

Let $V = [v_1, ..., v_m]$ be an $n \times m$ matrix whose column vectors form a basis of $\mathcal{K}_m$ and similarly, $W = [w_1, ..., w_m]$ be an $n \times m$ matrix whose column vectors form a basis of $\mathcal{L}_m$. If the approximate solution is written as

$$\tilde{x} = x_0 + Vy, \tag{3.4}$$

then the orthogonality condition leads to the following system of equations for the vector y:

$$W^T AVy = W^T r_0. \tag{3.5}$$

If the assumption is made that the $m \times m$ matrix $W^T AV$ is nonsingular, the following expression for the approximate solution $\tilde{x}$ results:

$$\tilde{x} = x_0 + V(W^T AV)^{-1} W^T r_0. \tag{3.6}$$

It is important to note that the approximate solution is defined only when the matrix $W^T AV$ is nonsingular, a property that is not guaranteed to be true even when A is nonsingular. Nevertheless, it can be verified that the projection method is well defined, that is, $W^T AV$ is nonsingular in three particular cases.

Theorem: *Let A, $\mathcal{L}$, and $\mathcal{K}$ satisfy either of the following three conditions:*

1. *A is Hermitian positive definite (HPD) and $\mathcal{L} = \mathcal{K}$, or*
2. *A is Hermitian and invertible and $\mathcal{L} = A\mathcal{K}$, or*
3. A is invertible and $\mathcal{L} = A\mathcal{K}$.

The proof can be found in [33]. Moreover, in these cases, the result of the projection process can be interpreted easily in terms of actions of orthogonal projectors on the initial residual or the initial error. If we consider the cases in which $\mathcal{L} = A\mathcal{K}$, and let r_0 be the initial residual $r_0 = b - Ax_0$ and $\tilde{r} = b - A\tilde{x}$ the residual obtained after the projection process. Then

$$\tilde{r} = b - A(x_0 - \delta) = r_0 - A - \delta. \tag{3.7}$$

In addition, δ is obtained by enforcing the condition that $r_0 - A\delta$ be orthogonal to $A\mathcal{K}$. Therefore, the vector $A\delta$ is *the orthogonal projection of the vector* r_0 *onto the subspace* $A\mathcal{K}$. Hence the following proposition can be stated.

Proposition: *Let* $\tilde{x}$*, be the approximate solution obtained from a projection method onto* $\mathcal{K}$ *orthogonally to* $\mathcal{L} = A\mathcal{K}$ *and let* $\tilde{r} = b - A\tilde{x}$ *be the associated residual. Then*

$$\tilde{r} = (I - P)r_0, \tag{3.8}$$

where P *denotes the orthogonal projector onto the subspace* $A\mathcal{K}$*.*

From this proposition it follows that the 2-norm of the residual vector obtained after one projection step will not exceed the initial 2-norm of the residual; i.e.,

$$\|\tilde{r}\|_2 \leq \|r_0\|_2.$$

This class of methods are known as *residual projection* methods.
Now, if we consider the case where $\mathcal{L} = \mathcal{K}$ and A is HPD and let the initial error be denoted by $d_0 = x_* - x_0$, where x_* denotes the exact solution to the system, and, similarly, let $\tilde{d} = x_* - \tilde{x}$, where $\tilde{x} = x_0 + \delta$ is the approximate solution resulting from the projection step. Then (3.7) yields the relation

$$A\tilde{d} = \tilde{r} = A(d_0 - \delta),$$

where δ is now obtained by constraining the residual vector $r_0 - A\delta$ to be orthogonal to $\mathcal{K}$:

$$(r_0 - A\delta, w) = 0 \quad \forall w \in \mathcal{K}.$$

The above condition is equivalent to

$$(A(d_0 - \delta), w) = 0 \quad \forall w \in \mathcal{K}.$$

Since A is SPD, it defines an inner product, which is usually denoted by $(\cdot, \cdot)_A$, and the above condition becomes

$$(d_0 - \delta, w)_A = 0 \quad \forall w \in \mathcal{K}.$$

The above condition is now easy to interpret: *The vector* δ *is the* A*-orthogonal projection of the initial error* d_0 *onto the subspace* $\mathcal{K}$*.*

Proposition: *Let* $\tilde{x}$*, be the approximate solution obtained from a projection method onto* $\mathcal{K}$ *and let* $\tilde{d} = x_* - \tilde{x}$ *be the associated error vector. Then*

$$\tilde{d} = (I - P_A)d_0, \tag{3.9}$$

where P_A denotes the projector onto the subspace $\mathcal{K}$, which is orthogonal with respect to the A inner product.

A result of the proposition is that the A-norm of the error vector obtained after one projection step does not exceed the initial A-norm of the error; i.e.,

$$\|\tilde{d}\|_A \leq \|d_0\|_A,$$

which is expected because it is known that the A-norm of the error is minimized in $x_0 + \mathcal{K}$. This class of methods are known as *error projection methods.*

3.2 Krylov subspace methods

A Krylov subspace method is a projection method for which the subspace $\mathcal{K}_m$ is the Krylov subspace:

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, ..., A^{m-1} r_0\}. \tag{3.10}$$

The different versions of Krylov subspace methods arise from different choices of the subspace $\mathcal{L}_m$ and from the ways in which the system is preconditioned. Arnoldi's procedure is an algorithm for building an orthogonal basis of the Krylov subspace $\mathcal{K}_m$. One variant of the algorithm known as the Modified Gram-Schmidt (MGS) algorithm is as follows:

Algorithm 3.1 Arnoldi with Modified Gram Schmidt

1: Chose a vector v_1 such that $\|v_1\|_2 = 1$
2: **for** $j = 1$ to m **do**
3: Compute $w_j := Av_j$
4: **for** $i = 1$ to j **do**
5: $h_{ij} = (w_j, v_i)$
6: $w_j := w_j - h_{ij} v_i$
7: **end for**
8: $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ Stop
9: $v_{j+1} = w_j / h_{j+1,j}$
10: **end for**

The general procedure to form the orthonormal basis is as follows: assume we have an orthonormal basis $[v_1, ..., v_j]$ for $\mathcal{K}_j(A, r_0)$. This basis is expanded by computing $w = Av_j$ and orthonormalized with respect to the previous basis. Let the matrix V_j be given as

$$V_j = [v_1, ..., v_j], \ \textit{where } \text{span}(v_1, ..., v_j) = \mathcal{K}_j$$

Since the columns of V_j are orthogonal to each other. It follows that

$$AV_j = V_j H_j + w_j e_j^T \tag{3.11}$$

$$= V_{j+1} \bar{H}_j, \tag{3.12}$$

$$V_j^T A V_j = H_j \tag{3.13}$$

The $j \times j$ matrix H_j is upper Hessenberg, and its elements $h_{i,j}$ are defined by the algorithm. If A is symmetric, then $H_j = V_j^T A V_j$ is also symmetric and thus tridiagonal. This leads to a three term recurrence in the Arnoldi process. Each new vector has only to be orthogonalized with respect to two previous vectors. This process is called the Lanczos algorithm.

3.2.1 Using Arnoldi's Method for solving Linear Systems

Given an initial guess x_0 to the original linear system $Ax = b$, we now consider an orthogonal projection method, which takes $\mathcal{L} = \mathcal{K} = \mathcal{K}_m(A, r_0)$, with $\mathcal{K}_m(A, r_0)$ given by (3.10) in which $r_0 = b - Ax_0$. This method seeks an approximate solution x_m from the affine space $x_0 + \mathcal{K}_m$ of dimension m by imposing the Galerkin condition $b - Ax_m \perp \mathcal{K}_m$. If $v_1 = r_0/\|r_0\|_2$ in Arnoldi's method and we set $\beta = \|r_0\|_2$, then

$$V_m^T A V_m = H_m \tag{3.14}$$

as a consequence of equation (3.13), and

$$V_m^T r_0 = V_m^T(\beta v_1) = \beta e_1. \tag{3.15}$$

As a result, the approximate solution using the above m-dimensional subspace is given by

$$x_m = x_0 + V_m y_m \tag{3.16}$$

$$y_m = H_m^{-1}(\beta e_1). \tag{3.17}$$

A method based on this approach is called the full orthogonalization method (FOM), presented in [33].

The generalized minimal residual method (`GMRES`) is a projection method based on taking $\mathcal{K} = \mathcal{K}_m$ and $\mathcal{L} = A\mathcal{K}_m$, in which $\mathcal{K}_m$ is the m-th Krylov subspace, with $\|v_1\| = r_0/\|r_0\|_2$. As seen previously, such a technique minimizes the residual norm over all vectors in $x_0 + \mathcal{K}_m$. The implementation of an algorithm based on this approach is similar to that of the FOM algorithm.
We will derive the algorithm exploiting the optimality condition as well as relation (3.12). Any vector x in $x_0 + \mathcal{K}_m$ can be written as

$$x = x_0 + V_m y, \tag{3.18}$$

where y is an m-vector. Defining

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2, \tag{3.19}$$

the relation (3.12) results in

$$\begin{aligned} b - Ax &= b - A(x_0 + V_m y) \\ &= r_0 - AV_m y \\ &= \beta v_1 - V_{m+1}\bar{H}_m y \\ &= V_{m+1}(\beta e_1 - \bar{H}_m y). \end{aligned}$$

Since the column vectors of V_{m+1} are orthonormal, then

$$J(y) \equiv \|b - A(x_0 + V_m y)\|_2 = \|\beta e_1 - \bar{H}_m y\|_2. \tag{3.20}$$

The `GMRES` approximation is the unique vector of $x_0 + \mathcal{K}_m$ that minimizes (3.19). By (3.18) and (3.20), this approximation can be obtained quite simply as $x_m = x_0 + V_m y_m$, where y_m minimizes the function $J(y) = \|\beta e_1 - \bar{H}_m y\|_2$; i.e.

$$x_m = x_0 + V_m y_m, \text{ where} \tag{3.21}$$

$$y_m = \min_y \|\beta e_1 - \bar{H}_m y\|_2. \tag{3.22}$$

The minimizer y_m is inexpensive to compute since it requires the solution of an $(m+1) \times m$ least-squares problem, where m is typically small. This gives the following algorithm.

Algorithm 3.2 GMRES

1: Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2: **for** $j = 1$ to m **do**
3: Compute $w_j := Av_j$
4: **for** $i = 1$ to j **do**
5: $h_{ij} = (w_j, v_i)$
6: $w_j := w_j - h_{ij} v_i$
7: **end for**
8: $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ set $m := j$ and go to 11
9: $v_{j+1} = w_j / h_{j+1,j}$
10: **end for**
11: Define the $(m+1) \times m$ Hessenberg matrix $\hat{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$
12: Compute y_m, the minimizer of $\|\beta e_1 - \hat{H}_m y\|_2$, and $x_m = x_0 + V_m y_m$

All Krylov subspace methods are related to, as well as defined by, the choice of a basis of the Krylov subspace. The `GMRES` algorithm uses an orthonormal basis. In the `CG` algorithm, the p's are A orthogonal, i.e. conjugate, and so forth. A number of algorithms can be developed using a basis of this form in the nonsymetric case as well. The main result that is exploited in all these algorithms is the following lemma.

Lemma: *Let $p_0, p_1, ..., p_{m-1}$ be a sequence of orthonormal vectors such that each set $\{p_0, p_1, ..., p_{j-1}\}$ for $j \le m$ is a basis of the Krylov subspace $\mathcal{K}_j(A, r_0)$, which is $A^T A$-*

orthogonal, i.e., such that

$$(Ap_i, Ap_k) = 0, \quad \text{for } i \neq k. \tag{3.23}$$

Then the approximate solution x_m that has the smallest residual norm in the affine space $x_0 + \mathcal{K}_m(A, r_0)$is given by

$$x_m = x_0 + \sum_{i=0}^{m-1} \frac{(r_0, Ap_i)}{(Ap_i, Ap_i)} p_i. \tag{3.24}$$

In addition, x_m can be computed from x_{m-1} by

$$x_m = x_{m-1} + \frac{(r_{m-1}, Ap_{m-1})}{(Ap_{m-1}, Ap_{m-1})} p_{m-1}. \tag{3.25}$$

This lemma opens up many different ways to obtain algorithms that are mathematically equivalent to full `GMRES`. The simplest option computes the next basis vector p_{m+1} as a linear combination of the current residual r_m and all previous p_i's. The approximate solution is updated by using (3.25). This is called the generalized conjugate residual (`GCR`) algorithm.

Algorithm 3.3 GCR

1: Compute $r_0 = b - Ax_0$. Set $s_1 = r_0$
2: **for** $i = 1...$ until convergence **do**
3: Set $s_i = r_{i-1}$
4: Compute $v_i = As_i$
5: **for** $j = 1$ to $i - 1$ **do**
6: $\alpha = (v_j, v_i)$
7: $s_i := s_i - \alpha s_j$, $v_i := v_i - \alpha v_j$
8: **end for**
9: $s_i = \frac{s_i}{\|s_i\|_2}$, $v_i = \frac{v_i}{\|v_i\|_2}$
10: $\beta = (v_i, r_{i-1})$
11: $u_i := u_{i-1} + \beta s_i$
12: $r_i := r_{i-1} - \beta v_i$
13: **end for**

Both the set of s_i's and the set of v_i's need to be saved. This doubles the storage requirement compared to `GMRES`. The number of arithmetic operations per step is also roughly 50% higher than that with `GMRES` [33].

3.3 Preconditioners

Preconditioning is a key ingredient for the success of Krylov subspace methods. Preconditioning is a means of transforming the original linear system into one with the same solution, but that is cheaper to solve with an iterative solver. In general, the reliability of iterative techniques depends much more on the quality of the preconditioner than on the

particular Krylov subspace accelerators used [33].

The first step in preconditioning is to find a preconditioning matrix M. The matrix M should satisfy a few requirements, the most important being that it must be inexpensive to solve linear systems $Mx = b$. This is because the preconditioned algorithms require a linear system solution with the matrix M at each iteration. Also, M should be close to A in some sense and it should clearly be nonsingular. Given the matrix splitting

$$A = M - N, \tag{3.26}$$

where A is associated with the linear system (3.1). A *linear fixed-point iteration* can be defined by the recurrence

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b, \tag{3.27}$$

which at the same time is of the form

$$x_{k+1} = Gx_k + f, \tag{3.28}$$

with

$$G = M^{-1}N = M^{-1}(M - A) = I - M^{-1}A, \quad f = M^{-1}b$$

The iteration (3.28) can be viewed as a technique for solving the system

$$(I - G)x = f. \tag{3.29}$$

Since G has the form $G = I - M^{-1}A$, this system can be rewritten as

$$M^{-1}Ax = M^{-1}b. \tag{3.30}$$

This system, which has the same solution as the original system, is called a *preconditioned system* and M is the *preconditioning matrix* or *preconditioner*. In other words, *a relaxation scheme is equivalent to a fixed-point iteration on a preconditioned system.* Once a preconditioning matrix M is available there are three known ways of applying it. The preconditioner can be applied from the left, leading to the preconditioned system

$$M^{-1}Ax = M^{-1}b. \tag{3.31}$$

Alternatively, it can also be applied to the right:

$$AM^{-1}u = b, \quad x \equiv M^{-1}u. \tag{3.32}$$

Note that the above formulation amounts to making the change of variables $u = Mx$ and solving the system with respect to the unknown u. Finally, a common situation is when the preconditioner is available is the factored form

$$M = M_L M_R$$

where, typically, M_L and M_R come from an ILU factorization. In this situation, the preconditioning can be split:

$$M_L^{-1} A M_R^{-1} u = M_L^{-1} b, \quad x \equiv M_R^{-1} u. \tag{3.33}$$

It is of utmost importance to preserve symmetry whenever the original matrix is symmetric. The straightforward way of preserving symmetry is by applying the method described by (3.33) however symmetry can also be preserved even when the preconditioned matrix is not available in factored form. If we observe that $M^{-1}A$ is self-adjoint for the M-inner product:

$$(x, y)_M \equiv (Mx, y) = (x, My),$$

since

$$(M^{-1}Ax, y)_M = (Ax, y) = (x, Ay) = (x, M(M^{-1}A)y) = (x, M^{-1}Ay)_M. \tag{3.34}$$

We can exploit this fact in order to precondition Algorithm 3.2. An alternative is to replace the usual Euclidean inner product in the `GMRES` algorithm with the M inner product. In this particular case of nonsymetric iterative solvers, the three options for applying the preconditioning operation are available, namely left, split, and right preconditioning. However, the right preconditioning versions can give rise to what is called a flexible variant - a variant in which the preconditioner can change at each step. The right-preconditioned `GMRES` algorithm is based on solving

$$AM^{-1}u = b, \quad u = Mx. \tag{3.35}$$

As we now show, the new variable u never needs to be invoked explicitly. Indeed, once the initial residual $b - Ax_0 = b - AM^{-1}u_0$ is computed, all subsequent vectors of the Krylov subspace can be obtained without any reference to the u variables. Note that u_0 is not needed at all. The initial residual for the preconditioned system can be computed from $r_0 = b - Ax_0$, which is the same as $b - AM^{-1}u_0$. In practice, it is usually x_0 that is available, not u_0. At the end, the u variable approximate solution to (3.35) is given by

$$u_m = u_0 + \sum_{i=1}^{m} v_i \eta_i,$$

with $u_0 = Mx_0$. Multiplying through by M^{-1} yields the desired approximation in terms of the x variable:

$$x_m = x_0 + M^{-1} \left[\sum_{i=1}^{m} v_i \eta_i \right].$$

Thus, one preconditioning operation is needed at the end of the outer loop, instead of at the beginning which is the case for the left-preconditioned version.

Algorithm 3.4 GMRES with Right Preconditioning

1: Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2: **for** $j = 1$ to m **do**
3: Compute $w := AM^{-1}v_j$
4: **for** $i = 1$ to j **do**
5: $h_{ij} = (w_j, v_i)$
6: $w_j := w_j - h_{ij}v_i$
7: **end for**
8: Compute $h_{j+1,j} = \|w_j\|_2$ and $v_{j+1} = w/h_{j+1,j}$
9: Define $V_m := [v_1, ..., v_m]$, $\bar{H}_m = \{h_{i,j}\}_{1 \le i \le m+1, 1 \le j \le m}$
10: **end for**
11: Compute $y_m = \min_y \|\beta e_1 - \hat{H}_m y\|_2$, and $x_m = x_0 + M^{-1}V_m y_m$
12: If satisfied Stop, else set $x_0 := x_m$ and go to 1.

This time, the Arnoldi loop builds an orthogonal basis of the right-preconditioned Krylov subspace

$$\text{span}\{r_0, AM^{-1}r_0, ..., (AM^{-1})^{m-1}r_0\}.$$

Note that the residual norm is now relative to the initial system, i.e., $b - Ax_m$, since the algorithm obtains the residual $b - Ax_m = b - AM^{-1}u_m$ implicitly.

So far, it has been implicitly assumed that the preconditioning matrix M is constant; i.e., it does not change from step to step. However, in some cases no matrix M is available. Instead, the operation $M^{-1}x$ is the result of some unspecified computation, possibly another iterative process. In such cases, it may happen that M^{-1} is not a constant operator. The previous preconditioned iterative procedures will not converge if M is not constant. There are a number of variants that allow variations in the preconditioner from iteration to iteration. One of these variants of the `GMRES` algorithm is described next.
In line 11 of the `GMRES` with Right Preconditioning algorithm the approximate solution x_m is expressed as a linear combination of the preconditioned vectors $z_i = M^{-1}v_i$, $i = 1, ..., m$. These vectors are also computed in line 3, prior to their multiplication by A to obtain the vector w. They are all obtained by applying the same preconditioning matrix M^{-1} to the v_i's. As a result it is not necessary to save them. Instead, we only need to apply M^{-1} to the linear combination of the v_i's, that is to $V_m y_m$ in line 11.
Suppose now that the preconditioner could change at every step, i.e., that z_j is given by

$$z_j = M_j^{-1}v_j.$$

Then it would be natural to compute the approximate solution as

$$x_m = x_0 + Z_m y_m,$$

in which $Z_m = [z_1, ..., z_m]$ and y_m is computed as before, as the solution to the least-squares problem in line 11. These are the only changes that lead from the right-preconditioned algorithm to the flexible variant, described below.

Algorithm 3.5 Flexible GMRES

1: Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2: **for** $j = 1$ to m **do**
3: Compute $z_j := M_j^{-1} v_j$
4: Compute $w := Az_j$
5: **for** $i = 1$ to j **do**
6: $h_{i,j} = (w, v_i)$
7: $w := w - h_{i,j} v_i$
8: **end for**
9: Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
10: Define $Z_m := [z_1, ..., z_m]$, $\bar{H}_m = \{h_{i,j}\}_{1 \le i \le m+1, 1 \le j \le m}$
11: **end for**
12: Compute $y_m = \min_y \|\beta e_1 - \hat{H}_m y\|_2$, and $x_m = x_0 + Z_m y_m$
13: If satisfied Stop, else set $x_0 \leftarrow x_m$ and go to 1..

As can be seen, the main difference with the right-preconditioned version is that the preconditioned vectors $z_j = M_j^{-1} v_j$ must be saved and the solution updated using these vectors. It is clear that when $M_j = M$ for $j = 1, ..., m$, then this method is equivalent mathematically to `GMRES` with right preconditioning. It is important to observe that z_j can be defined in line 3 without reference to any preconditioner. That is, any given new vector z_j can be chosen. This added flexibility may cause the algorithm some problems. Indeed, z_j may be so poorly chosen that a breakdown may occur, as in the worst case scenario when z_j is zero.

3.4 Deflation

After preconditioning, the resulting linear system to be considered is

$$AM^{-1}u = b, \quad x = M^{-1}u, \tag{3.36}$$

where M^{-1} is a preconditioner. the spectrum of the preconditioned matrix, $\sigma(AM^{-1})$, often contains unfavorable eigenvalues that deteriorate the convergence of the iterative solver. The so-called deflation method effectively treats these eigenvalues so that the convergence of the iterative method can be significantly improved. (See [37], [18]) In order to describe the deflation method we start with some preliminaries:

Definition: *Let A be an SPSD matrix. Suppose that $Z \in \mathbb{R}^{n \times k}$ with full rank is given. Then we define the invertible Galerkin matrix, $E \in \mathbb{R}^{k \times k}$, the correction matrix, $Q \in$*

$\mathbb{R}^{n\times n}$, and the deflation matrix $P \in \mathbb{R}^{n\times n}$, as follows:

$$P = I - AQ, \quad Q = ZE^{-1}Z^T, \quad E = Z^TAZ \tag{3.37}$$

In Equation (3.37), Z is the so-called 'deflation subspace matrix' whose k columns are called the 'deflation vectors' or 'projection vectors'. These vectors remain unspecified for the moment, but they are chosen in such a way that E is nonsingular. Using (3.37) we next state some useful results:

Lemma: *Let A, Z, E, Q, and P be given as before. Let x and b be the solution and right-hand side of* (3.36)*, respectively. Then the following equalities hold:*

(a) $E^T = E$

(b) $Q^T = Q = QAQ$

(c) $QAZ = Z$

(d) $PAQ = \mathbf{0}$

(e) $P^2 = P$

(f) $AP^T = PA$

(g) $(I - P^T)x = Qb$

These results, specifically result (e) shows that P is a projector. This means that it divides the space $\mathbb{R}^n$ in a direct sum of two subspaces. This tells us that the original linear system $Ax = b$ can be solved by employing the splitting

$$x = (I - P^T)x + P^Tx. \tag{3.38}$$

In Equation (3.38), $(I - P^T)x$ can be computed immediately from the Lemma of this section part (g). Hence only P^Tx should be computed in (3.38) in order to find x. We can write

$$\begin{aligned} x = (I - P^T)x + P^Tx \Leftrightarrow \quad & x = Qb + P^Tx && (3.39)\\ \Leftrightarrow \quad & Ax = AQb + AP^Tx && (3.40)\\ \Leftrightarrow \quad & b = AQb + PAx && (3.41)\\ \Leftrightarrow \quad & Pb = PAx && (3.42) \end{aligned}$$

where we have used part (f) of the Lemma. Note that x at the end of Equation (3.39) is not necessarily a solution of the original linear system $Ax = b$, since it may consists of components of the null space of PA. Therefore, this 'deflated' solution is denoted as $\hat{x}$ rather than x. We can now solve the deflated system

$$PA\hat{x} = Pb, \tag{3.43}$$

using an iterative solver. The relation between the solutions x and $\hat{x}$ can be obtained from the following Corollary:

Corollary: *Let P and Q be given as before. Let b be the right-hand side of $Ax = b$. Then, the solution x of $Ax = b$ can be expressed as:*

$$x = Qb + P^T\hat{x}, \tag{3.44}$$

where $\hat{x}$ is a solution of (3.43).

Since PA preserves the properties of the original matrix, this can be interpreted as the new coefficient matrix of the linear system.

The deflated system (3.43) can also be solved using a preconditioner, M^{-1}. In this case, we solve:

$$\tilde{P}\tilde{A}\hat{\tilde{x}} = \tilde{P}\tilde{b} \tag{3.45}$$

with

$$\tilde{A} = AM^{-1}, \quad \hat{\tilde{x}} = \hat{x}M^{-1}, \quad \tilde{b} = b,$$

and

$$\tilde{P} = I - \tilde{A}\tilde{Q}, \quad \tilde{Q} = \tilde{Z}\tilde{E}^{-1}\tilde{Z}^T, \quad \tilde{E} = \tilde{Z}^T\tilde{A}\tilde{Z},$$

where $\tilde{Z} \in \mathbb{R}^{n\times k}$ can be interpreted as a preconditioned deflation-subspace matrix. The resulting method will be a deflated and preconditioned iterative method.
Eigenvectors corresponding to the smallest nonzero eigenvalues of A are usually used as deflation vectors since M^{-1} often treats the largest eigenvalues of A effectively. In this case, the deflation method is fast in convergence if P acts as a complementary part of the preconditioning by projecting the smallest eigenvalues to zero. However, in general, eigenvectors associated with the largest eigenvalues of A, or a combination of these two approaches, can also be used as deflation vectors in order to reduce $\kappa(M^{-1}PA)$. This approach is called 'eigenvector deflation' and can be very effective, but, unfortunately, eigenvectors are usually expensive to compute in practice. In addition, eigenvectors are often dense, leading to a possibly expensive deflation matrix P. Ideally, Z should consist of sparse and good approximations of eigenvectors.

Chapter 4

Block-type preconditioners for the incompressible Navier-Stokes equations

Lack of robustness is a widely recognized weakness of iterative solvers relative to direct solvers. This drawback hampers the acceptance of iterative methods in industrial applications despite their intrinsic appeal for very large linear systems. Both the efficiency and robustness of iterative techniques can be improved by using preconditioning. A term introduced in Chapter 3, preconditioning is simply a means of transforming the original linear system into one which has the same solution, but which is likely to be cheaper to solve with an iterative solver. In general, the reliability of iterative techniques, when dealing with various applications, depends much more on the quality of the preconditioner than on the particular Krylov subspace accelerators used.

4.1 Block preconditioners

One particular class of preconditioners is known as *Block Preconditioners*. These type of preconditioners are based on a block factorization of the coefficient matrix (2.22). After the factorization is performed, two subsystems for the velocity and pressure are solved separately during each iteration. The general approach of such separation is known as the *Schur Complement method*, which can be given as follows.
Consider a block factorized linear system written in the form:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{4.1}$$

in which A_{11} is assumed to be nonsingular. From the first equation the unknown x can be expressed as

$$x = A_{11}^{-1}(f - A_{12}y). \tag{4.2}$$

If we substitute this into the second equation, the following *reduced system* is obtained:

$$(A_{22} - A_{21}A_{11}^{-1}A_{12})y = g - A_{21}A_{11}^{-1}f. \tag{4.3}$$

The matrix

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12} \tag{4.4}$$

is called the *Schur complement* matrix associated with the y variable. If this matrix can be formed and the linear system (4.1) can be solved, all the interface variables y, that is the variables that couple both systems, will become available. Once these variables are known, the remaining unknowns can be computed via (4.2). Due to this particular fact, an important aspect of Block preconditioners is to have a good approximation of the Schur complement matrix.

In the context of the Navier-Stokes equations, Block preconditioners are based on a block factorization of the coefficient matrix (2.22). They are mostly based on a block LDU factorization of (2.22):

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = LDU = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{4.5}$$

where S is the Schur complement matrix discussed above. Similarly, Block triangular preconditioners (P_t) are based on the block DU factorization of (2.22) given by:

$$DU = \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} = P_t = \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \tag{4.6}$$

By investigating the following generalized eigenvalue problem, we can determine the eigenvalues of the preconditioned system:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} \tag{4.7}$$

We can see by inspecting the first row of (4.7) that,

$$(1 - \lambda)\left(Fu + B^T p\right) = 0.$$

This is only possible if $(1 - \lambda) = 0$ or $\left(Fu + B^T p\right) = 0$.

In the case $(1 - \lambda) = 0$ we thus have $\lambda = 1$ signifying that we have n_u eigenvalues equal to one, that is we have eigenvalues equal to 1 of multiplicity n_u. For the case $\left(Fu + B^T p\right) = 0$ we obtain:

$$\left(Fu + B^T p\right) = 0 \Rightarrow u = -F^{-1}B^T p \tag{4.8}$$

From the second row of (4.7) we obtain:

$$Bu - \lambda Sp = 0,$$

If we now substitute $u = -F^{-1}B^T p$ on the previous equation, we obtain:

$$-BF^{-1}B^T p = \lambda Sp. \tag{4.9}$$

This shows that whenever $S = -BF^{-1}B^T$ we have $\lambda = 1$ with multiplicity n_p. From this equation, we can see that a good approximation of the Schur complement matrix will dictate the convergence behavior of the preconditioned system with P_t. A better approximation of the Schur complement matrix will cluster the eigenvalues close to one thus causing a faster convergence. Moreover, the use of F^{-1} and S^{-1} is not practical due to the expensive calculation and storage of such matrices. In general, F^{-1} is approximated by a matrix $\hat{F}^{-1}$ obtained by a small number of iterations with an iterative method. Thus, the use of Block triangular preconditioners (4.6) involves the solution of $P_t z = r$, where $z = \begin{bmatrix} z_u \\ z_p \end{bmatrix}$ and $r = \begin{bmatrix} r_u \\ r_p \end{bmatrix}$ as given by the next Algorithm:

Algorithm 4.1 Preconditioner P_t

1: Solve $Sz_p = r_p$
2: Update $r_u = r_u - B^T z_p$
3: Solve $Fz_u = r_u$

We can see that the preconditioner involves the solution of two subproblems, one associated with the pressure part and the other with the velocity part of the problem. As we have mentioned before, the Schur complement matrix is not formed, but approximated by a simple matrix $\hat{S}$. The approximate inverse $\hat{S}^{-1}$ is replaced by a simple spectral equivalent matrix such that the preconditioned matrix has a tightly clustered spectrum. How this approximation is done defines the various block preconditioners.

4.2 Block preconditioners based on approximate commutators

Two popular preconditioners are based on approximating the commutator of the convection diffusion operator with the gradient operator. The commutator of two operators x and y is defined as

$$[x, y] = xy - yx.$$

And whenever $[x, y] = 0$ it is said that the operator x commutes with the operator y, that is $xy = yx$. The convection diffusion operator (See [21]) defined on the velocity space can be expressed as:

$$\mathcal{L} = -\nu\nabla^2 + \mathbf{w_h} \cdot \nabla \tag{4.10}$$

where $\mathbf{w_h}$ is the computed approximation to the discrete velocity at the most recent iteration.

4.2.1 Pressure convection-diffusion preconditioner

Based on the idea presented by Kay *et al.* [21] that the commutator of the convection diffusion operator acting on the gradient operator, on the velocity space, and the gradient operator acting on the convection diffusion operator on the pressure space ($\mathcal{L}_p$) is small, that is:

$$\varepsilon = \mathcal{L}\nabla - \nabla\mathcal{L}_p \ll 1, \tag{4.11}$$

then the discrete commutator in terms of finite element matrices given as:

$$\varepsilon_h = (Q_v^{-1}F)(Q_v^{-1}B^T) - (Q_v^{-1}B^T)(Q_p^{-1}F_p) \tag{4.12}$$

might also be small. Q_v denotes the velocity mass matrix and Q_p the pressure mass matrix (scaling matrices). F_p is a discrete convection diffusion operator on pressure space. The multiplication by Q_u^{-1} and Q_p^{-1} transforms quantities from integrated values to nodal values. If we now pre-multiply (4.12) by $BF^{-1}Q_v$, and post-multiplicate by $F_p^{-1}Q_p$ and assuming that the commutator is small, leads to an approximation to the Schur complement matrix:

$$BF^{-1}B^T \approx BQ_v^{-1}B^TF_p^{-1}Q_p. \tag{4.13}$$

in which the expensive part $BQ_v^{-1}B^T$ is replaced by its spectral equivalent matrix A_p known as the pressure Laplacian matrix, that is:

$$S = -BF^{-1}B^T \approx -A_pF_p^{-1}Q_p \tag{4.14}$$

The preconditioner (4.6) with the approximation given in (4.14) is known as the so called pressure convection-diffusion (PCD) preconditioner.
The convergence of this preconditioner combined with a Krylov method is very good for enclosed flows if the equations are linearized by the Picard method [35]. The preconditioner gives rise to many iterations in inflow/outflow problems, the reason could be that an approximation of $BQ_v^{-1}B^T$ by A_p is well-defined only for enclosed flow problems [39]. Boundary conditions are treated such that A_p and F_p are computed with Neumann boundary conditions for an enclosed flow problem. However in inflow/outflow problems, rows and columns of A_p and F_p corresponding to pressure nodes on an inflow boundary are treated as though they are associated with Dirichlet boundary conditions [39]. One of the main disadvantages of PCD is the necessity to construct the matrices A_p and F_p and the definition of boundary conditions for the pressure matrix. This makes implementation in standard finite element codes less obvious [35].

4.2.2 Least squares commutator preconditioner

Instead of building two extra operators F_p and A_p in PCD, Elman *et al.* devised another approach for approximating the Schur complement matrix known as the least squares

commutator (LSC) preconditioner [12].

The idea is to approximate the matrix operator F_p in (4.13) such that the discrete commutator (4.12) becomes small. This is done by solving a least squares problem. For the j-th column of the matrix F_p, the least squares problem has the form:

$$\min \|[Q_v^{-1}FQ_v^{-1}B^T]_j - Q_v^{-1}B^TQ_p^{-1}[F_p]_j\|_{Q_v}, \tag{4.15}$$

where $\|\cdot\|_{Q_v}$ is the $\sqrt{x^TQ_vx}$ norm. The normal equations associated with this problem are:

$$Q_p^{-1}BQ_v^{-1}B^TQ_p^{-1}[F_p]_j = [Q_p^{-1}BQ_v^{-1}FQ_v^{-1}B^T]_j, \tag{4.16}$$

which leads to the following definition of F_p:

$$F_p = Q_p(BQ_v^{-1}B^T)^{-1}(BQ_v^{-1}FQ_v^{-1}B^T). \tag{4.17}$$

Substituting this expression into (4.13) provides an approximation to the Schur complement matrix:

$$S = BF^{-1}B^T \approx (BQ_v^{-1}B^T)(BQ_v^{-1}FQ_v^{-1}B^T)^{-1}(BQ_v^{-1}B^T). \tag{4.18}$$

The preconditioner based on this approximation is known as the LSC preconditioner. Generally, the inverse of the velocity mass matrix Q_v^{-1} is dense. The preconditioner is expensive if the full velocity mass matrix is used in the preconditioner. Therefore, Q_v is replaced by $\hat{Q}_v$, the diagonal of the velocity mass matrix. In the LSC preconditioner, the first three steps are used to solve the approximate Schur complement (4.18). If we denote the residual of a Krylov subspace method by $r = \begin{bmatrix} r_v \\ r_p \end{bmatrix}$, where r_v and r_p refer to the velocity and pressure part, respectively. The preconditioning steps with the LSC preconditioner are given by:

Algorithm 4.2 Least Squares Commutator Preconditioner

1: Solve $S_f z_p = r_p$ where $S_f = B\hat{Q}_v^{-1}B^T$
2: Update $r_p = B\hat{Q}_v^{-1}F\hat{Q}_v^{-1}B^Tz_p$
3: Solve $S_f z_p = -r_p$
4: Update $r_u = r_u - B^Tz_p$
5: Solve $Fz_u = r_u$

The LSC preconditioner is built from readily available matrices and no extra boundary conditions are needed, however, per iteration LSC is more expensive than PCD since it requires two Poisson solves instead of one, whereas PCD requires two extra operators F_p and A_p on the pressure space including some boundary conditions. Nevertheless, its convergence is better provoking that in the literature it is concluded that LSC is faster than PCD [43].

4.3 Augmented Lagrangian approach

A completely different approach has been published by Benzi and Olshanskii [**?**]. In this method, it is necessary to augment the velocity matrix in the original equation by a penalty-like term $\gamma B^T W^{-1} B$ with γ relatively small and W a scaling matrix, usually the diagonal of the pressure matrix [35].
The system of equations (2.22) is replaced by

$$\begin{bmatrix} F_\gamma & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{4.19}$$

With $F_\gamma = F + \gamma B^T W^{-1} B$. Since $Bu = 0$, we can add the term $\gamma B^T W^{-1} Bu$ to the first row in (4.19) without modifying the right hand side. This technique suggests a preconditioner of the form:

$$P_{AL} = \begin{bmatrix} F_\gamma & B \\ 0 & \hat{S} \end{bmatrix}, \tag{4.20}$$

with the inverse of the Schur complement approximated by

$$\hat{S}^{-1} = -(\nu \hat{Q}_p^{-1} + \gamma W^{-1}). \tag{4.21}$$

$\hat{Q}_p$ denotes the approximate pressure matrix, ν is the viscosity and $\gamma > 0$ is a parameter. A good choice of the parameter γ is essential. Usually, W is also replaced by $\hat{Q}_p$. For constant pressure approximation, Q_p is a diagonal matrix. For a linear pressure approximation, Q_p is replaced by a spectrally equivalent diagonal matrix. For a diagonal matrix $\hat{Q}_p$, the computation of the inverse approximate Schur complement is very cheap. The preconditioner is known as augmented Lagrangian preconditioner (P_{AL}).

4.4 SIMPLE-type preconditioners

One family of block preconditioners is the *semi implicit method for pressure-linked equations-type preconditioners* or SIMPLE-type preconditioners. SIMPLE is used by Patanker as an iterative method used to solve the Navier-Stokes problem. The algorithm is based on the following steps:

1. The pressure is assumed to be known from the previous iteration.

2. The velocity is solved from the momentum equations.

3. Since the pressure is only a guess, the newly obtained velocities do not satisfy the continuity equation. In the subsequent substeps the velocities and pressures are corrected in order to satisfy the discrete continuity equation.

In the following, we will present a SIMPLE-type preconditioner for the Navier stokes equations discretized by the Finite Element Method according to Rehman, Vuik and Segal in [43]. The algorithms follows from a block LU decomposition of the coefficient matrix (2.22):

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & -BF^{-1}B^T \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{4.22}$$

The approximation $F^{-1} = D^{-1} = diag(F)^{-1}$ in the $(2,2)$ and $(1,2)$ block of the L and U block matrices, respectively, leads to the SIMPLE algorithm. Solve recursively the following systems:

$$\begin{bmatrix} F & 0 \\ B & -BD^{-1}B^T \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{4.23}$$

and

$$\begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} u^* \\ \delta p \end{bmatrix}. \tag{4.24}$$

This method leads to the following Algorithm for the SIMPLE method:

Algorithm 4.3 SIMPLE Preconditioner

1: p^* is given
2: Solve $Fu^* = r_u$
3: Solve $\hat{S}\delta p = r_p - Bu^*$
4: Update $z_u = u^* - D^{-1}B^T\delta p$
5: Update $z_p = p^* + \delta p$.
6: If not converged go to 2

Vuik *et al.* in [46] have used SIMPLE and it's variants as a preconditioner to solve the Navier-Stokes problem. One iteration of the SIMPLE algorithm is used as a preconditioner. The preconditioner consists of one velocity solve and one pressure solve. Since the systems of equations in the previous algorithm are solved to a certain accuracy, the preconditioner can not be considered constant in subsequent iterations. For that reason the Krylov subspace method `GCR`, which allows variable preconditioners, as outer iteration. Nevertheless, the convergence rate suffers from an increase in the number of grid elements and Reynolds number. It can be proven that the SIMPLE preconditioner improves the overall spectrum of the preconditioned system. Some of the eigenvalues are clustered around 1. The other ones depend on the approximation of the Schur complement matrix.

Proposition: *For the SIMPLE preconditioned matrix* $\tilde{A}$, 1 is an eigenvalue with multiplicity n_u, and the remaining eigenvalues are defined by the generalized eigenvalue problem $Sp = \lambda\hat{S}p$,

with $\hat{S} = -BF^{-1}B^T$. The proof can be found in [42].

4.4.1 SIMPLE(R)

A variant of SIMPLE, known as SIMPLER, is supposed to provide Reynolds-independent convergence. Instead of estimating the pressure p^* in the SIMPLE algorithm, p^* is obtained from solving a subsystem

$$\hat{S}p^* = r_p - BD^{-1}((D - F)u^k + r_u), \tag{4.25}$$

where u^k is obtained from the prior iteration. In case SIMPLER is used as a preconditioner, u^k is taken equal to zero, therefore:

$$\hat{S}p^* = r_p - BD^{-1}r_u. \tag{4.26}$$

The classical SIMPLER algorithm proposed by Patanker consists of two pressure solves and one velocity solve. The complete SIMPLER algorithm is given next:

Algorithm 4.4 SIMPLE(R) Preconditioner

1: Solve $\hat{S}p^* = r_p - BD^{-1}r_u$
2: Solve $Fu^* = r_u - B^T p^*$
3: Solve $\hat{S}\delta p = r_p - Bu^* - Cp^*$
4: Update $z_u = u^* - D^{-1}B^T\delta p$
5: Update $z_p = p^* + \delta p$

Unfortunately, if SIMPLER preconditioned `GCR` is used for finite element discretizations, the convergence may be poor or even divergence may occur, especially in case of low accuracy for the inner systems and in case of fine grids [35].

4.4.2 hSIMPLE(R)

Vuik *et al.* have observed that in the Stokes problem, the SIMPLER preconditioner shows a phase of stagnation at the start of the iterative method. This behavior is not seen in the SIMPLE preconditioner. This is shown in Figure 1 taken from [43]. A better convergence can be achieved if the first iteration is carried out with the SIMPLE preconditioner and after that SIMPLER is employed. The authors of [43] have named this combination hybrid-SIMPLER (hSIMPLER). This implementation gives a fair reduction in the number of iterations if the Stokes problem is solved. However, in the Navier-Stokes problem, SIMPLER performs better than hSIMPLER.

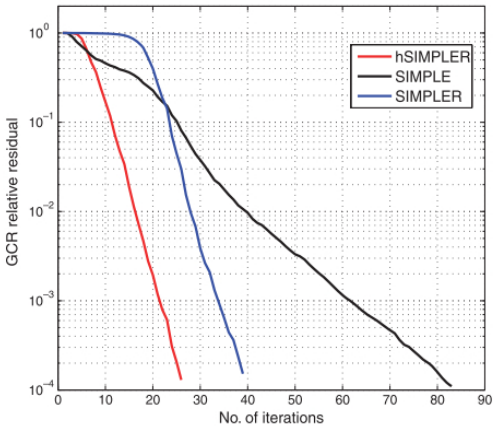


Figure 4.1: Convergence plot of SIMPLE-type peconditioners for the Stokes problem [43]

We can see from Figure 4.1 that there exists a clear stagnation behavior in the initial solution phase of the SIMPLER preconditioner.

4.4.3 M-SIMPLE(R)

Elman *et al.* in [12] discussed the relation between SIMPLE and approximate commutator preconditioners, which is presented next. The more general form of (4.18) is given by

$$(BF^{-1}B^T)^{-1} \approx F_p(BM_1^{-1}B^T)^{-1}, \tag{4.27}$$

where

$$F_p = (BM_2^{-1}B^T)^{-1}(BM_2^{-1}FM_1^{-1}B^T), \tag{4.28}$$

where M_1 and M_2 are scaling matrices. If we now consider a block factorization preconditioner in which the Schur complement is based on a commutator approximation but built on SIMPLE's approximate block factorization written as:

$$P = \begin{bmatrix} F & 0 \\ B & -BM_1^{-1}B^T \end{bmatrix} \begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & F_p^{-1} \end{bmatrix}. \tag{4.29}$$

where $M_1 = D$ and F_p is the identity matrix, then the preconditioner formulation (4.29) corresponds to SIMPLE. The formulation given in (4.29) is equivalent to the SIMPLE algorithm if the subsystem for the pressure part in step 3 in the SIMPLE algorithm is solved with the approximation given in (4.27),

$$\hat{S}\delta p = r_p - Bu^* \tag{4.30}$$

where

$$\hat{S} = -(BM_1^{-1}B^T)F_p^{-1}. \tag{4.31}$$

When FD^{-1} is close to identity, F_p will also be close to identity. This is true in a time dependent problem with small time steps where the diagonal of F has significantly larger

entries than the off-diagonal entries.

Now, we use the observation made by Elman *et al.* regarding time dependent problems. We know that in time dependent problems

$$F_t = \frac{1}{\Delta t} Q_v + F, \tag{4.32}$$

where F_t represents the velocity matrix for the time dependent problem and Δt represents the time step. For small time step $F_t \approx \frac{1}{\Delta t} Q_v$. This kind of approximation has been used in fractional step methods for solving the unsteady Navier-Stokes problem. We use this idea in solving the steady Navier-Stokes problem. Therefore, we choose $M1 = M2 = \hat{Q}_v$ in (4.27) resulting in:

$$F_p = (B\hat{Q}_v^{-1}B^T)^{-1}(B\hat{Q}_v^{-1}F\hat{Q}_v^{-1}B^T). \tag{4.33}$$

If we assume that the factor $F\hat{Q}_v^{-1}$ in F_p is close to identity, then

$$F_p = (B\hat{Q}_v^{-1}B^T)^{-1}(B\hat{Q}_v^{-1}B^T) \approx I, \tag{4.34}$$

and the approximation (4.27) becomes

$$BF^{-1}B^T \approx -B\hat{Q}_v^{-1}B^T. \tag{4.35}$$

Based on this result, we replace D^{-1} in the SIMPLER algorithm by $\hat{Q}_v^{-1}$. This method is referred to as MSIMPLER (Modified SIMPLER).

Algorithm 4.5 M-SIMPLER Preconditioner

1: Solve $\hat{S}p^* = r_p - B\hat{Q}_v^{-1}r_u$
2: Solve $Fu^* = r_u - B^T p^*$
3: Solve $\hat{S}\delta p = r_p - Bu^*$
4: Update $z_u = u^* - \hat{Q}_v^{-1}B^T\delta p$
5: Update $z_p = p^* + \delta p$

It is clear from the previous algorithm that the cost of MSIMPLER is equal to the cost of the SIMPLER preconditioner. However, in solving the Navier-Stokes problem, at each nonlinear iteration, the Schur complement approximation in the MSIMPLER does not to be built again because the operators used in the Schur complement approximation are independent of any change that may take place at each nonlinear iteration.

Chapter 5

Numerical Experiments

In search for an answer to the research questions posed in the introduction of this thesis, a collection of `MATLAB` programs was developed and a set of numerical experiments was performed. Mainly, the SIMPLE(R) and M-SIMPLER preconditioners were implemented as algorithms to solve the Navier-Stokes saddle-point algebraic problem. A routine to use deflation techniques was also implemented. Programs to investigate the spectrum of eigenvalues of the resulting preconditioned and deflated matrices were also developed, allowing for a clear comparison of the respective spectrums. The algorithms were implemented under the `IFISS` environment, making use of the data matrices generated by its discretization routines. `IFISS` is a graphical `MATLAB` package for the interactive numerical study of incompressible flow problems [11], [13]. It includes algorithms for discretization by mixed finite element methods and a posteriori error estimation of the computed solutions. The package can also be used as a computational laboratory for experimenting with preconditioned iterative solvers for the discrete linear equation systems that arise in incompressible flow modeling. For each problem addressed, the package allows for the study of the discretization and the iterative solution algorithms as well as the interaction between the two and the resulting effect on overall efficiency. In the following section, we present a series of numerical experiments that was carried out using the developed programs.

5.1 Reference problems

Three solutions of the Navier-Stokes problem (2.1) and (2.2) are presented here as reference problems. These solutions will work as a framework in which we can investigate the performance of the preconditioned and deflated `GMRES` method. The solutions have been calculated using the `IFISS` package.

2D Poiseuille Flow

This problem represents steady horizontal flow in a channel driven by a pressure difference between the two ends, more commonly known as Poiseuille flow. The domain is given by:

$$\boldsymbol{\Omega}_1 : \text{the square } (-1,1) \times (-1,1).$$

Here, a solution is computed numerically on Ω_1 using the velocity $u = (1-y^2, 0)$ to define a Dirichlet condition on the inflow boundary $x = -1$. The no-flow Dirichlet condition $u = 0$ is applied on the characteristic boundaries $y = -1$ and $y = 1$. At the outflow boundary $(x = 1, -1 < y < 1)$, there is a choice of applying a Neumann or a Dirichlet condition. The Poiseuille channel flow solution is an analytic solution of the Navier-Stokes equations and it is only obtainable since the convection term is identically zero. In the solution, the pressure gradient is proportional to the viscosity parameter. The solution is given next.

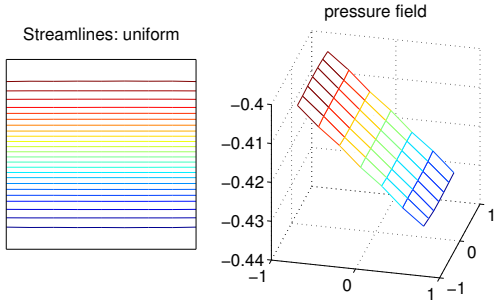


Figure 5.1: Solution to the Poiseuille flow problem.

Driven Cavity Flow

This is a classical test problem used in fluid dynamics, known as driven-cavity flow. It is a model of the flow in a square cavity, that is, the domain is Ω_1 with the lid moving from left to right. A Dirichlet no-flow condition is applied on the side and bottom boundaries. Different choices of the nonzero horizontal velocity on the lid give rise to different computational models:

$$\begin{aligned} \{y = 1; -1 \le x \le 1 | u_x = 1\}, &\quad \text{a leaky cavity;} \\ \{y = 1; -1 < x < 1 | u_x = 1\}, &\quad \text{a watertight cavity;} \\ \{y = 1; -1 \le x \le 1 | u_x = 1 - x^4\}, &\quad \text{a regularised cavity;} \end{aligned}$$

The solution of the driven cavity problem is presented next.

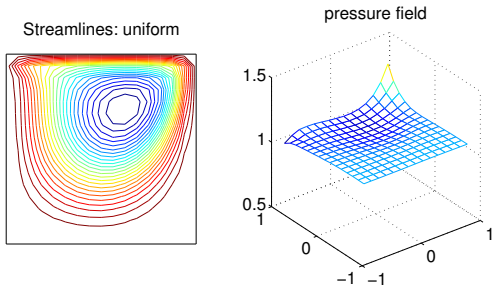


Figure 5.2: Solution to the Driven cavity problem with a watertight cavity.

Backward facing step Flow

This example represents the flow over a step of length L. The domain is given by:

$\mathbf{\Omega}_2$: the L-shaped region generated by taking the complement in $(-1, L) \times (-1, 1)$ of the quadrant $(-1, 0] \times (-1, 0]$.

A Poiseuille flow is imposed on the inflow boundary ($x = 0; 0 \leq y \leq 1$), and a no-flow (zero velocity) condition is imposed on the top and bottom walls. A Neumann condition is applied at the outflow boundary which automatically sets the mean outflow pressure to zero. The solution of this problem is presented next:

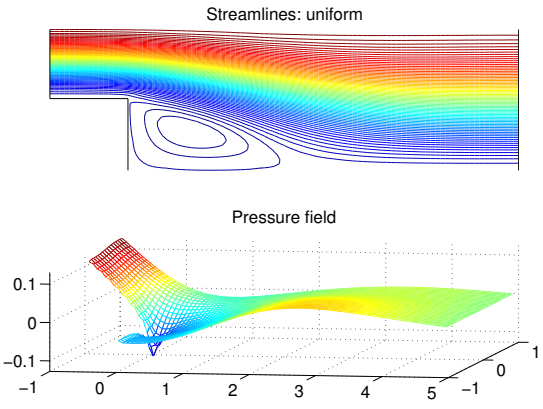


Figure 5.3: Solution to the backward facing step with stabilized $\mathbf{Q_1} - \mathbf{P_0}$ approximation.

The performance of iterative Krylov subspace methods and preconditioners can be studied using the built in functions of `IFISS`. As an example, we present the convergence history of the `GMRES` method with different methods of preconditioning. The convergence behavior is presented next.

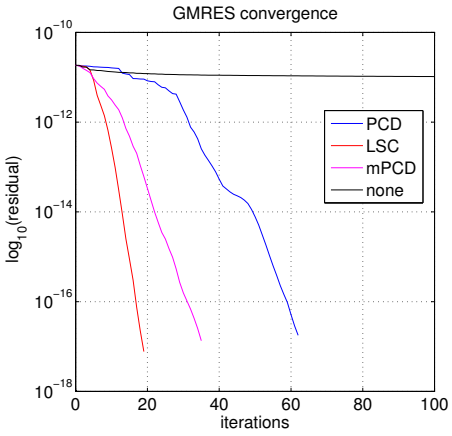


Figure 5.4: Convergence plot of GMRES.

Using these three solutions of the Navier-Stokes equations as a reference, we can test the performance of the implemented algorithms. The following sections focus on the results obtained by implementing the SIMPLER preconditioner.

5.2 Results of SIMPLE(R) preconditioner

First, we present the performance results of the `GMRES` method when SIMPLER is used as a preconditioner and is used to solve the backwards facing step flow, the Poiseuille flow and the driven cavity flow. Two sets of experiments were carried out for each of the distinct reference problems; the first set involves the construction of the SIMPLER preconditioner using the matrix operators with imposed boundary conditions and will be referred to as SIMPLER(WBC). In the second set, the preconditioners were constructed using the matrices without the boundary conditions imposed on them and we will call this variation the SIMPLER(WOBC) preconditioner. The difference between implementations is mainly how the Schur complement matrix is built and the precise matrices which are used are explained in the following subsections. In the first case, the divergence operator with the imposed boundary conditions is used to construct the Schur complement approximation matrix; the second one involves the construction of the Schur complement matrix without applying the boundary conditions. A stagnation behavior in the initial phase of the solution of the resulting linear system of equations using the `GMRES` method with right preconditioning as the iterative solver was found when the Schur complement approximation was constructed with the boundary conditions already applied to the matrix operators in the the SIMPLER algorithm. This stagnation behavior was found to be in accordance with the results presented by [39]. This stagnation behavior was absent when the preconditioner was built without the imposed boundary conditions.
Second, we present a series of experiments which was carried out using stretched grids. When SIMPLER(WBC) was used, the stagnation phase seemed to be independent of the stretch factor, yielding the same solution behavior and total number of iterations for the

different stretch factors applied to the same test problem. This is not the case for SIMPLER(WOBC). Here, a tendency towards stagnation is found when the grid is stretched throughout the whole solution process.
As a third set of results and due to the diversity of behaviors found, the eigenvalues of the preconditioned matrices were explored. We present the comparison of spectrums of the original matrix, the preconditioned matrix and the preconditioned and deflated matrix. The visualization of the spectrums helped in the construction of the deflation matrix in order to identify the eigenvalues and corresponding eigenvectors which represent a problem for the iterative solver.

5.2.1 Operators with B.C.

In this subsection we present the results obtained while using the SIMPLER(WBC) as a preconditioner. The preconditioner was obtained by constructing the Schur complement matrix needed in the SIMPLER preconditioner with the prescribed boundary conditions applied to them. That is, in the LU-decomposition of the coefficient matrix:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & -BF^{-1}B^T \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{5.1}$$

The matrix B in the $(2,2)$ block of the first matrix was identified as the matrix with imposed boundary conditions.
In the performed simulations, the discretization of the domains was obtained with the use of $Q_2 - Q_1$ finite elements in every case. The linearization of the nonlinear system of equations was performed via the Picard method. The tolerance of the linear approximation was set to $1e-8$, The number of iterations in order to achieve this accuracy are given in parenthesis in the table of results. In the solution of the resulting linear system of equations, the criteria used to measure the relative error of the residual norm was $1e-6$. That is, we accept an iterate x_k of the `GMRES` method with right preconditioning as a valid solution when,

$$\frac{\|M^{-1}(B - Ax_k)\|}{\|M^{-1}(B - Ax_0)\|} \leq 1e{-}6. \tag{5.2}$$

For each of the three distinct reference problems, the Number of iterations of the solution method were measured for varying viscosity as well as for different grid sizes. A convergence plot of the `GMRES` method was also explored for a system with a fixed number of grid points. The graph shows the logarithm of the relative residual norm with respect to the number of iterations. That is, $\log_{10}(\frac{\|M^{-1}(B-Ax_k)\|}{\|M^{-1}(B-Ax_0)\|})$ Vs. k where k is the iteration number.

For the Poiseuille flow the following table was obtained while measuring the iteration count for varying Reynold's number and system size, in this case, the convergence plot shown is for a system of 128×128 grid points:

ν	0.1	0.01	0.005
$h = \frac{1}{32}$	17 (1)	22 (1)	30 (1)
$h = \frac{1}{64}$	27 (1)	24 (1)	29 (1)
$h = \frac{1}{128}$	44 (1)	34 (1)	33 (1)
$h = \frac{1}{256}$	72 (1)	66 (1)	54 (1)

Table 5.1: Iteration numbers for the Poiseuille flow using the FGMRES method preconditioned with SIMPLER(WBC). The number of Picard iterations appears in parenthesis.

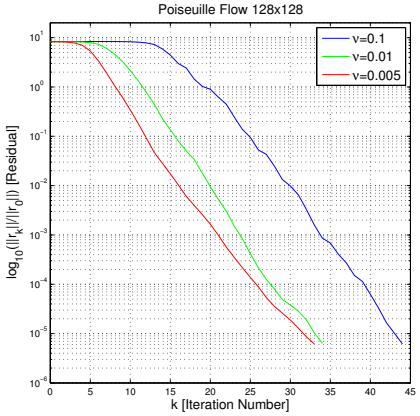


Figure 5.5: Convergence plot of FGMRES method preconditioned with SIMPLER(WBC) for the Poiseuille flow.

For the Driven Cavity problem the following iteration count table was obtained, the convergence plot presented is of a system of 128×128 regular grid points:

ν	0.1	0.01	0.005
$h = \frac{1}{32}$	16 (6)	18 (13)	21 (14)
$h = \frac{1}{64}$	24 (5)	23 (12)	24 (13)
$h = \frac{1}{128}$	37 (5)	40 (11)	36 (11)
$h = \frac{1}{256}$	58 (4)	66 (10)	67 (10)

Table 5.2: Iteration numbers for the driven cavity flow using the FGMRES method preconditioned with SIMPLER(WBC). The number of Picard iterations appears in parenthesis.

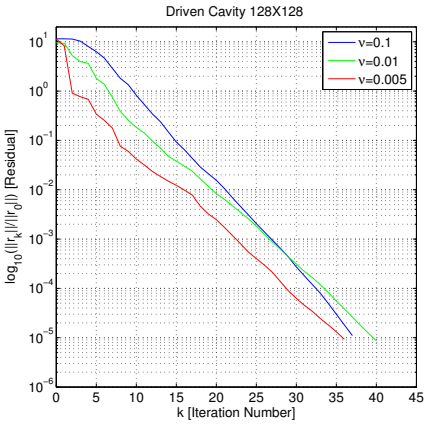


Figure 5.6: Convergence plot of FGMRES method preconditioned with SIMPLER(WBC) for the driven cavity flow.

For the backward facing step flow the following data was obtained, the convergence plot for this reference problem is of a system of 64×128 grid points:

ν	0.1	0.01	0.005
16×48	20 (7)	34 (25)	50 (64)
32×96	29 (7)	38 (23)	56 (70)
64×192	45 (6)	47 (21)	57 (63)
128×384	70 (6)	74 (19)	77 (54)

Table 5.3: Iteration numbers for the backward facing step flow using the FGMRES method preconditioned with SIMPLER(WBC). The number of Picard iterations appears in parenthesis.

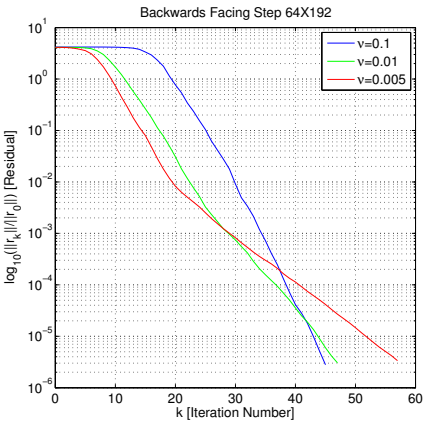


Figure 5.7: Convergence plot of FGMRES method preconditioned with SIMPLER(WBC) for the backwards facing step flow.

This set of results shows that in all of the different test problems, a stagnation behavior was found in the initial phase of the solution algorithm of the SIMPLE(R) preconditioner. The stagnation behavior seems to be somewhat absent in the driven cavity flow problem. The iteration count seems to increase with an increase of system size, except for the backwards facing step problem, in which the iteration count stays relatively constant for an increase in system size. For a change in the viscosity of the flow, the iteration count is relatively constant and in some cases it decreases for problems with lower viscosity. The spectrum of the preconditioned system will be analyzed in detail in further sections. First, we present the results of the alternative way of constructing the SIMPLER preconditioner.

5.2.2 Operators without BC

The results encountered in the previous subsection suggest a difference in the convergence behavior between problems with imposed Dirichlet boundary conditions and problems with imposed Neumann boundary conditions. A set of simulations was repeated for the same test problems following a different construction of the SIMPLER preconditioner. The difference being that the Schur complement matrix was constructed with the matrix operators before the boundary conditions were imposed to them. That is, the matrix B and B^T in the Schur complement approximation are identified as the matrices before the boundary conditions are imposed on them. We call the resulting algorithm the SIMPLER(WOBC). The simulations were performed with the same finite element choice and tolerance bounds as the previous section. The number of iterations for the relative residual to attain a value below the tolerance bound was measured for varying viscosity and system size. The number of Picard iterations measured to achieve the desired tolerance appear in parenthesis.

Now we present the results obtained from simulating the three different types of flow.

For the Poiseuille flow, the measured iteration counts are presented in the following table. For a visual representation of the convergence history of the iterative method, we present a convergence plot of a system of 256×256 nodes. Once again, the graph shows the logarithm of the relative residual norm with respect to the number of iterations.

ν	0.1	0.01	0.005
$h = \frac{1}{32}$	33 (1)	30 (1)	38 (1)
$h = \frac{1}{64}$	46 (1)	36 (1)	37 (1)
$h = \frac{1}{128}$	69 (1)	57 (1)	50 (1)
$h = \frac{1}{256}$	100 (1)	102 (1)	84 (1)

Table 5.4: Iteration count for the Poiseuille flow using the FGMRES method preconditioned with SIMPLER(WOBC).

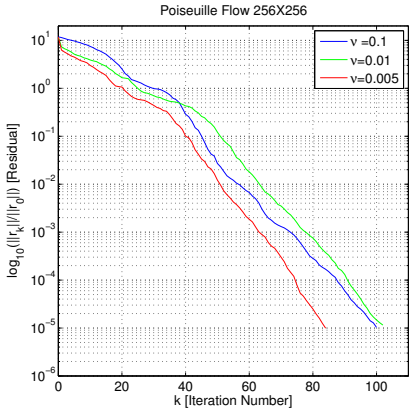


Figure 5.8: Convergence plot of FGMRES method preconditioned with SIMPLER(WOBC) for the Poiseuille flow.

In the case of the Driven cavity problem, the following iteration counts were measured, also, the convergence plot of the largest simulation with a size of 256×256 nodes is presented next:

ν	0.1	0.01	0.005
$h = \frac{1}{32}$	36 (6)	28 (13)	31 (14)
$h = \frac{1}{64}$	50 (5)	42 (12)	38 (13)
$h = \frac{1}{128}$	71 (5)	68 (11)	62 (11)
$h = \frac{1}{256}$	98 (4)	113 (10)	109 (10)

Table 5.5: Iteration count for the driven cavity flow using the FGMRES method preconditioned with SIMPLER(WOBC).

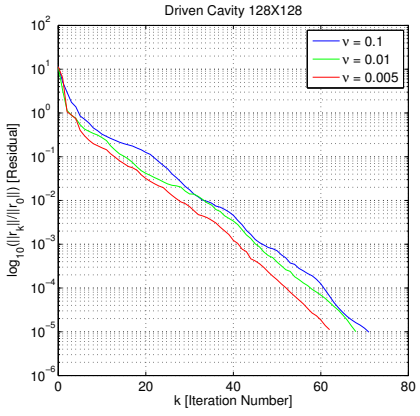


Figure 5.9: Convergence plot of FGMRES method preconditioned with SIMPLER(WOBC) for the driven cavity flow.

In the case of the backwards facing step flow, the following measurements were obtained, the convergence plot for a system of 64×192 nodes is also presented:

ν	0.1	0.01	0.005
16×48	40 (7)	44 (25)	56 (64)
32×96	55 (7)	51 (23)	62 (70)
64×192	80 (6)	70 (21)	70 (63)
128×384	113 (6)	115(19)	103(54)

Table 5.6: Iteration numbers for the backwards facing step flow using the FGMRES method preconditioned with SIMPLER(WOBC).

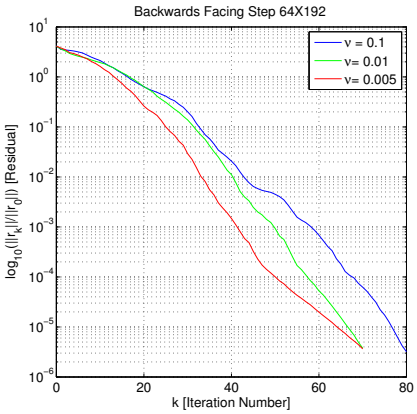


Figure 5.10: Convergence plot of FGMRES method preconditioned with SIMPLER(WOBC) for the backwards facing step flow.

We can see that for every simulation, the iteration number increased with respect to the simulations performed in the previous section. Nevertheless, a clear stagnation behavior is not found in the initial phase of the solution process. For the driven cavity flow, an initial phase of super-convergence is found. The iteration count decreases for an increase in viscosity of the flow, while there is an increase in iteration count for systems of a greater number of elements.

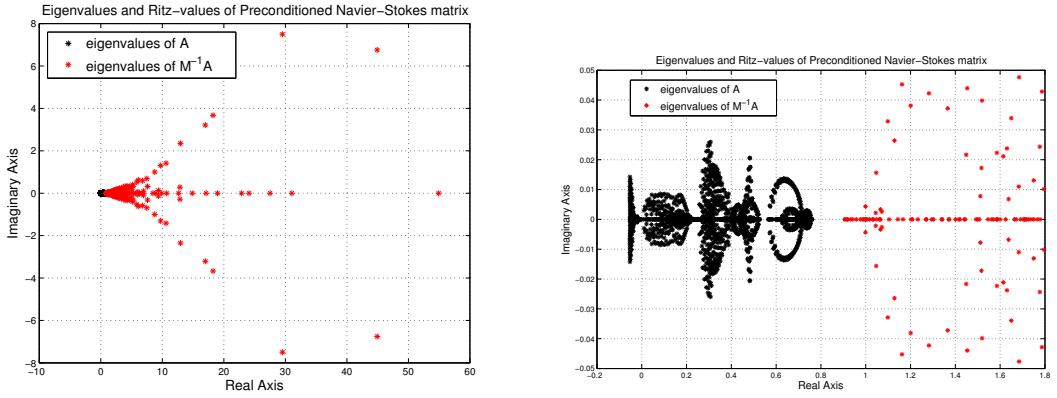
The combination of the results obtained in the preceding two subsections suggest that for enclosed flow problems, that is for problems without homogeneous Neumann boundary conditions, SIMPLER(WOBC) provides a better convergence behavior in the initial phase of the solver. Although SIMPLER(WOBC) results in more iterations of the iterative solver, it can be used in combination with SIMPLER(WBC) to achieve better convergence results. As a suggestion, one could implement a solver for which the first steps are performed using SIMPLER(WOBC) while the remaining solution steps may be performed using SIMPLER(WBC).

5.3 Eigenvalue Exploration

An eigenvalue exploration was carried out for problems where a stagnation behavior was found in the initial phase of the solver. This is the case for all the test problems when using SIMPLER(WBC) except maybe for some cases of the driven cavity flow. In the following section, a distinction is made for cases with and without stretching.

5.3.1 Stagnation without stretching

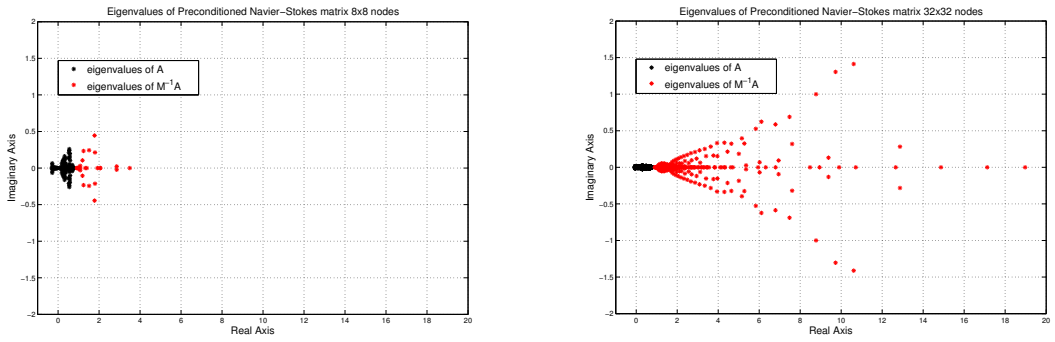
A clear stagnation behavior in the initial phase of the solver is found, for example, in the Poiseuille flow simulation with a system size of 32×32 nodes and a viscosity of $\nu = 0.1$ where the initial stagnation behavior is clear. The spectrum of eigenvalues for this problem is plotted in the next figures for the matrices with and without preconditioning.



(a) complete picture (b) zoomed section

Figure 5.11: Spectrum of eigenvalues of the Navier-Stokes system (A) and preconditioned with SIMPLER ($M^{-1}A$) for the Poiseuille flow

Indeed, a shifting of the eigenvalues towards the value one is found for the preconditioned matrix. Before the preconditioning, all the real parts of the eigenvalues are distributed between $[-0.05, 1]$ on the real axis. After the preconditioning, most of the eigenvalues are clustered around one, however, the range of distinct values of the real part of the eigenvalues extends to $[0.9, 54.9]$ on the real axis. This shows a change in the properties of the matrix. It can be seen that after the preconditioning, the matrix becomes positive definite. Krylov-type iterative solvers such as GMRES have proved to be particularly efficient for systems with a Hermitian positive definite matrix, or more generally, for systems with a matrix with all eigenvalues in the right half of the complex plane. Navier-Stokes-type systems, however, are highly indefinite, which means that the system matrix has eigenvalues with both negative and positive real parts, a characteristic that can result in a very slow convergence. It can be seen that most of the eigenvalues situate themselves around the value one with the exception of a few eigenvalues with a relatively large norm. This observation encouraged the exploration of the spectrum of eigenvalues of the preconditioned matrix for varying grid sizes. The results are plotted next:



(a) 8×8 nodes (b) 32×32 nodes

Figure 5.12: Spectrum of eigenvalues of the Navier-Stokes system (A) and preconditioned with SIMPLER ($M^{-1}A$) for the Poiseuille flow for different system size.

A clear pattern in the spectrum of eigenvalues for the different system sizes is not directly apparent. This apparent "lack of pattern" suggests that an investigation of the Ritz values should be done at different steps of the solution phase. If the iterative method is first trying to approximate the eigenvalues which are the ones far away from 1, this may cause the stagnation behavior and a deflation approach might be useful in order to eliminate the stagnation. Such a study of the Ritz values is presented next.
If we look at the convergence history of the Poiseuille flow for a system of 32×32 nodes, we can observe a "stagnation behavior", this is shown in the next figure:

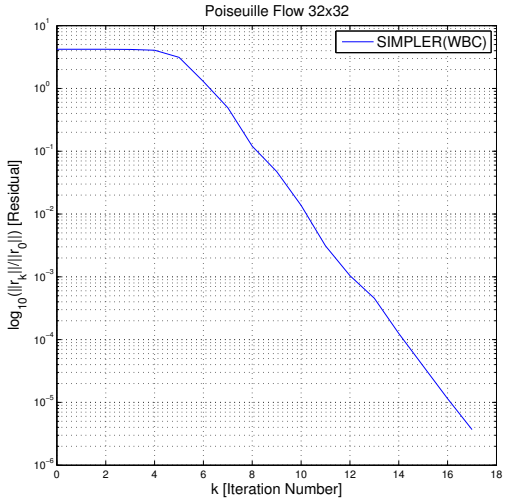
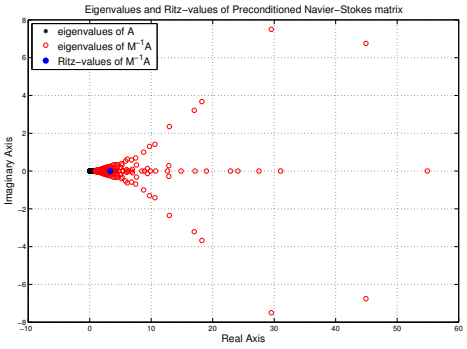
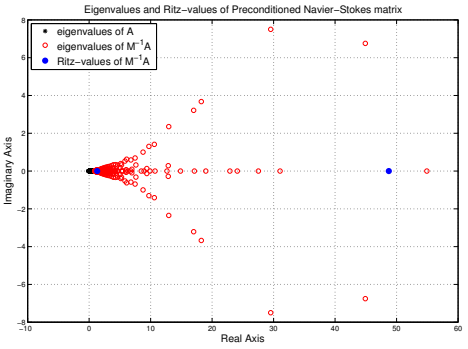


Figure 5.13: Convergence plot of FGMRES method preconditioned with SIMPLER(WOBC) for the Poiseuille flow.

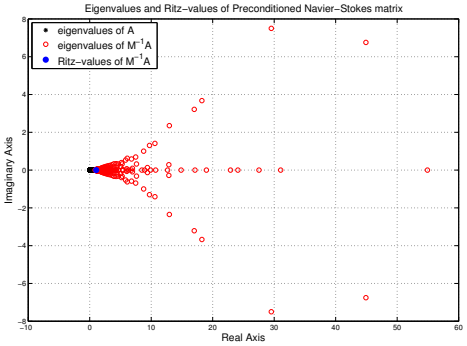
If we now look at the Ritz values at different iteration steps, we can see that not until the eigenvalues situated in (44.9,6.75) and (44.9,-6.75) in the complex plane are approximated correctly in iteration 7, then the iterative solver continues with a linear convergence:



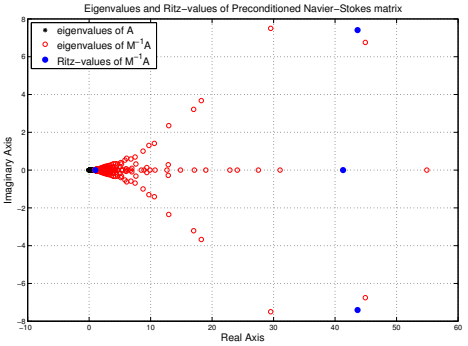
(a) iteration 3



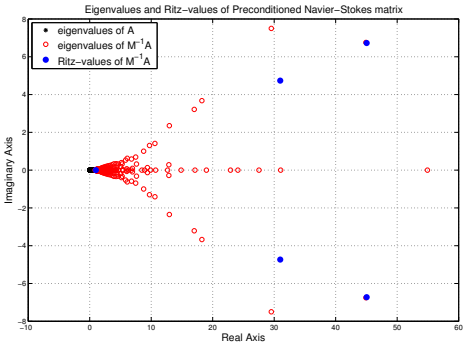
(b) iteration 4



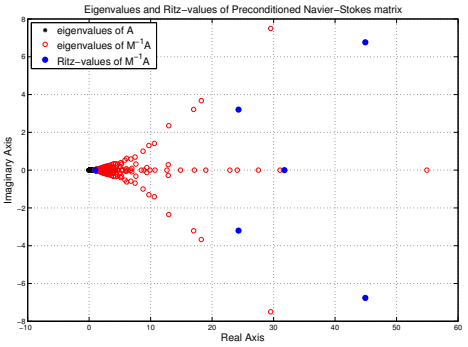
(c) iteration 5



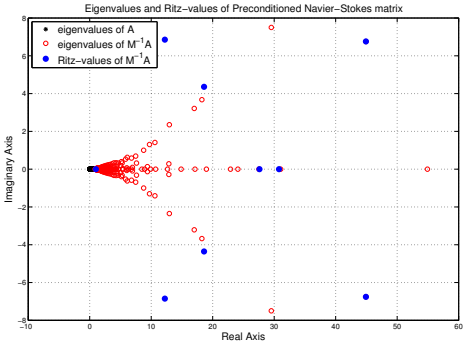
(d) iteration 6



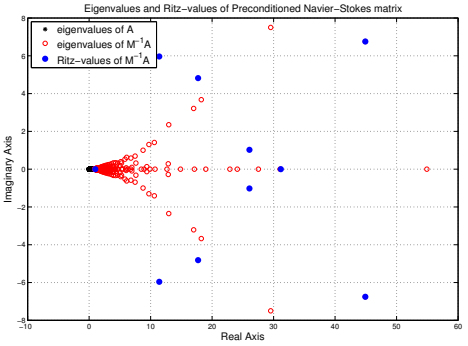
(e) iteration 7



(f) iteration 8



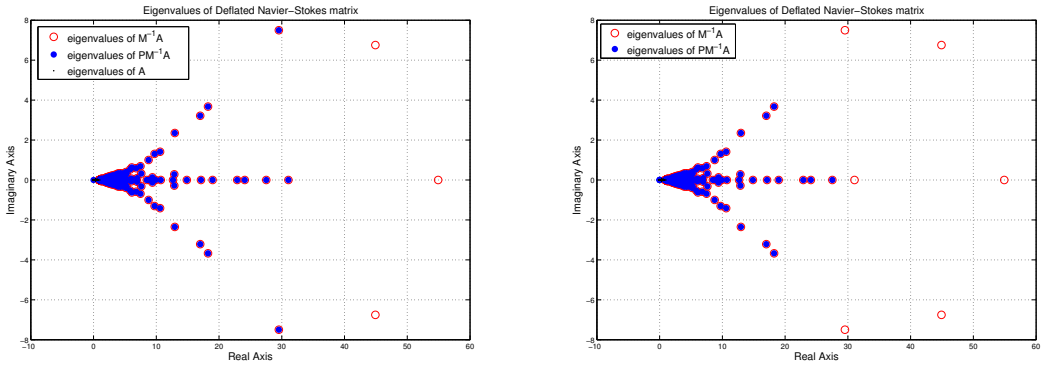
(g) iteration 9



(h) iteration 10

Figure 5.14: Comparison of eigenvalues and Ritz values of the Navier-Stokes matrix preconditioned with SIMPLER ($M^{-1}A$) for the Poiseuille flow for different iteration numbers.

This suggest that a deflation of the eigenvectors corresponding to these "ill conditioned" eigenvalues may cause the iterative solver to converge linearly. By following the deflation framework presented in Chapter 3, we can construct a deflation matrix which includes the desired eigenvectors as columns, in this case, the identified "ill conditioned" vectors which correspond to the vectors with the largest real part. The spectrum of the deflated system is given next:



(a) 3 deflation vectors

(b) 6 deflation vectors

Figure 5.15: Comparison of eigenvalues of the Navier-Stokes system preconditioned with SIMPLER ($M^{-1}A$) [red] and of the deflated system [blue] for the Poiseuille flow for different sizes of the deflation matrix

It can be seen that, after the projection, the desired eigenvalues are eliminated from the original spectrum of the Navier-Stokes matrix. In this case, the deflation matrix has complex vectors due to the wish of eliminating some eigenvalues with complex components. In the following chapter, we provide a deeper discussion of the overall impact of deflation with respect to the performance of the `GMRES` method.

5.3.2 Stretched Grids

A set of simulations with stretched grids was performed in order to look for stagnation behavior in the initial phase of the iterative solver. The backward facing step was the problem of choice for the simulations. A system of 32×96 nodes is chosen with a viscosity of $\nu = 0.1$. The convergence history of this problem with a stretched grid was investigated for both types of constructed preconditioners mentioned above (with and without imposed boundary conditions). The following graph was obtained:

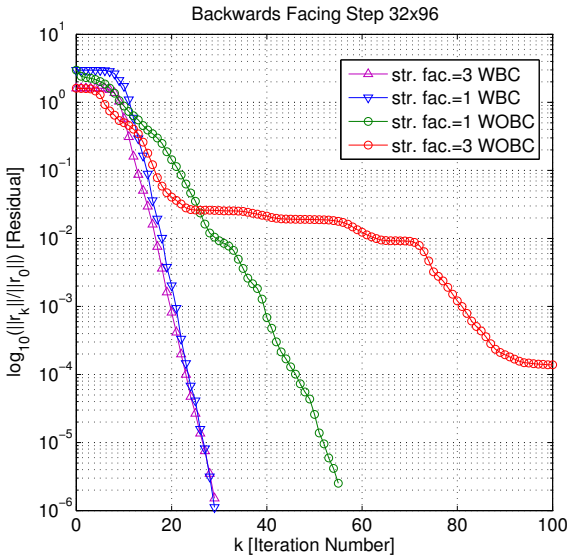
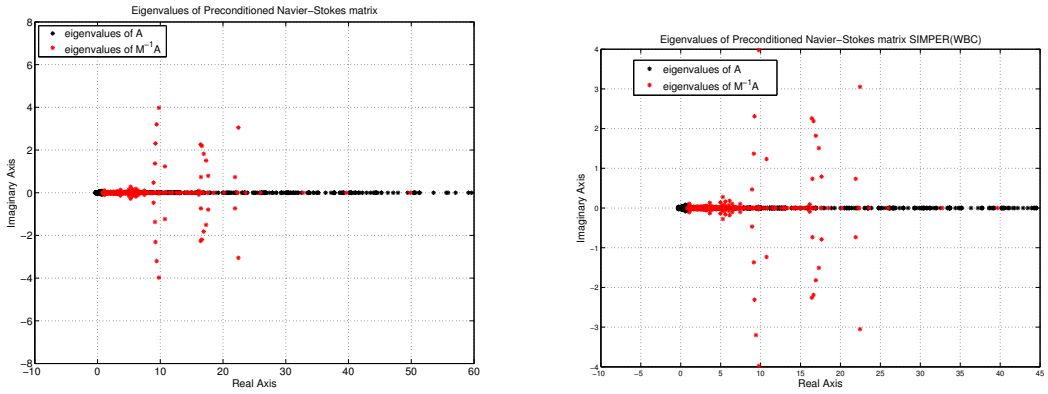


Figure 5.16: Convergence plot of FGMRES with preconditioning for a stretched grid for the backward facing step flow.

It was found that, when SIMPLER(WBC) is used as a preconditioner, the number of iterations for the stretched and un-stretched grids remains constant. Moreover, the stagnation phase at the beginning of the solution process remains unchanged regardless of the stretching. This can not be said when SIMPLER(WOBC) is used. In this case, we see that the stretch factor greatly influences the solution process creating stagnation behaviors, not at the beginning, but throughout the whole solution process. For a stretch factor of 3, as many as 100 iterations were not enough to achieve the desired tolerance value.

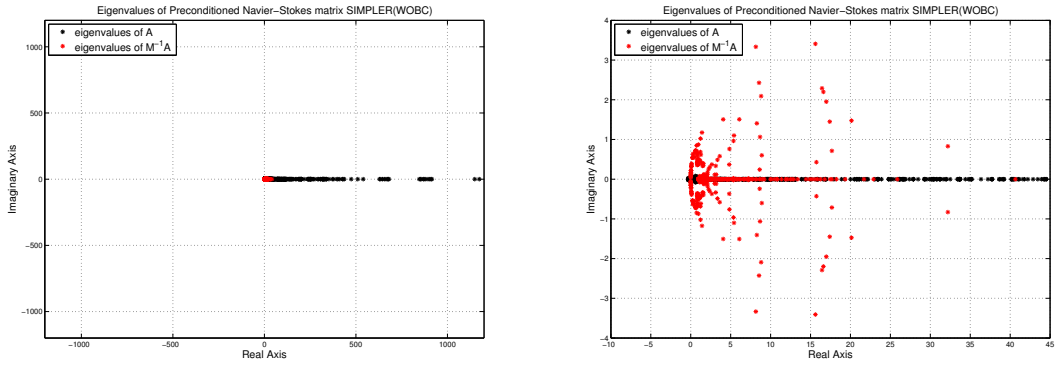
5.3.3 Stagnation with stretching

As mentioned before, the stagnation behavior was found in the simulation of the backward facing step used for the stretched grid exploration in the previous chapter. In this case, the eigenvalues were plotted for both types of preconditioners, SIMPLER(WBC) and SIMPLER(WOBC).



(a) Full Spectrum (b) Zoomed section

Figure 5.17: Spectrum of eigenvalues of the Navier-Stokes system (A) and preconditioned with SIMPLER(WBC) ($M^{-1}A$) for the Backward facing step with stretching.



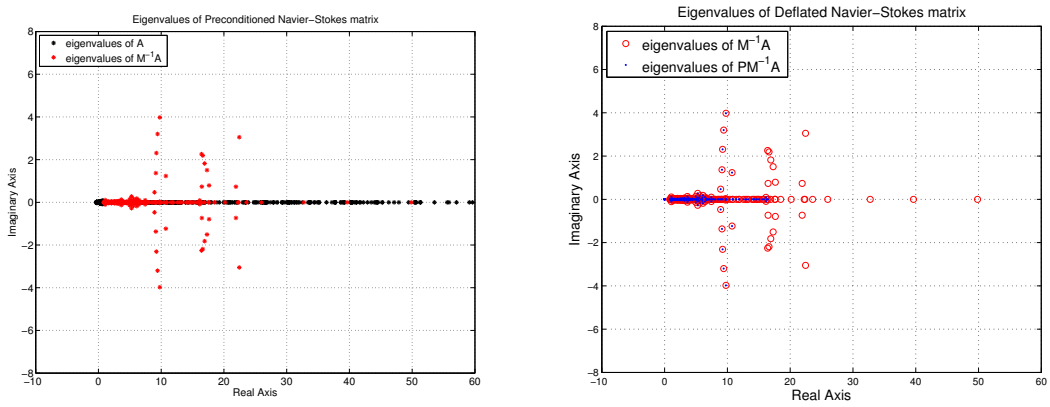
(a) Full Spectrum (b) Zoomed section

Figure 5.18: Spectrum of eigenvalues of the Navier-Stokes system (A) and preconditioned with SIMPLER(WOBC) ($M^{-1}A$) for the Backward facing step with stretching.

It was found that the clustering of the eigenvalues around one was a much more visible in the case of SIMPLER(WBC). The largest eigenvalue of SIMPLER(WBC) has a norm of roughly 7000 while the largest eigenvalue found for SIMPLER(WOBC) has a norm of 41. Moreover, many eigenvalues were found between $[0, 1]$ for SIMPLER(WOBC). This may cause the tendency for a general stagnation characteristic of the convergence history of the method. However, the stagnation phase at the initial solution of SIMPLER(WBC) is not clearly explained by an overlook of the eigenvalues neither for stretched or non-stretched grids.

Once again, the Arnoldi algorithm allows us to find some ”ill conditioned” eigenvalues, however, there was not a clear pattern in the eigenvalues for stretched grids. It was found that depending on the problem and depending on the stretch factor there was a

large gamma of structures for the spectrum of eigenvalues. Nevertheless, the deflation approach was carried out. As we have seen in the case of un-stretched grids, the deflation approach works better with a large quantity of eigenvalues, including, but not only, those identified with the Arnoldi algorithm. Once again, we include a plot of the deflated system for SIMPLER(WBC) in the following figure:



(a) Preconditioned Spectrum

(b) Deflated Spectrum

Figure 5.19: Comparison of eigenvalues of the Navier-Stokes system preconditioned with SIMPLER(WOBC) ($M^{-1}A$) [red] and of the deflated system [blue] for the Backward Facing Step flow.

It is clear that after the deflation scheme, a clustering of eigenvalues is pronounced. By getting rid of the eigenvalues of highest norm we obtain a decrease in the stagnation behavior of `GMRES` for SIMPLER(WBC) while the opposite is true for SIMPLER(WOBC), where, even though the spectrum is clustered together, the overall distribution of eigenvalues covers a larger area of the imaginary plane than that of SIMPLER(WBC). The convergence plots as well as a discussion on the convergence behavior of problems with stretched grids can be found in the Chapter 6 where we will discuss in greater detail the results obtained in the performed simulations as well as introduce a discussion on the stability of Finite Elements when dealing with stretched grids.

Chapter 6

Discussion

6.1 Stagnation Behavior

In this section, we present a discussion on the results obtained in the performed numerical experiments. We provide a discussion on the deflation approach to eliminate the stagnation behavior as well as a theoretical discussion on a possible way of constructing finite elements that which are stable when stretched grids are used.

6.1.1 Un-Stretched grids

In the previous chapter, a stagnation behavior was found in the initial phase of the `GMRES` method with right preconditioning when the SIMPLER(WBC) preconditioner was used. The performed simulations suggest that this behavior is dependent on the Reynolds number of the flow, or more precisely, on the viscosity of the fluid. For the Poiseuille flow, the following plot exemplifies this phenomenon:

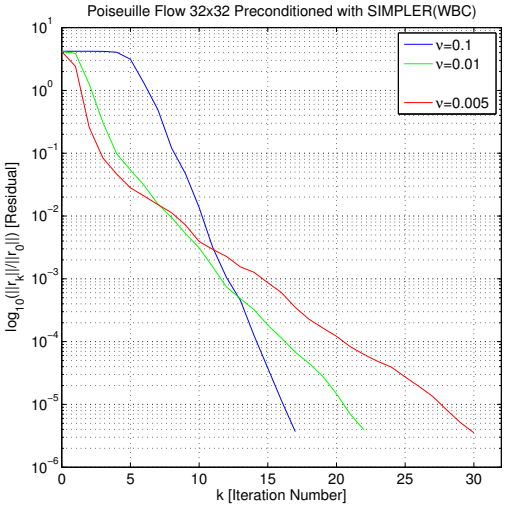


Figure 6.1: Spectrum of eigenvalues of the Navier-Stokes system preconditioned with SIMPLER ($M^{-1}A$) for the Poiseuille flow for varying viscosity.

These results point out the fact that for different viscosities (ν's) we can observe different behaviors of the `GMRES` method. The initial stagnation behavior can be observed when the viscosity $\nu \approx 1$. However, when the viscosity decreases, the stagnation behavior is absent but an increase in total iterations of the iterative solver can be observed. This may be due to the use of upwinding schemes in the `IFISS` environment. Whenever the viscosity decreases, a boundary layer is formed in order for the computations to be accurate. Due to this fact, we focus our attention on the regime of viscosity for which a stagnation behavior is clear ($\nu \approx 1$).

In order to apply the deflation scheme, it is necessary to identify the "ill conditioned" eigenvectors. In the previous chapter, we identified these vectors with the eigenvectors corresponding to the eigenvalues with largest real part. This identification was made by analyzing the Arnoldi iterations and following the behavior of the Ritz values as the number of iterations increased. In order to construct the deflation matrix, the exact eigenvectors were used. In practice, this way of constructing the deflation matrix may be too costly or merely impossible. Nevertheless, in order to show the impact of deflation with respect to the `GMRES` method, we present the following convergence behavior of the method for different sizes of the deflation matrix:

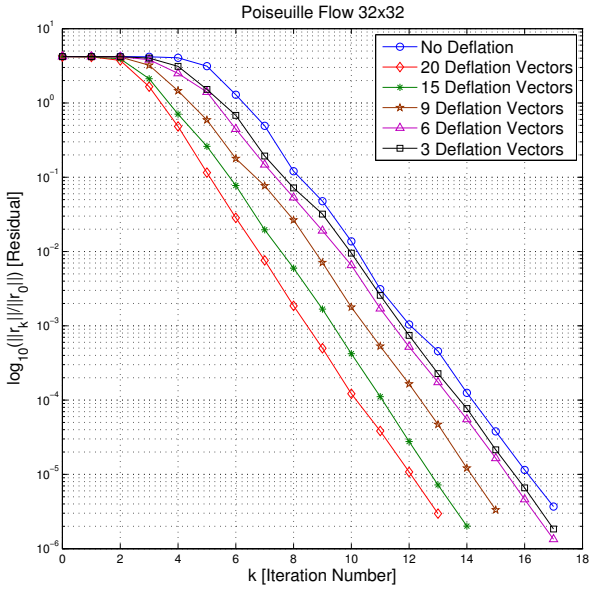


Figure 6.2: Convergence History of the SIMPLER(WBC) preconditioned Navier-Stokes system ($M^{-1}A$) and deflated ($PM^{-1}A$) for the Poiseuille Flow for different sizes of the deflation matrix.

It is clear that the stagnation behavior of the `GMRES` preconditioned from the right with the SIMPLER(WBC) preconditioner decreases as the size of the deflation matrix increases. The total iteration count also decreases with this approach. This shows that not only the first three eigenvectors corresponding to the eigenvalues with largest real part are the cause of the stagnation behavior, but the fault lies in most of the eigenvalues which are

not closely clustered together. The stagnation behavior does not completely vanish due to the fact that the deflation approach sends the ill conditioned eigenvalues to zero, however, due to round-off errors, they are not exactly zero. This is shown in Figure 6.3. This causes the iterative solver to assume that there are still small eigenvalues different from zero which need to be approximated, giving rise to a larger number of iterations and a stagnation behavior.

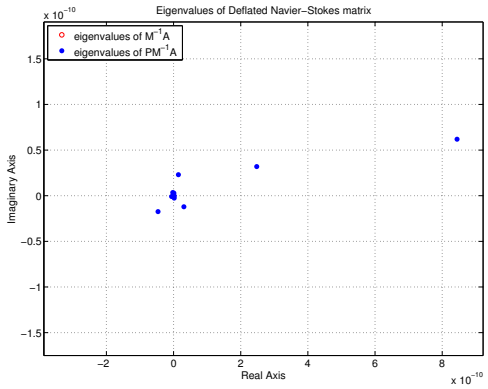


Figure 6.3: Deflated eigenvalues of the Navier-Stokes system preconditioned with SIMPLER ($M^{-1}A$) for the Poiseuille flow wich are not identically zero.

Now, the deflation approach used was that of first identifying the "ill conditioned" eigenvalues from the spectrum of the preconditioned matrix. Once this is done, we proceed to build the deflation matrix with the corresponding eigenvectors as its columns. Once the deflation matrix is constructed, the original matrix A is projected via the matrix multiplication PA. After the desired components are projected, we proceed to implement the preconditioned `GMRES` algorithm to solve the matrix PA. This approach is known as DEF1. It is important to emphasize the fact that the deflation matrix is composed of complex-valued eigenvectors. This is due to the fact that the "ill-conditioned" eigenvalues are complex. The usual deflation approach found in [37] focuses on real-valued deflation schemes, however, in the case of the Navier-Stokes linearized operator, this is not the case. Experiments were carried out in which the deflation matrix was chosen to consist of purely real eigenvalues, however, this apporach did not bring forward positive results due to the fact that the most "ill conditioned" eigenvalues are the complex ones. A variation of this approach, known as ADEF1 helps in this problem. This variation projects the deflated eigenvalues to the value one instead of zero. This avoids the "close to zero" round off errors and lets the iterative solver work with values close to one. A convergence plot of this approach is presented next:

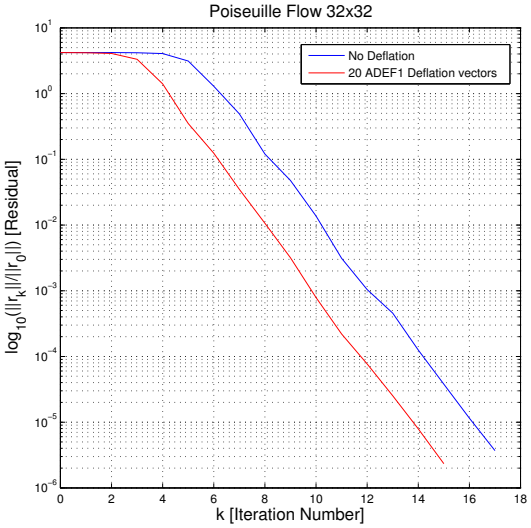
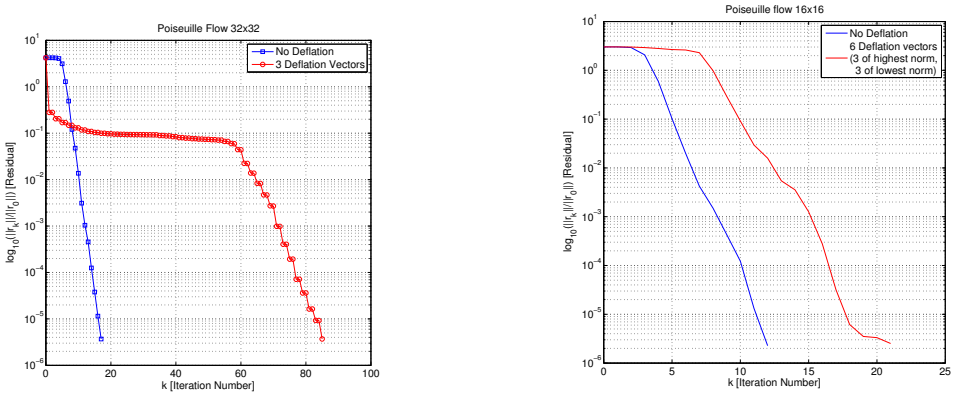


Figure 6.4: Convergence history of the SIMPLER(WBC) preconditioned Navier-Stokes system ($M^{-1}A$) and deflated ($PM^{-1}A$) for the Poiseuille Flow for a 20-vector ADEF1 deflation matrix.

We can see from the figure that the ADEF1 approach also decreases the stagnation behavior and obtains a lower number of total iterations compared to the SIMPLE-preconditioned system. This fact is the main ingredient for the construction of a new two-level preconditioner for the Navier-Stokes equations. In [37], Tang proves a theorem which states that the deflation projector P is defined by the space spanned by the columns of the deflation matrix Z rather than the actual columns. This is reflected in the construction of the deflation matrix Z. In the ideal case, Z should consists of eigenvectors associated with the most unfavorable eigenvalues of $M^{-1}A$. These eigenvalues do not play a role on the convergence behavior of the iterative method so that a faster convergence of the iterative method can be expected. However, the computations of these eigenvectors can be very expensive, and, in addition, these dense vectors may be inefficient is use due to the storage capability and the expensive computations with P. Therefore various techniques have been investigated to find sparse eigenvectors that span the unfavorable eigenspace.

Another deflation approach, referred to as DEF2, would be to apply the projection matrix to the preconditioned matrix $M^{-1}A$. This however is not possible in our case. Since the SIMPLER(WBC) is implemented as an algorithm, the application of the projection matrix would change the structure of the matrix, breaking the structure needed for the SIMPLER algorithm to function properly. A convergence plot of this approach is presented next:



(a) 3 deflation vectors

(b) 6 deflation vectors

Figure 6.5: Convergence History of the SIMPLER(WBC) preconditioned Navier-Stokes system ($M^{-1}A$) and deflated ($PM^{-1}A$) for the Poiseuille Flow.

This approach needs to be investigated in more detail in order to fully understand its drawbacks and benefits. It seems as if the stagnation behavior is "cured" for the first few iterations, however a greater stagnation phase is encountered shortly afterwards. The previous image reminds of the qualitative behavior of the iterative solver when used to solve matrices generated by stretched finite elements. On the next subsection we will discuss the stability of finite elements on such non-uniform meshes.

6.1.2 Stretched grids

The stagnation behavior was also clearly found in the case of stretched grids. As it was done in the last chapter, the backward facing step simulations with stretched grids was investigated for both SIMPLER(WBC) and SIMPLER(WOBC). Just as in the case for unstretched grids, if we apply the deflation scheme on the stretched grid problems preconditioned with SIMPLER(WBC) we obtain a reduction of the initial stagnation phase of the iterative solver, the following convergence graph exemplifies this results:

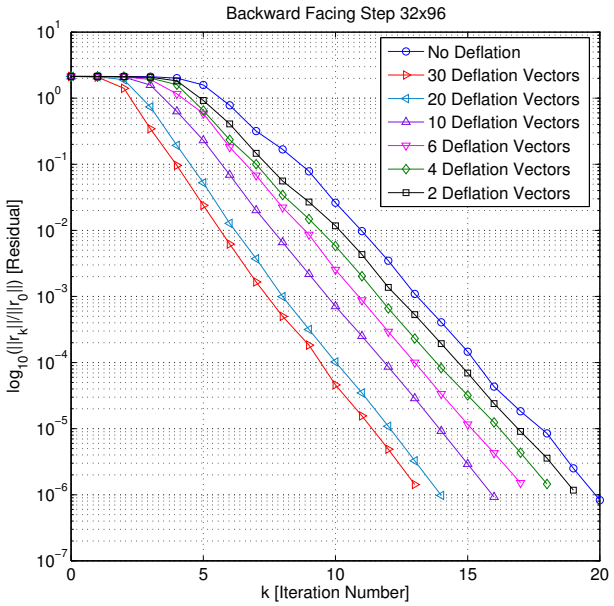


Figure 6.6: Convergence History of the SIMPLER(WBC) preconditioned Navier-Stokes system ($M^{-1}A$) and deflated ($PM^{-1}A$) for the Backward Facing Step for different sizes of the deflation matrix with stretched grid factor of 3

Once again, we find a positive reduction of the stagnation behavior when a large deflation matrix is used. However, when using SIMPLER(WOBC) the result of applying deflation is counterproductive. The following graph shows the comparison for a problem with stretch factor of 3. Most unintuitively, even though the conditions for fast convergence of the iterative method are met, that is a clustered spectrum and positive eigenvalues, the method fails absolutely. The results are shown next:

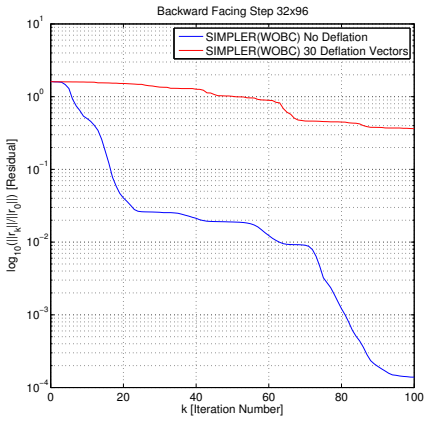


Figure 6.7: Convergence History of the SIMPLER(WOBC) preconditioned Navier-Stokes system ($M^{-1}A$) and deflated ($PM^{-1}A$) for the Backward Facing Step with stretched grid factor of 3.

The previous plot shows that even after deflating the system with 30 eigenvectors, choosing the ones that have the highest norm, the stagnation behavior becomes dominant, and the results resemble the solution of an un-preconditioned matrix. The next section gives an alternative, a theoretical way of approaching the elimination of the stagnation behavior for stretched grids.

From the collection of all results obtained in order to have a better understanding of the initial stagnation behavior of the SIMPLER preconditioner, it is clear that a deflation approach tackles the problem. However, the results also show that the approach should be "selective". The "ill conditioned" eigenvalues that can be identified by analyzing how the Arnoldi algorithm approximates the eigenvalues of the system are not enough for the deflation scheme to be completely successful.

6.2 Stability of Finite Elements

In order to understand the stagnation behavior more commonly present in problems with stretched grids, we turn the discussion towards an analysis of the inf-sup stability condition of finite elements. The approach followed in this subsection is completely assimilated from the investigations done by Elman, Silvester and Liao in [25], [26], [12]. The stability of such elements has been only in literature for some months, and implementation is needed to corroborate the assumptions taken. As we have seen in Section 2, the so-called LBB condition, better known as the *inf-sup* condition is given by:

$$\inf_{q \neq \text{constant}} \sup_{\mathbf{v} \neq \mathbf{0}} \frac{|(q, \nabla \cdot)\mathbf{v}|}{\|\mathbf{v}\|_{1,\Omega} \|q\|_{0,\Omega}} \geq \gamma > 0 \tag{6.1}$$

where $\|\mathbf{v}\|_{1,\Omega} = (\int_\Omega \mathbf{v} \cdot \mathbf{v} + \nabla \mathbf{v} : \nabla \mathbf{v})^{1/2}$ is a norm of functions in $\mathbf{H}^1_{E_0}$, and $\|q\|_{0,\Omega} = \|q - (1/|\Omega|) \int_\Omega q\|$ is a *quotient space* norm. This condition can be viewed as a coercivity estimate for the bilinear form $b(\mathbf{v}, q) := \int_\Omega q \nabla \cdot \mathbf{v}$ on the spaces $L_2(\Omega)$ and $\mathbf{H}^1_{E_0}$, and it

is used to establish the existence of a pressure solution. (A bilinear form is said to be coercive if there is a constant c such that for all $\mathbf{u}$ $b(\mathbf{u},\mathbf{u}) \geq c\|\mathbf{u}\|^2$). The inf-sup condition can be used to ensure:

$$\int_\Omega p\nabla\cdot\mathbf{v} = 0 \text{ for all } \mathbf{v}\in\mathbf{H^1_{E_0}} \Rightarrow \{p = \textit{constant} \text{ if } \partial\Omega = \partial\Omega_D; p = 0 \text{ otherwise}\}. \quad (6.2)$$

In other words, the inf-sup condition is a sufficient condition for the pressure to be unique up to a constant in enclosed flow. The discrete analog of the inf-sup condition, referred to as *uniform inf-sup stability* or *the inf-sup condition*, requires the existence of a positive constant γ independent of h such that, for any conceivable grid,

$$\inf_{q_h\neq\text{constant}} \sup_{\mathbf{v}_h\neq\mathbf{0}} \frac{|(q_h,\nabla\cdot)\mathbf{v}_h|}{\|\mathbf{v}_h\|_{1,\Omega}\|q_h\|_{0,\Omega}} \geq \gamma. \quad (6.3)$$

This condition provides a guarantee that the discrete solvability condition:

$$\int_\Omega p_h\nabla\cdot\mathbf{v}_h = 0 \text{ for all } \mathbf{v}_h\in\mathbf{X}_0^h \Rightarrow p_h = \text{constant}, \quad (6.4)$$

holds is a delicate matter. In general, care must be taken to make the velocity space rich enough compared to the pressure space, otherwise the discrete solution will be "over-constrained".

One of the most elegant ways of establishing inf-sup stability is to use a *macroelement* construction. The idea is to consider "local enclosed-flow Stokes problems (Oseen problems)" posed on a subdomain $\mathcal{M}$ of Ω consisting of a small patch of elements arranged in a simple topology. This subdomain is referred to as a *macroelement*, and if the discrete solvability condition (6.4) is satisfied on it, it is called a *stable macroelement*. Theoretical results show that for any grid that can be constructed by "patching together" such stable macroelements, the inf-sup condition (6.3) is satisfied. The method of analysis is to establish the discrete solvability condition (6.4) on macroelements, and then to establish a connectivity condition showing that (6.3) holds on the union of macroelement patches. We first address the question of weather a macroelement consisting of a single rectangular element of the Taylor-Hood $\mathbf{Q}_2-\mathbf{Q}_1$ approximation is stable.

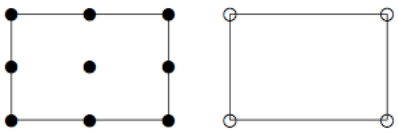


Figure 6.8: $\mathbf{Q}_2-\mathbf{Q}_1$ element (• velocity, ∘ pressure).

Let $\mathcal{M}=\square_k$ denote such an element, with the local numbering as illustrated in Figure 6.9. For an enclosed flow problem posed over $\mathcal{M}$, there is one interior velocity node(9, marked with •), and four pressure nodes namely 1,2,3, and 4.

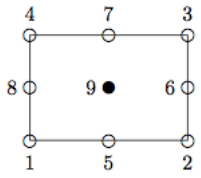


Figure 6.9: Numbering for a single element macroelement

To investigate the stability of the macroelement, we need to construct the matrix $B = [B_x B_y]$ using its definitions and then, in light of (6.4), check to see if $\text{NULL}(B^T) = \{\mathbf{1}\}$. In this particular case, $\mathbf{v}$ has dimension 2 for the two velocity components defined on node 9, and $\mathbf{p}$ has dimension 4. In this case, B_x^T and B_y^T are 1×4 vectors,

$$B_x^T = [b_{x,j}], \quad b_{x,j} = -\int_{\square_k} \psi_j \frac{\partial \phi_9}{\partial x}, \tag{6.5}$$

$$B_y^T = [b_{y,j}], \quad b_{y,j} = -\int_{\square_k} \psi_j \frac{\partial \phi_9}{\partial y}, \tag{6.6}$$

where ϕ_9 is the standard biquadratic basis function $(ax^2 + bx + c)(dy^2 + ey + f)$. This function takes the value one at node 9 and is zero at the other eight nodes. Similarly $\psi_j : j = 1, 2, 3, 4$ are the standard bilinear basis functions defined on $\square_k$, we can evaluate the previous integrals by first integrating by parts, and then using the tensor product form of Simpson's rule:

$$\int_{\square_k} f(x, y) = \frac{|\square_k|}{36}(f_1 + f_2 + f_3 + f_4 + 4f_5 + 4f_6 + 4f_7 + 4f_8 + 16f_9),$$

with f_i representing $f(\mathbf{x}_i)$. This gives:

$$b_{x,j} = -\int_{\square_k} \phi_9 \frac{\partial \psi_j}{\partial x} = \frac{4}{9}|\square_k| \frac{\partial \psi_j}{\partial x}(\mathbf{x_9}), \tag{6.7}$$

$$b_{y,j} = -\int_{\square_k} \phi_9 \frac{\partial \psi_j}{\partial y} = \frac{4}{9}|\square_k| \frac{\partial \psi_j}{\partial y}(\mathbf{x_9}). \tag{6.8}$$

Evaluating the linear functions $\frac{\partial \psi_j}{\partial x}$, $\frac{\partial \psi_j}{\partial y} : j = 1, 2, 3, 4$ at the centroid, for example

$$\frac{\partial \psi_j}{\partial x}(\mathbf{x}_9) = -\frac{h_y}{2|\square_k|},$$

gives

$$B^T = \begin{bmatrix} B_x^T \\ B_y^T \end{bmatrix} = \begin{bmatrix} -2/9h_y & 2/9h_y & 2/9h_y & -2/9h_y \\ -2/9h_x & -2/9h_x & 2/9h_x & 2/9h_x \end{bmatrix}. \tag{6.9}$$

Note that the discrete operator B_x^T represents a scaled central difference approximation of the pressure x-derivative at the node $\mathbf{x}_9$:

$$0 = \frac{2}{9}h_y(-p_1 + p_2 + p_3 - p_4) \tag{6.10}$$

$$= \frac{4}{9}|\square_k| \frac{(p_3+p_2)/2-(p_1+p_4/2)}{h_x} \approx \frac{4}{9}|\square_k| \frac{\partial p}{\partial x}(\mathbf{x_9}). \tag{6.11}$$

Finally, the linear system $B^T\mathbf{p} = \mathbf{0}$ is equivalent to

$$-p_1 + p_2 + p_3 - p_4 = 0 \tag{6.12}$$

$$-p_1 - p_2 + p_3 + p_4 = 0 \tag{6.13}$$

which is satisfied whenever $p_1 = p_3$ and $p_2 = p_4$. That is, B^T has a two dimensional null space and the macroelement $\mathcal{M}$ in Figure 6.9 is not stable.
This derivation shows that the macroelement test fails when it is applied to a single element of the $\mathbf{Q}_2 - \mathbf{Q}_1$ discretization, so that the single element is not a macroelement. This does not mean that the discretization itself is unstable. Consider now the *two element* patch $\mathcal{M} = \square_k \bigcup \square_m$, illustrated in Figure 6.10.

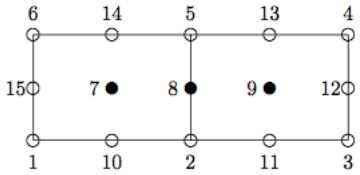


Figure 6.10: $\mathbf{Q}_2 - \mathbf{Q}_1$ macroelement numbering for two-element patch $\square_k \bigcup \square_m$.

For an enclosed flow problem posed over $\mathcal{M}$, there are three interior velocity nodes (7, 8, and 9), and six pressure nodes (1, 2, 3, 4, 5, and 6). In this case, the matrix B^T is a 6×6 matrix, which offers the hope that the macroelement may be stable. To check this, we first note that the one-element patch matrix (6.9) gives the four rows of the stability system $B^T\mathbf{p} = 0$ that correspond to the velocity nodes 7 and 9. Specifically, the node 7 equations imply that $p_1 = p_5$ and $p_2 = p_6$, whereas the node 9 equations imply that $p_2 = p_4$ and $p_3 = p_5$. Combining these four equations implies that

$$p_1 = p_3 = p_5 = \xi, \tag{6.14}$$

$$p_2 = p_4 = p_6 = \eta. \tag{6.15}$$

To establish stability, we have to show that $\xi = \eta$ using one of the two equations associated with velocity node 8.
We first consider the vertical velocity component since it represents a discrete approximation of $\partial p/\partial y = 0$ at the node $\mathbf{x}_8$. This suggests that the column entries corresponding to nodes 1, 3, 4, and 6 will be zero. To see this, consider the coefficients of p_1,

$$B^T_{y,k1} = -\int_{\mathcal{M}} \psi_1 \frac{\partial \phi_8}{\partial y} \tag{6.16}$$

$$= -\int_{\square_k} \psi_1 \frac{\partial \phi_8}{\partial y} \quad (\psi_1 = 0 \text{ in } \square_m) \tag{6.17}$$

$$= -\int_{\square_k} \phi_8 \frac{\partial \psi_1}{\partial y} \quad (\text{since } \phi_8 n_y = 0 \text{ on the boundary}) \tag{6.18}$$

$$= \frac{1}{9}|\square_k| \frac{\partial \psi_1}{\partial y}(\mathbf{x}_8) \quad (\text{using Simpson's rule}) \tag{6.19}$$

$$= 0, \tag{6.20}$$

since $\psi_1 = 0$ on the vertical edge between nodes 2 and 5. By evaluating the other coefficients in the same way, it can be shown that

$$(B_y^T)_k = [\ 0 \quad -\tfrac{2}{9}h_x \quad 0 \quad 0 \quad \tfrac{2}{9} \quad 0]. \tag{6.21}$$

In the system $B^T\mathbf{p} = 0$, the equation corresponding to (6.21) implies that $p_2 = p_5$. This means that $\xi = \eta$, and $\text{NULL}(B^T) = \mathbf{1}$ as required.

The analysis above shows that two-element patches of $\mathbf{Q}_2 - \mathbf{Q}_1$ elements are stable. Once stability has been established on a reference macroelement, it holds for all patches of elements with the same topology. Moreover, any grid consisting of an even number of elements can be decomposed into two-element patches. This means that $\mathbf{Q}_2 - \mathbf{Q}_1$ approximation is stable on *every* possible grid with an even number of grid points.

Chapter 7

Conclusions and future Research

7.1 Conclusion

The study of preconditioners for the Navier-Stokes equations is still a field of intensive research in contemporary computational fluid dynamics (CFD). The main goal of this area of research is to develop solvers that are both fast and robust, however, in the case of Navier-Stokes problems, the ultimate goal is to develop preconditioners that provides mesh and Reynolds number independent convergence. Towards achieving this goal, this thesis is devoted to the study of block-preconditioners with special attention to SIMPLE-type preconditioners. We focus our attention on building preconditioners to accelerate Krylov subspace methods, and two questions related to previously established problems in literature were the main points of research, mainly:

- Why is there a stagnation phase in the iterative solution of the SIMPLE-preconditioned Navier-Stokes algebraic system?
- Why does the number of iterations increase for stretched grids?

In the first three chapters, the theoretical bases of this field of research are laid out. The FEM discretization of the N-S system of equations is presented in the first chapter together with appropriate linearization techniques. In Chapter 2, a collection of concepts in linear algebra are presented. The theory behind Krylov subspace methods as well as a brief explanation of preconditioning and deflation techniques is explained. In the fourth chapter, a review of block-type preconditioners for the N-S equations is reviewed, and the setting of the main problems of this thesis is shown. In Chapter 5, a collection of numerical experiments in order to answer the main questions of this research project is presented. In the second to last chapter, a discussion on the results obtained in the numerical simulations is put forth. A theoretical answer to the main research questions was provided in both cases, and a practical answer was found in one case. The answer of the question regarding the stagnation behavior in the solution phase of the `GMRES` method when solving the

SIMPLE-preconditioned N-S system of equations lies in the investigation of the spectrum of eigenvalues of the preconditioned system. By looking at the Ritz-value approximation at each Arnoldi iteration step of the `GMRES` algorithm, it was found that the innermost eigenvalues, are the ones which are approximates last, while the eigenvalues with largest real part are the first to be approximated. This observation helped in the construction of a deflation matrix for which these eigenvalues are projected out of the spectrum of the preconditioned matrix. The deflation approach was followed and the stagnation behavior was eliminated in its majority. The remaining stagnation behavior was found to be caused by the round off error approximation of the "close to zero" deflated eigenvalues. In order to circumvent this problem, the ADEF1 deflation strategy was implemented in order to project the undesired eigenvalues towards the value one, hence eliminating the problems of "close to zero" round off errors. This approach was also successful.
The success of this approach allows us to suggest the construction of a *two-level preconditioned `GMRES` method* for solving the Navier-Stokes equations. Since the deflation approach DEF1, which projects the undesired eigenvalues to the value zero, is substituted by the ADEF1 approach, who projects the undesired eigenvalues to the value one, we can view the approach followed in this thesis as an algorithm to construct a two-level preconditioner for the solution of the linearized version of the N-S system of equations which provides Reynolds independent convergence but which is mesh-size dependent.
Also, as a side result originating from the numerical experiments and by observing the difference in in the behavior of the iterative solver when dealing with Neumann or Dirichlet problems, an educated way of constructing the Schur complement depending on the problem type is presented which eliminates the qualitative stagnation behavior but increases the total number of iterations in the solution process.

When dealing with stretched grids, the deflation approach is also successful. We can observe a decrease in the total number of iterations when a reasonable deflation matrix is constructed. However, a theoretical answer to the question regarding the increase in iterations when using stretched grids was found in present year literature. The answer is found when studying the theoretical effects of applying a stretching to the grid. It was found that the *inf-sup* condition of the elements is no longer satisfied in this case. However, it was also found that if a macroelement construction is used in the discretization of the problem, the inf-sup condition holds for any type of conceivable grid. This macroelement construction suggested by Elman, Silvester and Wathen in [11] and tested by Liao in [26] allows the element discretization to remain stable even in the case of extreme mesh stretching.

7.2 Future Research

The work done in this master thesis project reveals some issues which need to be examined in greater detail in order to obtain a full understanding of the answers to the research questions. A direct continuation of the direction of research of this thesis would focus on the next points:

- Formalize the concepts presented in this thesis and construct a two-level preconditioned SIMPLER algorithm.
- Investigate the effect of using different deflation techniques (different approaches of constructing the deflation matrix) applied to the SIMPLER-preconditioned Navier-Stokes matrix for stretched and non-stretched grids in the efficiency of the solution.
- Investigate the theory of deflation techniques in order to find the optimal way of choosing the deflation vectors and construct a more efficient deflation matrix.
- Implement the macroelement construction in the FEM discretization of the Navier-Stokes system of equations.
- Investigate the possibilities of parallel implementation of the SIMPLE-type preconditioners together with a deflation approach.

The study and solution of these issues would be of great importance for the creation of a better preconditioner for the Navier Stokes system of equations. A fast and efficient method of solving this system of equations would benefit not only the scientific knowledge but it may bring advances to any industry who performs computer fluid dynamics (CFD) simulations.

Bibliography

[1] J. Baglama, D. Calvetti, G.H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. *SIAM Journal on Scientific Computing*, 20(1):243–269, January 1998.

[2] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, November 2002.

[3] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, May 2005.

[4] M. Benzi and M. Tuma. A sparse approximate inverse preconditioner for nonsymetric linear systems. *SIAM Journal on Scientific Computing*, 19(3):968–994, 1998.

[5] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4:43–66, 1997.

[6] A.C. de Niet and F.W. Wubs. Two preconditioners for saddle point problems in fluid flows. *International Journal for Numerical Methods in Fluids*, 54(4):355–377, 2007.

[7] C.R. Dohrmann and P.B. Bochev. A stabilized finite element method for the Stokes problem based on polynomial pressure projections. *International Journal for Numerical Methods in Fluids*, 46(2):183–201, September 2004.

[8] H.C. Elman. Preconditioning for the steady-state Navier-Stokes equations with low viscosity. *SIAM Journal on Scientific Computing*, 20(4):1299–1316, January 1999.

[9] H.C. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing*, 27(5):1651–1668, January 2006.

[10] H.C. Elman and A. Ramage. An analysis of smoothing effects of upwinding strategies for the convection-diffusion equation. *SIAM Journal on Numerical Analysis*, 40(1):254–281, January 2002.

[11] H.C. Elman, A. Ramage, and D.J. Silvester. Algorithm 866: IFISS, a Matlab toolbox for modelling incompressible flow. *ACM Transactions on Mathematical Software*, 33(2):14–34, June 2007.

[12] H.C. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers*. Oxford University Press, New York, NY, 2006.

[13] H.C. Elman, D.J. Silvester, and A. Ramage. Incompressible Flow Iterative Solution Software (IFISS) installation & software guide version 3.2. pages 1–18, 2012.

[14] H.C. Elman, D.J. Silvester, and A.J. Wathen. Performance and analysis of saddle point preconditioners for the discrete steady-state Navier-Stokes equations. *Numerische Mathematik*, 90(4):665–688, February 2002.

[15] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69:303–318, 1995.

[16] V. Faber, J. Liesen, and P. Tichý. The Faber-Manteuffel theorem for linear operators. *SIAM Journal on Numerical Analysis*, 46(3):1323–1337, January 2008.

[17] M. Fortin. Old and new finite elements for incompressible flows. *International Journal for numerical methods in fluids*, 1(March):347–364, 1981.

[18] A. Gaul, M.H. Gutknecht, J. Liesen, and R. Nabben. A framework for deflated and augmented Krylov subspace methods. *Arxiv*, pages 1–24, 2012.

[19] L. Giraud, S. Gratton, X. Pinel, and X. Vasseur. Flexible GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 32(4):1858–1878, 2010.

[20] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, PA, 1987.

[21] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier-Stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, 2002.

[22] S.A. Kharchenko. Eigenvalue translation based preconditioners for the GMRES(k) method. *Numerical Linear Algebra with Applications*, 2(September 1992):51–77, 1995.

[23] C.M. Klaij and C. Vuik. SIMPLE-type preconditioners for cell-centered, colocated finite volume discretization of incompressible Reynolds-averaged Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 17:830–849, 2013.

[24] C. Li and C. Vuik. Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow. *Numerical Linear Algebra with Applications*, 11(56):511–523, June 2004.

[25] Q. Liao. *Error Estimation and Stabilization for Low Order Finite Elements*. PhD thesis, The University of Manchester, 2010.

[26] Q. Liao and D. Silvester. Robust stabilized Stokes approximation methods for highly stretched grids. *IMA Journal of Numerical Analysis*, 33(2):413–431, July 2012.

[27] J. Liesen and Z. Strakos. GMRES convergence analysis for a convection-diffusion model problem. *SIAM Journal on Scientific Computing*, 26(6):1989–2009, January 2005.

[28] J. Liesen and Z. Strakos. *Krylov Subspace Methods Principles and Analysis*. Oxford University Press, 2012.

[29] M.A. Olshanskii. An iterative solver for the Oseen problem and numerical solution of incompressible Navier-Stokes equations. *Numerical Linear Algebra with Applications*, 6(5):353–378, July 1999.

[30] M.A. Olshanskii and Y.V. Vassilevski. Pressure Schur complement preconditioners for the discrete Oseen problem. *SIAM Journal on Scientific Computing*, 29(6):2686–2704, 2007.

[31] Y. Saad. Preconditioning techniques for nonsymmetric and indefinite linear systems. *Journal of Computational and Applied Mathematics*, 24(1):89–105, November 1988.

[32] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 1993.

[33] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.

[34] A. Segal. Finite element methods for the incompressible Navier-Stokes equations. Technical report, Delft University of Technology, 2012.

[35] A. Segal, M. ur Rehman, and C. Vuik. Preconditioners for incompressible Navier-Stokes solvers. *Numerical Mathematics: Theory, Methods and Applications*, 3(3):245–275, 2010.

[36] V. Simoncini and D.B. Szyld. Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14:1–59, 2007.

[37] J.M. Tang. *Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems*. PhD thesis, Delft University of Technology, 2008.

[38] M. Tismenetsky. A new preconditioning technique for solving large sparse linear systems. *Linear Algebra and its Applications*, pages 331–353, 1991.

[39] M. ur Rehman. *Fast Iterative Methods for The Incompressible Navier-Stokes Equations*. Phd thesis, Delft University of Technology, 2010.

[40] M. ur Rehman, T. Geenen, C. Vuik, G. Segal, and S.P. MacLachlan. On iterative methods for the incompressible Stokes problem. *International Journal for Numerical Methods in Fluids*, 65:1180–1200, 2011.

[41] M. ur Rehman, C. Vuik, and G. Segal. A comparison of preconditioners for incompressible Navier-Stokes solvers. *International Journal for Numerical Methods in Fluids*, 57:1731–1751, 2008.

[42] M. ur Rehman, C. Vuik, and G. Segal. Preconditioners for the steady incompressible Navier-Stokes problem. *International Journal of Applied Mathematics*, 38(4):1–10, 2008.

[43] M. ur Rehman, C. Vuik, and G. Segal. SIMPLE-type preconditioners for the Oseen problem. *International Journal for Numerical Methods in Fluids*, 61(4):432–452, 2009.

[44] H.A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[45] H.A. Van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, July 1994.

[46] C. Vuik, A. Saghir, and G.P. Boerstoel. The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces. *International Journal for Numerical Methods in Fluids*, 33:1027–1040, 2000.

[47] M. Wang and L. Chen. Multigrid methods for the Stokes equations using distributive Gaussâ€“Seidel relaxations based on the least squares commutator. *Journal of Scientific Computing*, February 2013.

[48] S.Ø. Wille and A.F.D. Loula. A priori pivoting in solving the Navier-Stokes equations. *Communications in Numerical Methods in Engineering*, 18(10):691–698, August 2002.

List of Tables

List of Figures

List of Algorithms