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 ECHEVERRIA SERUR
Delft, the Netherlands
February 2013

TU Delft

MSc THESIS APPLIED MATHEMATICS

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

CARLOS ECHEVERRIA SERUR

Delft University of Technology

Daily supervisor	**Responsible professor**
Prof. Dr. Ir. C. Vuik	Prof. Dr. Ir. C. Vuik

Other thesis committee members

Dr. Ir. M. B. van Gijzen	...
...	...

February 2013

Delft, the Netherlands

Contents

1 Introduction

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. Due to the nonlinear character of the behavior of fluids, the solution of the (N-S) system of equations requires a suitable linearization of the discrete formulation of the original system. The resulting linear system of equations gives rise to a so-called saddle-point problem. The efficient solution of this linear system of equations is of primary interest 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. In the following report we adopt an iterative approach to solving this linear system, mainly by the use of a Krylov subspace method combined with a preconditioned linear system of equations. In particular, the type of preconditioners studied are known as *Block-Preconditioners* in literature.

In Section 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 methods. Lastly, a brief comment on finite element selection is given. In Section 3, the main concepts of linear algebra are presented. The theory behind Krylov subspace methods is discussed and the general theory of Preconditioners is presented. In Section 4, specific Block-type preconditioners for the Navier Stokes equations are studied including preconditioners based on approximate commutators and SIMPLE-type preconditioners. In Section 5, a set of problems (benchmarks) that make use of the theory presented in the previous sections is presented. 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. Finally, in Section 6, the direction of research for the master thesis project is given.

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

2 Discretization and linearization of the N-S 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 of 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

$$-\nu\nabla^2\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \text{ in } \Omega \tag{2}$$

Equation (1) represents the conservation of momentum, while equation (2) represents the incompressibility condition, or mass conservation. The boundary value problem that is considered is the system composed of equations (1) and (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

$$\mathbf{u} = \mathbf{w} \text{ on } \partial\Omega_D, \quad \nu\frac{\partial\mathbf{u}}{\partial n} - \mathbf{n}p = \mathbf{s} \text{ on } \partial\Omega_N. \tag{3}$$

The weak formulation of the Navier-Stokes equations is given by:

$$\nu\int_\Omega(\nabla^2\mathbf{u})\cdot\mathbf{v} + \int_\Omega(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} - \int_\Omega\nabla p\cdot\mathbf{v}) = \int_\Omega\mathbf{f}\cdot\mathbf{v} \tag{4}$$

$$\int_\Omega(\nabla\cdot\mathbf{u})\cdot q = 0 \tag{5}$$

where $\mathbf{v}$ and q are test functions in velocity and pressure space, respectively. After applying the Gauss divergence theorem and substitution of the boundary conditions, we obtain the equivalent problem:

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

$$\nu \int_\Omega \nabla \mathbf{u} \otimes \nabla \mathbf{v} + \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} - \int_\Omega p(\nabla \cdot \mathbf{v}) = \int_{\partial \Omega_N} \mathbf{s} \cdot \mathbf{v} \tag{6}$$

$$\int_\Omega q \cdot (\nabla \cdot \mathbf{u}) = 0 \tag{7}$$

where H_E^1 denotes the Sobolev space of functions whose generalized derivatives are in $L_2(\Omega)$. The symbol $\otimes$ denotes the outer product. The discrete version of equations (6) and (7) is formulated as:

Given the finite dimensional subspaces $\mathbf{X}_0^h \subset H_{E_0}^1, \mathbf{X}^h \subset H_E^1$ and $M^h \subset L_2(\Omega)$, find $\mathbf{u}_h \in \mathbf{X}_E^h$ and $p_h \in M^h$ such that:

$$\nu \int_\Omega \nabla \mathbf{u}_h \otimes \nabla \mathbf{v}_h + \int_\Omega (\mathbf{u}_h \cdot \nabla \mathbf{u}_h) \cdot \mathbf{v}_h - \int_\Omega p_h(\nabla \cdot \mathbf{v}_h) = \int_{\partial \Omega_N} \mathbf{s} \cdot \mathbf{v}_h \text{ for all} \mathbf{v}_h \in \mathbf{X}_0^h \tag{8}$$

$$\int_\Omega q_h \cdot (\nabla \cdot \mathbf{u}_h) = 0 \text{ for all } q_h \in M^h \tag{9}$$

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

$$p_h = \sum_{j=1}^{n_p} p_j \psi_j(x), \quad n_p \text{ is the number of pressure unknowns} \tag{10}$$

and

$$\mathbf{u}_h = \sum_{j=1}^{\frac{n_u}{2}} u_{1j}\phi_{j1}(x) + u_{2j}\phi_{j2}(x) = \sum_{j=1}^{n_u} u_j \phi_j(x) \tag{11}$$

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

$$\nu \int_\Omega \nabla \mathbf{u}_h \otimes \nabla \phi_i + \int_\Omega (\mathbf{u}_h \cdot \nabla \mathbf{u}_h) \cdot \phi_i - \int_\Omega p_h(\nabla \cdot \phi_i) = \int_{\partial \Omega_N} \mathbf{s} \cdot \phi_i \text{ for all} i = 1, ..n_u, \tag{12}$$

$$\int_\Omega \phi_i \cdot (\nabla \cdot \mathbf{u}_h) = 0 \text{ for all } i = 1, ...n_p. \tag{13}$$

This system of equations can be represented in matrix form as

$$A_d u + N(u) + B^T p = f, \tag{14}$$

$$Bu = g, \tag{15}$$

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:

$$-\nu\Delta u^{k+1} + (u^k \cdot \nabla)u^{k+1} + \nabla p^{k+1} = f \text{ in } \Omega, \tag{16}$$

$$\nabla \cdot u^{k+1} = 0 \text{ in } \Omega, \tag{17}$$

In this approach, the nonlinear term is substituted by an approximation including the velocity vector calculated at distinct time steps, 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 time-step is the sum of the velocity field at the previous time step plus a correction, that is:

$$u^{k+1} = u^k + \delta u^k. \tag{18}$$

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

$$-\nu\Delta u^k + u^k \cdot \nabla u^{k-1} + u^{k-1} \cdot \nabla u^k + \nabla p^k = f - u^{k-1} \cdot \nabla u^{k-1} \text{ in } \Omega, \tag{19}$$

$$\nabla \cdot u^k = 0 \text{ in } \Omega. \tag{20}$$

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

$$Fu + B^T p = f, \tag{21}$$

$$Bu = g, \tag{22}$$

where $F = A_d + N(u^k)$ is the linearized operator and u^k is the solution of the velocity one iteration before.

The linear system arising from the linearization can be written as:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{23}$$

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

$$\begin{aligned} \min : J(x) &= \frac{1}{2} u^T A u - f^T u \\ \text{subject to: } & Bu = g. \end{aligned}$$

In this case the variable p represents the vector of Lagrange multipliers. Any solution (u^*, p^*) of (23) 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 (23). The zero block reflects the absence of the pressure in the continuity equation. As a consequence the system of equations may be underdetermined for an arbitrary combination of pressure and velocity unknowns [2].

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 (22) and (21) contain more rows than unknowns 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 is derived. This condition is known under the name Brezzi-Babuŝka condition (BB condition). However, the BB condition is rather abstract and in practice it is very difficult to verify whether the BB condition is satisfied or not. Fortin (1981) has given a simple method to check the BB 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.*

3 Projection techniques, Krylov subspace methods, and Preconditioners

As we have seen in the previous section, 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. If we consider the most general form of (23), the resulting matrix equation is in the form:

$$Ax = b, \tag{24}$$

where A is an $n \times n$ real matrix.

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 [19]. 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 (24) 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:

$$\tilde{x} = x_0 + \delta, \quad \delta \in \mathcal{K}_m, \tag{25}$$

$$(r_0 - A\delta, w) = 0 \quad \forall w \in \mathcal{L}_m. \tag{26}$$

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

$$\tilde{x} = x_0 + Vy, \tag{27}$$

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

$$W^T AVy = W^T r_0. \tag{28}$$

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:

$$\tilde{x} = x_0 + V(W^T AV)^{-1} W^T r_0. \tag{29}$$

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 [19]. 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

$$\tilde{r} = b - A(x_0 - \delta) = r_0 - A - \delta. \tag{30}$$

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*

$$\tilde{r} = (I - P)r_0, \tag{31}$$

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 (30) 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 $\mathcal{K}$:

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

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*

$$\tilde{d} = (I - P_A)d_0, \tag{32}$$

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

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0\}. \tag{33}$$

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 Gramm-Schmidt (MGS) algorithm is as follows:

Algorithm: Arnoldi with MGS

1. Chose a vector v_1 such that $\|v_1\|_2 = 1$
2. For $j = 1, 2, ..., m$, Do
3. Compute $w_j := Av_j$
4. For $i = 1, ..., j$, Do
5. $h_{ij} = (w_j, v_i)$
6. $w_j := w_j - h_{ij}v_i$
7. End Do
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 Do

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

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

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

$$AV_j = V_j H_j + w_j e_j^T \tag{34}$$

$$= V_{j+1}\bar{H}_j, \tag{35}$$

$$V_j^T A V_j = H_j \tag{36}$$

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 (33) 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

$$V_m^T A V_m = H_m \tag{37}$$

as a consecuence of equation (36), and

$$V_m^T r_0 = V_m^T(\beta v_1) = \beta e_1. \tag{38}$$

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

$$x_m = x_0 + V_m y_m \;\; y_m = h_m^{-1}(\beta e_1). \tag{39}$$

A method based on this approach is called the full orthogonalization method (FOM), presented in [19].

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 mth 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 (35). Any vector x in $x_0 + \mathcal{K}_m$ can be written as

$$x = x_0 + V_m y, \tag{40}$$

where y is an m-vector. Defining

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2, \tag{41}$$

the relation (35) 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 ortonormal, then

$$J(y) \equiv \|b - A(x_0 + V_m y)\|_2 = \|\beta e_1 - \bar{H}_m y\|_2. \tag{42}$$

The GMRES approximation is the unique vector of $x_0 + \mathcal{K}_m$ that minimizes (41). By (40) and (42), 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.

$$x_m = x_0 + V_m y_m, \text{ where} \tag{43}$$

$$y_m = \min_y \|\beta e_1 - \bar{H}_m y\|_2. \tag{44}$$

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: GMRES

1. Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2. For $j = 1, 2, ..., m$, Do
3. Compute $w_j := Av_j$
4. For $i = 1, ..., j$, Do
5. $h_{ij} = (w_j, v_i)$
6. $w_j := w_j - h_{ij} v_i$
7. End Do
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 Do
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 umber 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*

$$(Ap_i, Ap_k) = 0, \quad \text{for } i \neq k. \tag{45}$$

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

$$x_m = x_0 + \sum_{i=0}^{m-1} \frac{(r_0, Ap_i)}{(Ap_i, Ap_i)} p_i. \tag{46}$$

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

$$x_m = x_{m-1} + \frac{(r_{m-1}, Ap_{m-1})}{(Ap_{m-1}, Ap_{m-1})} p_{m-1}. \tag{47}$$

This lemma opens up many different ways to obtain algorithms that are mathematically equivalent to the 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 (47). This is called the generalized CR (GCR) algorithm.

Algorithm: GCR

1. Compute $r_0 = b - Ax_0$. Set $p_0 = r_0$
2. For $j = 0, 1, ...,$ until convergence, Do
3. Compute $\alpha_j = \frac{r_j, Ap_j}{(Ap_j, Ap_j)}$
4. $x_{j+1} = x_j + \alpha_j p_j$
5. $r_{j+1} = r_j - \alpha_j Ap_j$
6. Compute $\beta_{ij} = \frac{(Ar_{j+1}, Ap_i)}{(Ap_i, Ap_i)}$ for $i = 0, 1, ..., j$
7. $p_{j+1} = r_{j+1} + \sum_{i=0}^{j} \beta_{ij} p_i$
8. End Do

To compute the scalars β_{ij} in the above algorithm, the vector Ar_j and the previous Ap_i's are required. In order to limit the number of matrix-by-vector products per step to one, we can proceed as follows. Follow line 5 with a computation of Ar_{j+1} and then compute Ap_{j+1} after line 7 from the relation

$$Ap_{j+1} = Ar_{j+1} + \sum_{i=0}^{j} \beta_{ij} Ap_i.$$

Both the set of p_i's and the set of Ap_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 that with GMRES [19].

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 easier 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 [19].

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

$$A = M - N, \tag{48}$$

where A is associated with the linear system (24). A *linear fixed-point iteration* can be defined by the recurrence

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b, \tag{49}$$

which at the same time is of the form

$$x_{k+1} = Gx_k + f, \tag{50}$$

with

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

The iteration (50) can be viewed as a technique for solving the system

$$(I - G)x = f. \tag{51}$$

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

$$M^{-1}Ax = M^{-1}b. \tag{52}$$

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

$$M^{-1}Ax = M^{-1}b. \tag{53}$$

Alternatively, it can also be applied to the right:

$$AM^{-1}u = b, \quad x \equiv M^{-1}u. \tag{54}$$

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 incomplete Cholesky factorization. In this situation, the preconditioning can be split:

$$M_L^{-1}AM_R^{-1}u = M_L^{-1}b, \quad x \equiv M_R^{-1}u. \tag{55}$$

It is of utmost importance to preserve symmetry whenever the original matrix is symmetric. The straight forward way of preserving symmetry is by applying the method

described by (55) 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

$$(M^{-1}Ax, y)_M = (Ax, y) = (x, Ay) = (x, M(M^{-1}A)y) = (x, M^{-1}Ay)_M. \tag{56}$$

We can exploit this fact in order to precondition Algorithm 3.3. An alternative is to replace the usual Euclidean inner product in the CG algorithm with the M inner product. If the CG algorithm is rewritten for this new inner product, denoting the original residual by $r_j = b - Ax_j$ and the residual for the preconditioned system by $z_j = M^{-1}r_j$, the following algorithm is obtained:

Algorithm: Preconditioned CG

1. Compute $r_0 = b - Ax_0$, $z_0 = M^{-1}r_0$, and $p_0 := z_0$
2. For $j = 0, 1, ...,$ until convergence, Do
3. $\quad \alpha_j := (r_j, z_j)/(Ap_j, p_j)$
4. $\quad x_{j+1} := x_j + \alpha_j p_j$
5. $\quad r_{j+1} := r_j - \alpha_j Ap_j$
6. $\quad z_{j+1} := M^{-1}r_{j+1}$
7. $\quad \beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
8. $\quad p_{j+1} := z_{j+1} + \beta p_j$
9. End Do

It is interesting to note that since $(z_j, z_j)_M = (r_j, z_j)$ and $(M^{-1}Ap_j, p_j)_M = (Ap_j, p_j)$, the M inner products do not have to be formed explicitly. It is also interesting to observe that $M^{-1}A$ is also self-adjoint with respect to the A inner product:

$$(M^{-1}Ax, y)_A = (AM^{-1}Ax, y) = (x, AM^{-1}Ay) = (x, M^{-1}Ay)_A. \tag{57}$$

In the case of generalized minimal residual (GMRES) or the nonsymetric iterative solvers, the same three options for applying the preconditioning operation as for the CG are available, namely left, split, and right preconditioning. However, the right preconditioning versions will 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

$$AM^{-1}u = b, \quad u = Mx. \tag{58}$$

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. Atthe end, the u variable approximate solution to (58) 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: GMRES with Right Preconditioning

1. Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2. For $j = 1, 2, ..., m$, Do
3. Compute $w := AM^{-1}v_j$
4. For $i = 1, ..., j$, Do
5. $h_{i,j} = (w, v_i)$
6. $w := w - h_{i,j}v_i$
7. End Do
8. Compute $h_{j+1,j} = \|w\|_2$ 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 Do
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 preconditionning 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 algorthm to the flexible variant, described below.

Algorithm: GMRES with Right Preconditioning

1. Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2. For $j = 1, 2, ..., m$, Do
3. Compute $z_j := M_j^{-1} v_j$
4. Compute $w := Az_j$
5. For $i = 1, ..., j$, Do
6. $h_{i,j} = (w, v_i)$
7. $w := w - h_{i,j} v_i$
8. End Do
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 Do
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 preconditionning. 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.

4 Block-type preconditioners for the incompressible N-S 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 Section 4, preconditioning is simply a means of transforming the original linear system into one which has the same solution, but which is likely to be easier 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 (23). 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:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{59}$$

in which B is assumed to be nonsingular. From the first equation the unknown x can be expressed as

$$x = A_{11}^{-1}(f - A_{12}y). \tag{60}$$

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

$$(A_{22} - A_{21}A_{11}^{-1}A_{12})y = g - A_{21}A_{11}^{-1}f. \tag{61}$$

The matrix

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12} \tag{62}$$

is called the *Schur complement* matrix associated with the y variable. If this matrix can be formed and the linear system (59) 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 (60). 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 (23). They are mostly based on a block LDU factorization of (23):

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = LDU = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{63}$$

where S is the Schur complement matrix discussed above. Similarly, Block triangular preconditioners (P_t) are based on the block DU factorization of (23) given by:

$$DU = \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} = P_t = \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \tag{64}$$

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

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} \tag{65}$$

We can see by inspecting the first row of (65) 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:

$$\left(Fu + B^T p\right) = 0 \Rightarrow u = -F^{-1}B^T p \tag{66}$$

From the second row of (65) we obtain:

$$Bu - \lambda S p = 0,$$

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

$$-BF^{-1}B^T p = \lambda S p. \tag{67}$$

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 zero 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 (64) 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: 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 [13] defined on the velocity space can be expressed as:

$$\mathcal{L} = -\nu\nabla^2 + \mathbf{w_h} \cdot \nabla \tag{68}$$

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.* [13] 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:

$$\varepsilon = \mathcal{L}\nabla - \nabla\mathcal{L}_p \ll 1, \tag{69}$$

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

$$\varepsilon_h = (Q_v^{-1}F)(Q_v^{-1}B^T) - (Q_v^{-1}B^T)(Q_p^{-1}F_p) \tag{70}$$

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-multiplicate (70) 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:

$$BF^{-1}B^T \approx BQ_v^{-1}B^T F_p^{-1}Q_p. \tag{71}$$

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:

$$S = -BF^{-1}B^T \approx -A_p F_p^{-1}Q_p \tag{72}$$

The preconditioner (64) with the approximation given in (72) 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 [21]. 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 [**?**]. 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 [**?**]. 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 [21].

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 [7].
The idea is to approximate the matrix operator F_p in (71) such that the discrete commutator (70) 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:

$$\min \|[Q_v^{-1}FQ_v^{-1}B^T]_j - Q_v^{-1}B^TQ_p^{-1}[F_p]_j\|_{Q_v}, \tag{73}$$

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

$$Q_p^{-1}BQ_v^{-1}B^TQ_p^{-1}[F_p]_j = [Q_p^{-1}BQ_v^{-1}FQ_v^{-1}B^T]_j, \tag{74}$$

which leads to the following definition of F_p:

$$F_p = Q_p(BQ_v^{-1}B^T)^{-1}(BQ_v^{-1}FQ_v^{-1}B^T). \tag{75}$$

Substituting this expression into (71) provides an approximation to the Schur complement matrix:

$$S = BF^{-1}B^T \approx (BQ_v^{-1}B^T)(BQ_v^{-1}FQ_v^{-1}B^T)^{-1}(BQ_v^{-1}B^T). \tag{76}$$

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 (76). 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: LSC 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 [27].

4.3 Augmented Lagrangian approach

A completely different approach has been published by Benzi and Olshanskii [3]. 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 [21].
The system of equations (23) is replaced by

$$\begin{bmatrix} F_\gamma & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{77}$$

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 (77) without modifying the right hand side. This technique suggests a preconditioner of the form:

$$P_{AL} = \begin{bmatrix} F_\gamma & B \\ 0 & \hat{S} \end{bmatrix}, \tag{78}$$

with the inverse of the Schur complement approximated by

$$\hat{S}^{-1} = -(\nu \hat{Q}_p^{-1} + \gamma W^{-1}). \tag{79}$$

$\hat{Q}_p$ 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 [27]. The algorithms follows from a block LU decomposition of the coefficient matrix (23):

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & -BF^{-1}B^T \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{80}$$

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:

$$\begin{bmatrix} F & 0 \\ B & -BF^{-1}B^T \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{81}$$

and

$$\begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} u^* \\ \delta p \end{bmatrix}. \tag{82}$$

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

Algorithm: SIMPLE Preconditioner

1. p^* is given.
2. Solve $Fu^* = r_u - B^T p^*$.
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.* 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 [**?**].

4.4.1 SIMPLER

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

$$\hat{S}p^* = r_p - BD^{-1}((D - F)u^k + r_u), \tag{83}$$

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

$$\hat{S}p^* = r_p - BD^{-1}r_u. \tag{84}$$

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

Algorithm: SIMPLER 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 [21].

4.4.2 hSIMPLER

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 [27]. A better convergence can b e achieved if the first iteration is carried out with the SIMPLE preconditioner and after that SIMPLER is employed. The authors of [27] 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.

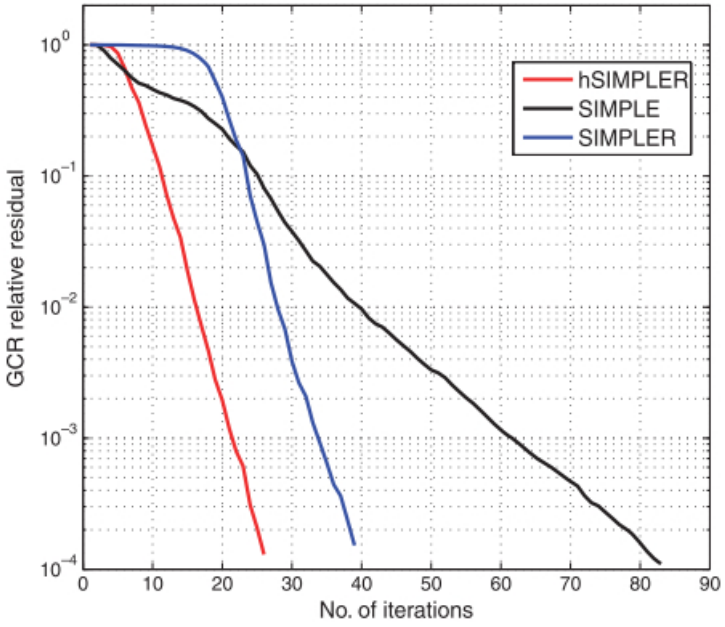

Figure 1: Convergence plot of SIMPLE-type peconditioners for the Stokes problem [27]

4.4.3 MSIMPLER

Elman *et al.* [7] discussed the relation between SIMPLE and approximate commutator preconditioners, which is presented next. The more general form of (76) is given by

$$(BF^{-1}B^T)^{-1} \approx F_p(BM_1^{-1}B^T)^{-1}, \tag{85}$$

where

$$F_p = (BM_2^{-1}B^T)^{-1}(BM_2^{-1}FM_1^{-1}B^T), \tag{86}$$

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:

$$P = \begin{bmatrix} F & 0 \\ B & -BM_1^{-1}B^T \end{bmatrix} \begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & F_p^{-1} \end{bmatrix}. \tag{87}$$

where $M_1 = D$ and F_p is the identity matrix, then the preconditioner formulation (87) corresponds to SIMPLE. The formulation given in (87) 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 (85),

$$\hat{S}\delta p = r_p - Bu^* \tag{88}$$

where

$$\hat{S} = -(BM_1^{-1}B^T)F_p^{-1}. \tag{89}$$

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

$$F_t = \frac{1}{\Delta t} Q_v + F, \tag{90}$$

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 (85) resulting in:

$$F_p = (B\hat{Q}_v^{-1}B^T)^{-1}(B\hat{Q}_v^{-1}F\hat{Q}_v^{-1}B^T). \tag{91}$$

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

$$F_p = (B\hat{Q}_v^{-1}B^T)^{-1}(B\hat{Q}_v^{-1}B^T) \approx I, \tag{92}$$

and the approximation (85) becomes

$$BF^{-1}B^T \approx -B\hat{Q}_v^{-1}B^T. \tag{93}$$

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: MSIMPLER 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.

5 Test problems

A set of benchmarks is presented here for solving the Navier-Stokes problem (1) and (2). The following test problems have been solved using the IFISS package. IFISS is a graphical MATLAB package for the interactive numerical study of incompressible flow problems. 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 state-of-the-art preconditioned iterative solvers for the discrete linear equation systems that arise in incompressible flow modeling. A unique feature of the package is its comprehensive nature; for each problem addressed, it enables the study of both discretization and iterative solution algorithms as well as the interaction between the two and the resulting effect on overall efficiency.

5.1 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:

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

Here a solution is computed numerically on Ω 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, pressure gradient is proportional to the viscosity parameter. The solution is given next.

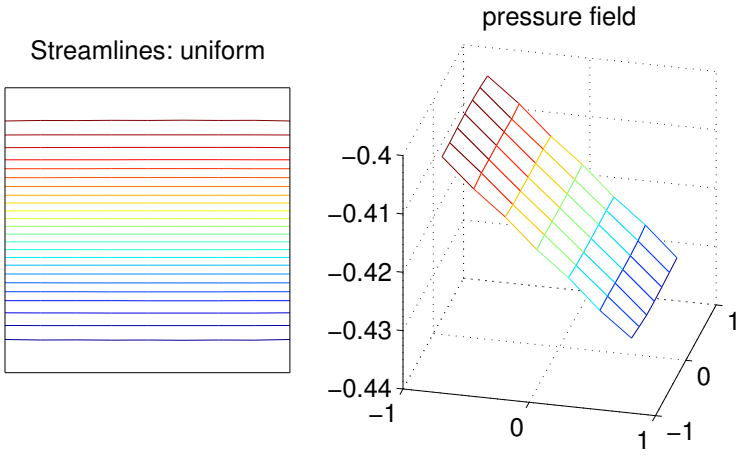

Figure 2: Solution to the Poiseuille flow problem.

5.2 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:

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

The solution of the driven cavity problem is presented next.

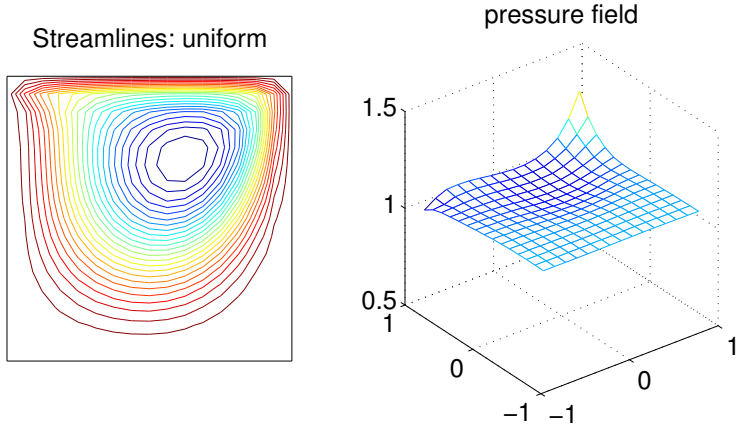

Figure 3: Solution to the Driven cavity problem with a watertight cavity.

The performance of a specific preconditioner applied to the Krylov subspace method can be studied via the built in functions of IFISS. The convergence plot of the BICGSTAB(2) method preconditioned with different preconditioners is presented next:

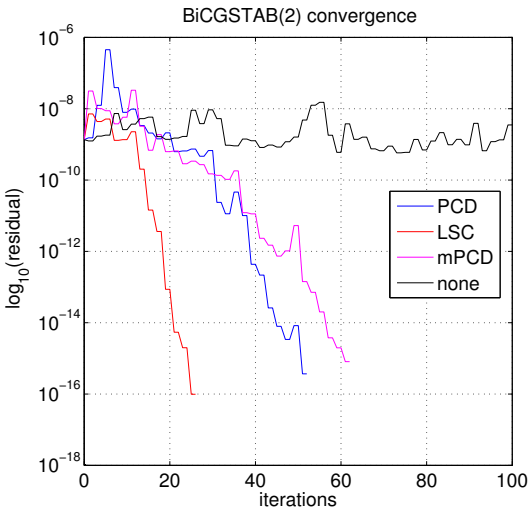

Figure 4: Convergence plot of BICGSTAB(2).

5.3 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:

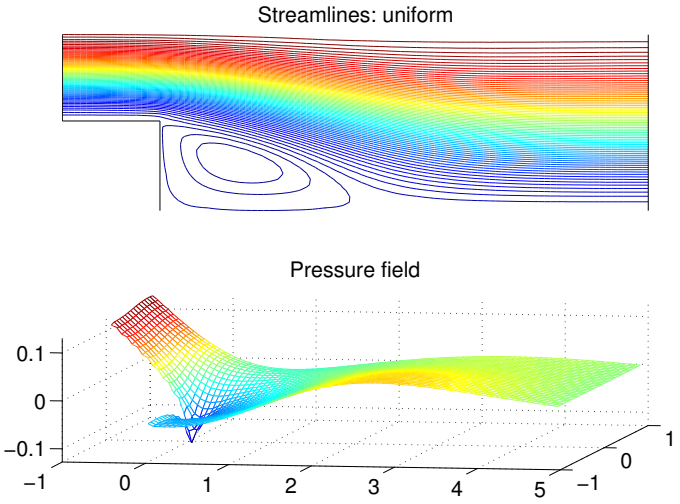

Figure 5: Solution to the backward facing step with stabilized $\mathbf{Q_1} - \mathbf{P_0}$ approximation.

The performance of iterative solution methods and preconditioners can be explored using the driver `it_solve`. Here, the chosen iterative method is `GMRES` with different methods of preconditioning. The convergence behavior of the GMRES method is presented next.

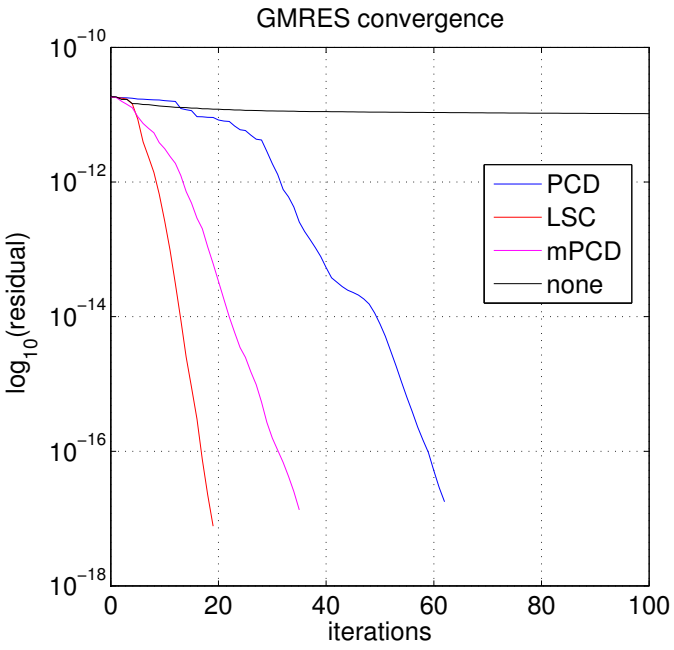

Figure 6: Convergence plot of GMRES.

6 Research questions

Iterative methods generate a sequence of approximate solutions $x^{(k)}$ and essentially involve the matrix A only in the context of matrix-vector multiplication. The evaluation of an iterative method invariably focuses on how quickly the iterates $x^{(k)}$ converge. The study of round-off errors is in general not very well developed. A reason for this is that the iterates are only approximations of the exact solution, so round-off errors in general only influence the speed of convergence but not the quality of the final approximation.

- Investigation of the stagnation behavior of the initial phase of SIMPLER.
- Why does the number of iterations get worse for stretched grids?

References

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

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

[3] M. Benzi and M. A. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing*, 28(6):2095–2113, 2006.

[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. 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.

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

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

[8] 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, 2007.

[9] H. C. Elman, A. Ramage, and D. J. Silvester. Incompressible Flow Iterative Solution Software (IFISS) installation & software guide version 3.2. *http://www.maths.manchester.ac.uk/˜djs/ifiss/document.html*, pages 1–18, 2012.

[10] 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):665–688, 2002.

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

[12] A. Gauthier, F. Saleri, and A. Veneziani. A fast preconditioner for the incompressible Navier Stokes equations. *Computing and Visualization in science*, 112:105–112, 2004.

[13] 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.

[14] C. M. Klaij and C. Vuik. SIMPLE-type preconditioners for cell-centered , colocated finite volume discretization of incompressible Reynolds-averaged NavierStokes equations. *International Journal for Numerical Methods in Fluids*, di, 2012.

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

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

[17] 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.

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

[19] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2 edition, 2003.

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

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

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

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

[24] 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.

[25] M. ur Rehman, C. Vuik, and G. Segal. A comparison of preconditioners for incompressible NavierStokes solvers. *International Journal for Numerical methods in fluids*, (57):1731–1751, 2008.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.