Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA CIVILE, EDILE E AMBIENTALE
Corso di Laurea Magistrale in Ingegneria Matematica

Block preconditioners for saddle point linear systems arising in the FE discretization of the Navier-Stokes equations. Application to the driven cavity problem.

Candidato:
Filippo Zanetti
Matricola 1179931

Relatore:
Prof. Luca Bergamaschi

Anno Accademico 2018-2019

Contents

List of Figures

List of Tables

Introduction

The task of solving numerically the Navier-Stokes equations is of fundamental importance in many scientific and industrial applications; the strong nonlinearity of the equations and the lack of any theoretical result about existence and regularity of the solutions leaves the scene only to numerical approximations. These have been developed since the dawn of the computing era, but despite the enormous efforts put into the development of new algorithms, the problem of finding a good solver for most of the practical situations involving the Navier-Stokes equations has always been elusive, particularly when the Reynolds number becomes large. One of the approaches to the numerical solution of the Navier-Stokes equations is given by the Finite Element Method, which, after a linearization of the nonlinear terms, gives rise to a saddle point linear system: this particular system has a structure that appears in many other problems, but in this context it is possible to exploit the underlying continuous formulation to develop efficient solvers.

In this work, we focus in finding a scalable preconditioner for these saddle point linear systems: many preconditioners have already been developed and tested successfully, for instance in [2, 8, 9, 11, 20]; we choose to use a constraint preconditioner, already analyzed in all its forms in [3, 4, 12]. This preconditioner embeds in its structure many information about the matrix of the system, so we hope that it will provide better results than its competitors. We also employ a relaxation technique, presented in [5], in order to accelerate its convergence.

In order to obtain an efficient solver, we use the GMRES method and we look for a scalable preconditioner, i.e. a preconditioner that allows GMRES to converge in a number of iterations that does not deteriorate as the mesh is refined. The success of this task lays on finding scalable preconditioners for the $(1,1)$ block and the Schur complement of the saddle point system. It is well known that a Multigrid technique can be used as a scalable preconditioner for the Poisson problem; the $(1,1)$ block of our system corresponds to a discrete convection-diffusion operator, which is a variation of a Poisson problem that involves also convective processes. The generalization of Multigrid preconditioners to this kind of situations requires a robust smoother, that can be built using a stationary

method involving a pattern that follows the convective flow. This approach has already been tested in [8, 14, 17].

With regard to the preconditioner for the Schur complement, the possible approaches are different: some techniques are developed starting from the preconditioner built for the Stokes problem, while some others follow a more algebraic approach. We use mainly the preconditioner developed in [7] and improved in [15]. These techniques are developed assuming that a particular commutator is sufficiently small to neglect it, while they do not exploit the particular underlying structure of the problem, as it happens in the Stokes case.

The problem that we use as test is the famous 2-dimensional lid-driven cavity, discretized using $P_2 - P_1$ elements and with values of the viscosity as low as 10^{-3}. The main goals that we want to achieve are:

- Develop a smoother that is able to follow the convective flow in the case of the recirculating problem considered; this in turn would allow us to build a scalable Multigrid preconditioner for the $(1, 1)$ block.
- Understand the differences between the Schur complement preconditioners in the Stokes and Navier-Stokes problems and find a suitable scalable preconditioner for our test case.

The thesis is structured as follows:

- In the first chapter, we present the finite element method and we derive the formulation that will lead to the saddle point linear system.
- In the second chapter, we analyze the algebraic properties of saddle point matrices, underlying the connections with the continuous formulation; we also set the ground for the introduction of some preconditioning techniques.
- In the third chapter, we study the iterative methods used to solve large linear systems and in particular the GMRES method; we also introduce Multigrid methods and show why they are such interesting preconditioners.
- In the fourth chapter, we present the preconditioner that is used; we derive eigenvalue bounds and talk about the possible implementation strategies. We then focus in finding suitable preconditioners for the $(1, 1)$ block and the Schur complement.
- In the last chapter, we report the numerical results for all the combinations of preconditioners that we tried; we show both spectral information and convergence results, together with some pictures that clarify the differences between the various schemes.

Chapter 1

Navier-Stokes equations

In this Chapter, we will introduce the Navier-Stokes equations, discuss how to solve them numerically, underline all the challenges related to this task and finally present the model problem that will be analyzed in the next Chapters.

1.1 Mathematical formulation

The motion of incompressible newtonian fluids is governed by the well known Navier-Stokes equations, a system of partial differential equations that arises from the conservation of mass and momentum. In the general non-stationary case they take the form

$$\begin{cases} \rho \dfrac{\partial \mathbf{u}}{\partial t} - \mu \Delta \mathbf{u} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \mathbf{f}, & \mathbf{x} \in \Omega, t > 0 \\ \operatorname{div} \mathbf{u} = 0, & \mathbf{x} \in \Omega, t > 0 \end{cases}$$

where $\Omega \subset \mathbb{R}^3$ is the domain on which the motion evolves; $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the velocity field; $p = p(\mathbf{x}, t)$ is the pressure field; ρ and μ are the fluid density and dynamic viscosity; $\mathbf{f}$ is a forcing term.

The first equation is often used in a different form: dividing by the density, we obtain the following version of the Navier-Stokes equations

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \mathbf{f}, & \mathbf{x} \in \Omega, t > 0 \\ \operatorname{div} \mathbf{u} = 0, & \mathbf{x} \in \Omega, t > 0 \end{cases} \tag{1.1}$$

where now $\nu = \mu/\rho$ is the kinematic viscosity, p is the density-scaled pressure field and $\mathbf{f}$ is a forcing term per unit mass.

The first of the two equations imposes the conservation of momentum; the term $\nu \Delta \mathbf{u}$ takes into account the diffusive processes, while $(\mathbf{u} \cdot \nabla)\mathbf{u}$ describes

the convective processes. The equation $\operatorname{div} \mathbf{u} = 0$ imposes the incompressibility of the fluid, i.e. the density ρ is a constant, both in space and time.

For the problem to be well posed, equations (1.1) need some initial condition

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega$$

and boundary conditions, e.g. $\forall t > 0$

$$\begin{cases} \mathbf{u}(\mathbf{x}, t) = \phi(\mathbf{x}, t), & \forall \mathbf{x} \in \Gamma_D \\ \left(\nu \dfrac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n}\right)(\mathbf{x}, t) = \psi(\mathbf{x}, t), & \forall \mathbf{x} \in \Gamma_N \end{cases}$$

where $\mathbf{u}_0$, ϕ and ψ are given functions, Γ_D and Γ_N form a partition of the boundary of Ω and $\mathbf{n}$ is the outward-facing unit normal to $\partial\Omega$.

The first kind of boundary condition is said of Dirichlet type, and Γ_D will be addressed as the Dirichlet portion of the boundary, while the second kind is said of Neumann type, and Γ_N will be called the Neumann portion of the boundary.

The Navier-Stokes equations (1.1) are nonlinear, due to the term $(\mathbf{u} \cdot \nabla)\mathbf{u}$; moreover, there are no general results about existence, regularity and uniqueness of the solution, in particular in three dimensions. In fact, this is one of the most important open problems in mathematics and one of the Millennium problems.

In the following, we will always consider the stationary Navier-Stokes equations, i.e. equations (1.1) without the time derivative of $\mathbf{u}$.

1.1.1 Stokes equations

Define the Reynolds number as the ratio between inertial and viscous forces, namely $Re = \frac{UL}{\nu}$, where L is a characteristic length of the domain Ω and U is a representative velocity scale of the fluid. It turns out that, if the Reynolds number is sufficiently small (i.e. if the flow is particularly slow or if the viscosity is high enough), the Navier-Stokes equations can be simplified: in fact, the term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ is negligible with respect to the viscous term and the nonlinear equations (1.1) become a linear system of equations, known as the stationary Stokes equations:

$$\begin{cases} -\nu \Delta \mathbf{u} + \nabla p = \mathbf{f}, & \mathbf{x} \in \Omega, t > 0 \\ \operatorname{div} \mathbf{u} = 0, & \mathbf{x} \in \Omega, t > 0 \end{cases} \tag{1.2}$$

to which the same previously cited initial and boundary conditions must be applied. In these equations there is no more the presence of convection, but only diffusive processes survive.

The Stokes equations (1.2) have the great advantage of being linear, which

makes them easier to work with, both analytically and numerically.

1.1.2 Weak formulation

The general formulation of equations (1.1) and (1.2), which require as solution a function $\mathbf{u}$ twice differentiable and a function p continuously differentiable, can be relaxed, allowing for solutions that satisfy weaker requirements. Sometimes in fact, there does not exist any solution that satisfies the strong form of the equations, while it is possible to find weak solutions.

Before deriving the weak form of the Navier-Stokes equations, let us define the space of square-integrable functions

$$L^2(\Omega) = \left\{ f \colon \Omega \mapsto \mathbb{R} \mid \int_\Omega |f(\mathbf{x})|^2 \, d\Omega < +\infty \right\}$$

and the Sobolev Space

$$H^k(\Omega) = \left\{ f \in L^2(\Omega) \mid D^\alpha f \in L^2(\Omega) \quad \forall \alpha \colon |\alpha| \leq k \right\},$$

i.e. the space of square integrable functions whose derivatives up to order k are still square integrable. Here, $D^\alpha f$ represents a weak derivative.

Now, starting from the stationary Navier-Stokes equations, it is possible to obtain the weak formulation multiplying by a test function $\mathbf{v} \in V$, where the space V will be defined later, and integrating over Ω.

$$-\int_\Omega \nu \Delta \mathbf{u} \cdot \mathbf{v} \, d\Omega + \int_\Omega [(\mathbf{u} \cdot \nabla)\mathbf{u}] \cdot \mathbf{v} \, d\Omega + \int_\Omega \nabla p \cdot \mathbf{v} \, d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} \, d\Omega \quad \forall \mathbf{v} \in V.$$

In this form there is still the necessity for the function $\mathbf{u}$ to be twice differentiable, which is the condition that we want to relax. In order to do so, we exploit Green's formulas (i.e. integration by parts in multiple dimensions) and rewrite the integral involving $\Delta \mathbf{u}$ as the sum of an integral involving only $\nabla \mathbf{u}$ and a boundary integral. The same can be done for the pressure term. The result is

$$\int_\Omega \nu \nabla \mathbf{u} \cdot \nabla \mathbf{v} \, d\Omega + \int_\Omega [(\mathbf{u} \cdot \nabla)\mathbf{u}] \cdot \mathbf{v} \, d\Omega - \int_\Omega p \, \mathrm{div} \mathbf{v} \, d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} \, d\Omega + \int_{\partial\Omega} \left(\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n} \right) \cdot \mathbf{v} \, dS \quad \forall \mathbf{v} \in V. \quad (1.3)$$

The same can be done for the second equation, considering a test function $q \in Q$. The result is

$$\int_\Omega q \, \mathrm{div} \mathbf{u} \, d\Omega = 0 \quad \forall q \in Q. \quad (1.4)$$

So, an alternative version of the Navier-Stokes equations is: find $\mathbf{u} \in V$ and

$p \in Q$ such that (1.3) and (1.4), together with the proper initial and boundary conditions, hold for every possible choice of $\mathbf{v} \in V$ and $q \in Q$. The couple $(\mathbf{u}, p)$ is then called a *weak solution* of the Navier-Stokes equations. For this formulation to make sense, all the integrals involved must be well defined: this is achieved by choosing properly the spaces V and Q. The correct choice is to set V equal to the space of functions in $H^1(\Omega)$ such that they satisfy the Dirichlet boundary condition on the Dirichlet portion of the boundary; Q should simply be the space $L^2(\Omega)$. It can be checked that in this way all the integrals are well defined (see [16, pp. 431-435]).

With this formulation, the strong requirements about differentiability that were stated at the beginning are lost; in fact, the solution does not even need to be continuous. The same thing can be done for the Stokes equations, yielding the same result, only without the integral containing the convective term. This formulation will be the basis from which the finite element method is constructed.

1.2 Finite element method

The previously stated weak formulation can be expressed in a more compact form: suppose to solve a Stokes problem with homogeneous Dirichlet boundary condition on all the boundary. Then, the weak formulation is equivalent to:

find $\mathbf{u} \in V$, $p \in Q$ such that

$$\begin{cases} a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) = (\mathbf{f}, \mathbf{v}) & \forall \mathbf{v} \in V \\ b(\mathbf{u}, q) = 0 & \forall q \in Q \end{cases} \tag{1.5}$$

where $a\colon V \times V \mapsto \mathbb{R}$ and $b\colon V \times Q \mapsto \mathbb{R}$ are bilinear forms defined as

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega \nu \nabla \mathbf{u} \cdot \nabla \mathbf{v} \, d\Omega,$$

$$b(\mathbf{u}, q) = -\int_\Omega q \mathrm{div} \mathbf{u} \, d\Omega$$

and

$$(\mathbf{f}, \mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} \, d\Omega.$$

If instead the Navier-Stokes problem is solved, there is one more term in the left-hand side of the first equation of (1.5): the convective term is represented with a trilinear form $c\colon V \times V \times V \mapsto \mathbb{R}$:

$$c(\mathbf{u}, \mathbf{u}, \mathbf{v}) = \int_\Omega [(\mathbf{u} \cdot \nabla)\mathbf{u}] \cdot \mathbf{v} \, d\Omega.$$

In the following of this Section, we will consider the formulation for the Stokes problem, since the theoretical results are easier to understand. Analogous results hold also for the Navier-Stokes problem.

1.2.1 Galerkin approximation

Equations (1.5) are defined on the spaces V and Q, which are subspaces of H^1 and L^2 of infinite dimension. To develop a numerical scheme able to approximate the solution, we need to find an approximate problem defined on spaces of finite dimension. Consider, $V_h \subset V$ and $Q_h \subset Q$, both of finite dimension, then the problem

find $\mathbf{u}_h \in V_h$, $p_h \in Q_h$ such that

$$\begin{cases} a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = (\mathbf{f}, \mathbf{v}_h) & \forall \mathbf{v}_h \in V_h \\ b(\mathbf{u}_h, q_h) = 0 & \forall q_h \in Q_h \end{cases} \tag{1.6}$$

is an approximation of the original problem, and it is called Galerkin approximation.

The ability to solve problem (1.6) then relies on finding a good basis for the spaces V_h and Q_h. Indeed, denote by $\varphi_i,\ i = 1, \dots, N_V$ a basis of V_h, where N_V is its dimension; denote also by $\psi_i,\ i = 1, \dots, N_Q$ a basis for Q_h. To check that the first equation of (1.6) holds $\forall \mathbf{v}_h \in V_h$ it is sufficient to check that it holds $\forall \varphi_i,\ i = 1, \dots, N_V$. Then, since $u_h \in V_h$ and $p_h \in Q_h$, it follows that they can be written as

$$u_h(\mathbf{x}) = \sum_{j=1}^{N_V} u_j \varphi_j(\mathbf{x}), \qquad p_h(\mathbf{x}) = \sum_{j=1}^{N_Q} p_j \psi_j(\mathbf{x})$$

where u_j and p_j are unknown coefficients.

Therefore, the first equation of (1.6), due to the linearity of $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$, becomes

$$\sum_{j=1}^{N_V} u_j a(\varphi_j, \varphi_i) + \sum_{j=1}^{N_Q} p_j b(\varphi_i, \psi_j) = (\mathbf{f}, \varphi_i) \quad \forall i = 1, \dots, N_V. \tag{1.7}$$

The second equation instead becomes

$$\sum_{j=1}^{N_V} u_j b(\varphi_j, \psi_i) = 0 \quad \forall i = 1, \dots, N_Q. \tag{1.8}$$

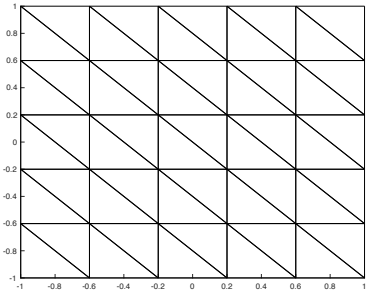
Once the basis functions are fixed, the previous equations can be expressed as a linear system of algebraic equations, involving the matrices $A_{ij} = a(\varphi_j, \varphi_i)$ and

$B_{ij} = b(\varphi_j, \psi_i)$, and with unknowns u_j and p_j. Therefore, using the Galerkin approximation, it is possible to transform a linear system of partial differential equations into a linear system of algebraic equations.

The properties of the linear system arising from this approximation depend greatly on the choice of the basis functions. The *Finite Element Method* (FEM) chooses to use very large dimensions for the spaces V_h and Q_h, and therefore very large matrices A and B arise. In order to be able to work with these big matrices, they must be sparse (a matrix is said to be sparse if only few of its entries are different from zeros; therefore, the number of its nonzero elements depends linearly on the dimension of the matrix and not quadratically, as it happens for the more common dense matrices).

The question now is how to choose φ_j and ψ_j so that the matrices A and B are sparse. We start by subdividing the domain Ω in a large number of small polygonal regions, called *elements*; Figure 1.1 shows an example of subdivision of the square $[-1, 1] \times [-1, 1]$ into triangular elements.

Figure 1.1: Example of subdivision of the domain.



Let us consider the vertexes of this polygonal regions, we will call them *nodes*. The set of all nodes forms a discrete set of points over which the solution will be approximated. Therefore, we will not be able to know the solution everywhere we want, but only over the nodes that we decided at the beginning. The unknowns u_j and p_j previously defined will thus be the values of the approximated solution at every node. To do so, we need to build basis functions for every node, for both the spaces V_h and Q_h.

Recall that

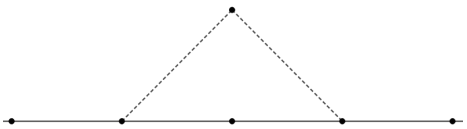
$$A_{ij} = a(\varphi_j, \varphi_i) = \int_\Omega \nu \nabla \varphi_j \cdot \nabla \varphi_i \, d\Omega = \sum_{\Omega_e} \int_{\Omega_e} \nu \nabla \varphi_j \cdot \nabla \varphi_i \, d\Omega,$$

where Ω_e represents a single element. This means that the global integral over Ω can be seen as the sum of the integrals over all the elements. In order for A to be sparse, these integrals need to be almost always zero. A way to do it, is to choose φ_j so that is has compact support: in particular, φ_j will be different

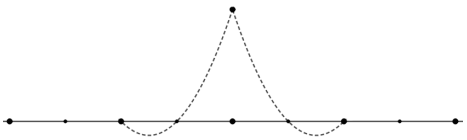
from zero only over the elements to which the node j belongs. In this way, the entry i, j of the matrix will be nonzero only if nodes i and j are adjacent on the domain. For the triangular discretization of Figure 1.1, every node has at most 7 adjacent nodes, counting also the node itself; thus, every row of matrix A will have at most 7 entries different from zero, regardless of the dimension of the matrix. The same reasoning holds for matrix B.

So, the basis functions have to be continuous, with compact support and it should be easy to evaluate the integrals involved in the weak formulation. A good idea is to consider piecewise polynomial functions. The simplest choice that allows continuity is piecewise linear functions (piecewise constants do not permit continuity, unless the basis function is identically zero). The functions are chosen in such a way that they attain the value 1 on the node to which they correspond, while they attain the value 0 on any other node. Figure 1.2 illustrates this kind of basis functions in one dimension for two different choices: linear and quadratic polynomials. It can be seen that to define quadratic functions, we need to consider as nodes also the midpoints of the edges of the discretization.

Figure 1.2: Piecewise polynomial basis functions in one dimension.



(a) *Linear.*



(b) *Quadratic.*

In multiple dimensions, the elements can have various shapes and the basis functions can have various degrees, so there are many possibilities. The most used choices are triangular or quadrilateral elements, with polynomials of degree 1 or 2. Some care must be put in the determination of the nodes, depending on the shape of the element and on the degree of the polynomials, in order to guarantee continuity and uniqueness of the basis functions.

For triangular elements, the choice of using linear basis functions is denoted as P_1, for quadratic functions as P_2, and in general using polynomials of degree k is denoted as P_k. As it is intuitive, the higher the degree of the basis functions, the better the approximation is; however, also the computational cost and the dimension of the linear system to solve grow. Accuracy is also strongly related

to the number of elements used to discretize the domain: a finer partition gives more accurate results, but also implies a greater computational cost.

1.2.2 Some grid properties

Let us now state some properties that a generic grid, used to solve the Navier-Stokes equations with the finite element method, may have. These definitions will be useful in the later Chapters.

Definition 1.1 (Quasi-uniform subdivision)**.** A sequence of triangular grids $\{\tau_h\}$ is said to be *quasi-uniform* if there exists a constant $\rho > 0$ such that $\underline{h} \geq \rho \overline{h}$ for every grid in the sequence, where $\underline{h}$ is the minimum edge among all the triangles in the triangulation considered and $\overline{h}$ is the maximum one.

Definition 1.2 (Shape regular elements)**.** A sequence of triangular grids $\{\tau_h\}$ is said to be *shape regular* if there exists a minimum angle $\theta \neq 0$ such that every element in τ_h has its minimum angle greater or equal than θ.

Definition 1.3 (H^2 regularity)**.** A problem like (1.5) is said to be H^2 *regular* if, for every $\mathbf{f} \in L^2$, the solution $u \in V$ is also in the space H^2 and its H^2-norm is controlled by the L^2-norm of $\mathbf{f}$.

In all the problems that we will consider, we will always use quasi-uniform subdivisions and shape-regular elements; moreover, we will always assume that the problem is H^2 regular.

1.2.3 The Oseen problem

The Galerkin approximation can be formulated also for the Navier-Stokes problem and it takes the form

$$\begin{cases} a(\mathbf{u}_h, \mathbf{v}_h) + c(\mathbf{u}_h, \mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = (\mathbf{f}, \mathbf{v}_h) & \forall \mathbf{v}_h \in V_h \\ b(\mathbf{u}_h, q_h) = 0 & \forall q_h \in Q_h \end{cases}. \tag{1.9}$$

The problem with this formulation is that, following the same steps as before, the algebraic system that arises is no more linear, but, due to the trilinear form $c(\cdot, \cdot, \cdot)$, it becomes nonlinear. This complicates enormously the method, since solving a nonlinear system of equations involves methods far more complicated and computationally costly (e.g. Newton method). The simplest way to solve this problem is to use a fixed-point scheme (known also as Picard iteration) and try to linearize the nonlinear term $c(\mathbf{u}_h, \mathbf{u}_h, \mathbf{v}_h)$. This will require an iterative scheme: suppose to start from a given velocity field $\mathbf{u}_h^0$; to find the next iterate $\mathbf{u}_h^1$ one could solve equations (1.9) where, instead of the nonlinear term, there

is the linear term $c(\mathbf{u}_h^0, \mathbf{u}_h^1, \mathbf{v}_h)$. In this way, the trilinear form that was giving problems becomes simply a bilinear form and can be treated like the other ones. So, the method can be formalized as follows

$$\begin{cases} a(\mathbf{u}_h^k, \mathbf{v}_h) + c(\mathbf{u}_h^{k-1}, \mathbf{u}_h^k, \mathbf{v}_h) + b(\mathbf{v}_h, p_h^k) = (\mathbf{f}, \mathbf{v}_h) & \forall \mathbf{v}_h \in V_h \\ b(\mathbf{u}_h^k, q_h) = 0 & \forall q_h \in Q_h \end{cases} \tag{1.10}$$

for $k = 1, 2, \ldots$ until convergence. The initial field $\mathbf{u}_h^0$ must be taken divergence free, in order to be consistent with the problem. This can be achieved solving a Stokes problem at the beginning (which assures a divergence-free velocity field) and then using the solution as initial datum for the iteration. This is consistent with setting $\mathbf{u}_h^{-1} = \mathbf{0}$ and starting the iterations from $k = 0$. This formulation of the Navier-Stokes problem is known as *Oseen problem* and it is the method that will be used in this work.

1.2.4 Stabilization of the convection-diffusion term

The first terms in the Oseen problem (1.10) represent a convection-diffusion operator of the form $-\nu\Delta\mathbf{u}_h + \mathbf{w}\cdot\nabla\mathbf{u}_h$, where the wind $\mathbf{w}$ is given by the solution of the previous iteration. In the numerical solution of this kind of problems, one important quantity to estimate the stability of the method is the Péclet number, defined as the ratio between convective and diffusive forces, i.e. $Pe = \frac{Lw}{\nu}$, where L is a characteristic length and w is the local wind velocity. When discretizing a convection-diffusion problem, this number is used considering as characteristic length the dimension of the elements. In this way, this parameter is able to tell whether the solution will be stable or not: small values of Pe assure a good solution, while a large Pe leads to instability.

This can be solved using a finer discretization, but when the viscosity is very low, the elements would need to be so small that the computational cost would become enormous. Another option is to use a stabilization technique: instead of solving the unstable problem, one can solve a slight modification of the problem that is stable. This is achieved by adding some artificial diffusion to the problem, which assures that the Péclet number decreases. Of course, the solution will be slightly different from the real one, but at least it will be stable. If the stabilization is done correctly, then the solution will not be affected too much.

There exist various methods of stabilization; in this work the simplest one will be used, called streamline diffusion (SD), which adds diffusion only in the direction of the wind. Other methods include SUPG, ASGS, GLS but their theory and implementation are far more complicated. The SD method

introduces another bilinear form in the formulation of the Navier-Stokes equations, $s(\cdot,\cdot)\colon\ V_h \times V_h \mapsto \mathbb{R}$ defined as

$$s(\mathbf{u}_h, \mathbf{v}_h) = \int_\Omega \tau_{SD}(\mathbf{w}\cdot\mathbf{u}_h)(\mathbf{w}\cdot\mathbf{v}_h)\, d\Omega,$$

where τ_{SD} is defined as follows: call $Pe_h = \frac{hw}{2\nu}$ the mesh Péclet number, with h the local dimension of the element. Then, if $Pe_h < 1$, τ_{SD} is set to zero, i.e. no stabilization is performed on those elements where Pe_h is sufficiently small. Instead, where $Pe_h \geq 1$

$$\tau_{SD} = \frac{h}{2w}\Big(1 - \frac{1}{Pe_h}\Big).$$

This choice of the stabilization parameter is taken from [8, p. 253].

This stabilization technique will be useful when solving the Navier-Stokes equation with low values of viscosity.

1.2.5 Model problem

The model problem used in this work is the famous lid-driven cavity problem. The source term $\mathbf{f}$ is set to zero, the viscosity ν is set to 1 for the Stokes equations, while it varies between 0.1 and 0.001 for the Navier-Stokes equations. The domain Ω corresponds to the 2-dimensional square $[-1,1]\times[-1,1]$. The equations, together with the boundary conditions, are

$$\begin{cases} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{0} \\ \operatorname{div}\mathbf{u} = 0 \\ \mathbf{u} = 0, \quad \mathbf{x}\in\{-1,1\}\times[-1,1]\cup[-1,1]\times\{-1\} \\ u_y = 0, \quad \mathbf{x}\in[-1,1]\times\{1\} \\ u_x = 1, \quad \mathbf{x}\in[-1,1]\times\{1\} \end{cases} . \tag{1.11}$$

The problem evolves inside a square domain; on the two lateral sides and on the bottom side there is a no-slip condition (i.e. the velocity is zero), while on the top side there is an horizontal velocity imposed. There is no normal velocity on any portion of the boundary, thus the flow is enclosed (no fluid can enter or exit the domain).

Figures 1.3 and 1.4 show the velocity magnitude and direction for the Stokes problem; it is clear the formation of a vortex in the middle. This feature makes this model problem especially challenging to treat and particularly interesting to evaluate the robustness of a numerical solver.

Figure 1.3: Velocity magnitude for the Stokes problem.

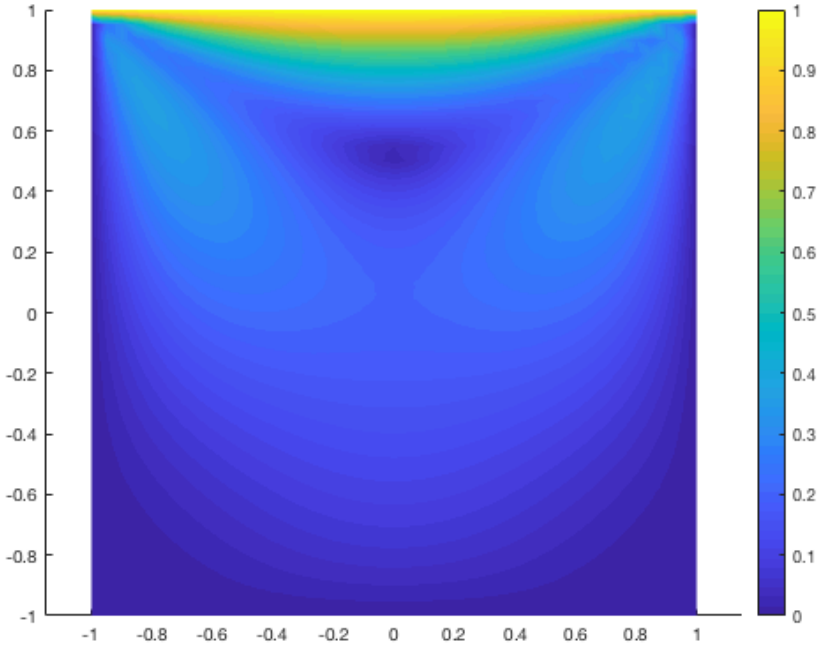
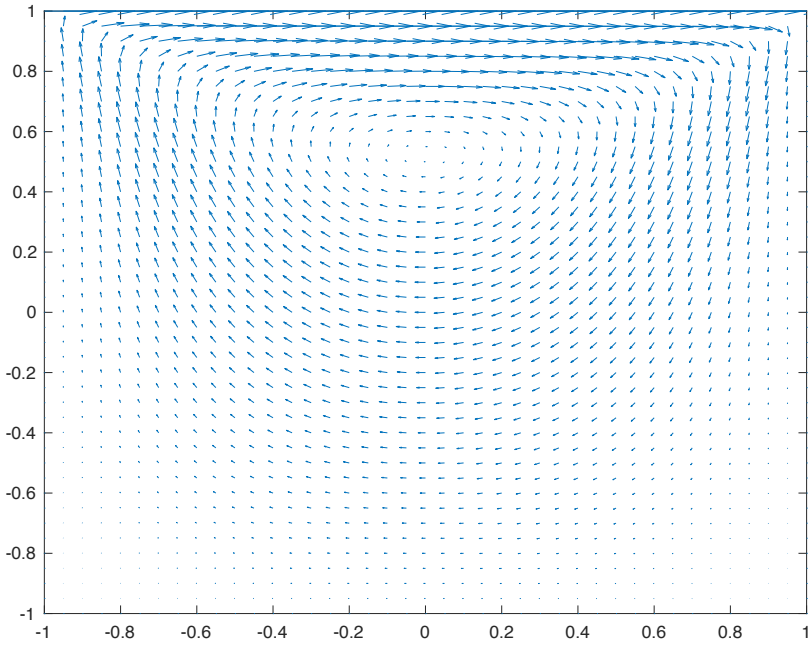


Figure 1.4: Velocity direction for the Stokes problem.



1.3 Stability and convergence

A bilinear form $a : U \times V \mapsto \mathbb{R}$ is said to be *continuous* if

$$\exists\, C > 0 \colon |a(\mathbf{u}, \mathbf{v})| \leq C \|\mathbf{u}\|_U \|\mathbf{v}\|_V \quad \forall \mathbf{u} \in U,\, \mathbf{v} \in V$$

and *coercive* if

$$\exists\, C > 0 \colon a(\mathbf{v}, \mathbf{v}) \geq C \|\mathbf{v}\|_V^2 \quad \forall \mathbf{v} \in V.$$

With these definitions, it is now possible to state the following

Theorem 1.1. *The Galerkin approximation* (1.6) *admits one and only one solution if the following conditions hold:*

1. $a(\cdot, \cdot)$ *is continuous on* $V_h \times V_h$*, with constant* γ
2. $a(\cdot, \cdot)$ *is coercive on* $V_h^0 = \{\mathbf{v}_h \in V_h \colon b(\mathbf{v}_h, q_h) = 0,\ \forall q_h \in Q_h\}$*, with constant* α
3. $b(\cdot, \cdot)$ *is continuous on* $V_h \times Q_h$*, with constant* δ
4. *there exists* $\beta > 0$ *such that*

$$\inf_{\substack{q_h \in Q_h \\ q_h \neq 0}} \sup_{\substack{\mathbf{v_h} \in V_h \\ \mathbf{v_h} \neq 0}} \frac{b(\mathbf{v_h}, q_h)}{\|\mathbf{v_h}\|_{H^1} \|q_h\|_{L^2}} \geq \beta. \tag{1.12}$$

Moreover, the following convergence results hold

$$\|\mathbf{u} - \mathbf{u_h}\|_V \leq \Big(1 + \frac{\delta}{\beta}\Big)\Big(1 + \frac{\gamma}{\alpha}\Big) \inf_{\mathbf{v_h} \in V_h} \|\mathbf{u} - \mathbf{v_h}\|_V + \frac{\delta}{\alpha} \inf_{q_h \in Q_h} \|p - q_h\|_Q, \tag{1.13}$$

$$\begin{aligned} \|p - p_h\|_Q \leq & \frac{\gamma}{\beta}\Big(1 + \frac{\gamma}{\alpha}\Big)\Big(1 + \frac{\delta}{\beta}\Big) \inf_{\mathbf{v_h} \in V_h} \|\mathbf{u} - \mathbf{v_h}\|_V + \\ & + \Big(1 + \frac{\delta}{\beta} + \frac{\delta\gamma}{\alpha\beta}\Big) \inf_{q_h \in Q_h} \|p - q_h\|_Q \end{aligned} \tag{1.14}$$

where $\mathbf{u}$ *and* p *represent the exact solutions.*

The proof of this theorem is very technical and requires some advanced functional analysis. A sketch of the proof can be found in [16, pp. 441-446]

Theorem 1.1 states the conditions under which the Galerkin approximation is a well-posed problem; conditions 1, 2 and 3 are in general verified for many choices of spaces V_h and Q_h. The main problem is to be able to satisfy inequality (1.12), which is usually called *inf-sup condition* or LBB condition (Ladyzhenskaya-Babuska-Brezzi).

The convergence results (1.13) and (1.14) allow to bound the overall error on the velocity ($\|\mathbf{u} - \mathbf{u_h}\|_V$) and pressure fields ($\|p - p_h\|_Q$), with the best

possible approximation of the solution that can be found on the spaces V_h ($\inf_{\mathbf{v_h}\in V_h} \|\mathbf{u}-\mathbf{v_h}\|_V$) and Q_h ($\inf_{q_h\in Q_h} \|p-q_h\|_Q$). The constants on which these bounds depend are greatly influenced by the inf-sup constant β: if it is very small, the bound on the solution will be very loose. If such a constant does not exist, i.e. (1.12) is satisfied only for $\beta = 0$, then the control on the stability of the solution is completely lost.

Therefore, it is critical to choose correctly the spaces V_h and Q_h, since not all the possible couples of spaces will provide good solutions. To understand how condition (1.12) affects the choice of these spaces, let us consider again problem (1.5), but this time we will use some operators and some basic functional analysis. First of all, suppose that V and Q are Hilbert spaces; then, define their dual spaces V' and Q', the spaces of all linear and bounded functionals defined on V and Q. Define an operator A that maps V into V' and that acts as follows

$$(A\mathbf{w}, \mathbf{v}) = a(\mathbf{w}, \mathbf{v}) \qquad \forall \mathbf{w}, \mathbf{v} \in V$$

i.e. A takes a vector $\mathbf{w} \in V$ and produces a functional $(A\mathbf{w}, \cdot) \in V'$. In the same way, we define an operator B that maps V into Q' and that acts as

$$(B\mathbf{v}, q) = b(\mathbf{v}, q) \qquad \forall \mathbf{v} \in V,\, q \in Q.$$

Its adjoint operator, B^T, maps Q into V' and acts as

$$(B^T q, \mathbf{v}) = (B\mathbf{v}, q) = b(\mathbf{v}, q) \qquad \forall \mathbf{v} \in V,\, q \in Q.$$

Given these definitions, we can write problem (1.5) as

$$\begin{cases} (A\mathbf{u}, \mathbf{v}) + (B^T p, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) & \forall \mathbf{v} \in V \\ (B\mathbf{u}, q) = 0 & \forall q \in Q \end{cases}$$

and therefore also as

$$\begin{cases} A\mathbf{u} + B^T p = \mathbf{f} & \text{in } V' \\ B\mathbf{u} = 0 & \text{in } Q' \end{cases}.$$

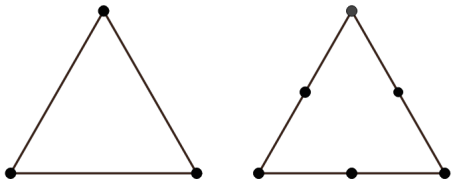
Define now the kernel of operator B: $V^0 = \ker(B) = \{\mathbf{v} \in V : b(\mathbf{v}, q) = 0,\, \forall q \in Q\}$. Therefore, a reduced version of the original problem is

$$\text{find} \quad \mathbf{u} \in V^0 \quad \text{such that} \quad a(\mathbf{u}, \mathbf{v}) = \mathbf{f}\ \forall \mathbf{v} \in V^0.$$

It is clear that a solution of the original problem also solves the reduced problem, but it turns out that also the converse holds, given some suitable conditions. In particular, we would like B to be an isomorphism between V and Q', but in

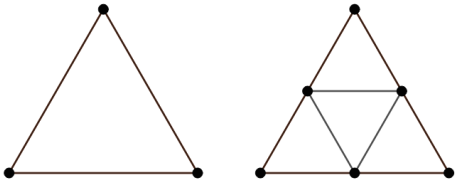
general its kernel V^0 will be non empty, so we can only obtain an isomorphism between $(V^0)^\perp$ and Q', where $(V^0)^\perp$ is the space orthogonal to V^0. As it turns out, if there exists an inf-sup condition that links the spaces V and Q, similarly to the discrete version (1.12), then B is indeed an isomorphism and the reduced problem is equivalent to the original one. Again, the details have not been discussed properly and the necessity of the inf-sup condition has not been proved, since this goes beyond the scope of this work; however, this discussion is useful to understand that the space V must have larger dimension than the space Q, since $(V^0)^\perp$, which is a subspace of V, must be isomorphic to Q.

Figure 1.5: Linear and quadratic triangular elements, used in the Taylor-Hood approximation.



This reasoning sets the ground for the difficult task of choosing the correct spaces V_h and Q_h, which should inherit the properties of their continuous counterparts V and Q. Since V_h must have larger dimension, it is surely not acceptable to use P_1 elements for both spaces, which would have been the easiest possibility. Luckily, all the couples of spaces of the kind $P_k - P_{k+1}$, with $k \geq 1$, do satisfy the inf-sup condition and can then be used in the finite elements setting. The simplest of these couples is the one which uses P_1 functions for the space Q_h and P_2 functions for V_h. This type of choice is sometimes referred to as *Taylor-Hood approximation* and it will be the one used in this work. Figure 1.5 shows the two kind of elements used for the spaces Q_h and V_h.

Figure 1.6: Linear triangular elements, used in the $P_1 \text{iso} P_2 - P_1$ approximation.



Another common choice is called $P_1 \text{iso} P_2 - P_1$: these spaces are both made of linear functions, but the velocity basis functions are defined on a mesh that is finer, so that the dimension of the space V_h is actually bigger than the one of Q_h. This choice simplifies the computation of the basis functions and of the integrals involved in the formulation, but adds the problem of managing two grids at the

same time. The elements used for this choice are shown in Figure 1.6.

Another possibility is to use a couple of spaces that would not be stable, according to Theorem 1.1, but that becomes stable adding another term to the finite element formulation. This term changes the structure of the linear system and requires some special care. We will not deal with these stabilized problems, which are discussed in [8, p. 139] or in [16, p. 451].

Chapter 2

Saddle point systems

In this Chapter, we will discuss the properties of the matrices that arise from the finite element method; we will also show some results that set the ground for the preconditioning, that will be presented further on, and finally we will investigate the singularity of the system and understand that this does not threaten the performance of the numerical solver.

2.1 Algebraic formulation

In the previous Chapter, we introduced the finite element method and used it to obtain relations (1.7) and (1.8). We then defined matrices $A_{ij} = a(\varphi_i, \varphi_j)$ and $B_{ij} = b(\varphi_j, \psi_i)$. Let us now introduce the vectors $\mathbf{u}$ and $\mathbf{p}$, that contain the unknowns u_j and p_j, i.e. the values of velocity and pressure of the approximated solution on the nodes of the grid. With all these concepts, we can now formulate the linear system of equations that arises from the finite element discretization:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix}. \tag{2.1}$$

Linear systems with this particular block structure are usually called *saddle point systems* and matrix A is often referred to as the $(1,1)$ block.

It is important to understand the structure of matrix A, since from its definition it may not be clear; in fact, the basis functions φ_i involved are vectorial functions. To simplify things, let us change notation and call ϕ_i the vectorial basis function, which will have two components. The simplest way to define a vectorial basis function, based on the piecewise polynomial functions

that we already introduced, is the following:

$$\phi_i = \begin{cases} (\varphi_i \ 0) & 1 \leq i \leq n_v \\ (0 \ \varphi_i) & n_v < i \leq 2n_v \end{cases}$$

where $2n_v$ is the total number of degrees of freedom related to the velocity components. In this way the first half of the basis functions ϕ_i generates only the first component of the velocity, while the second half generates the second component.

This choice of ϕ_i leads to a simple structure of matrix A: indeed, recall that $A_{ij} = a(\phi_i, \phi_j) = \int_\Omega \nu \nabla \phi_j \cdot \nabla \phi_i \, d\Omega$, but now the two gradients have always some zeros. In particular, if i and j are both less than n_v, then the zeros are in the same places and the scalar product can be nonzero; if instead $i \leq n_v$ and $j > n_v$, then the zeros are in complementary places and the scalar product vanishes. Thus, the matrix A has the following block structure

$$A = \begin{bmatrix} A_{1,1} & 0 \\ 0 & A_{2,2} \end{bmatrix},$$

where the two diagonal blocks are identical. For the discretization of the Navier-Stokes equation, an additional term is present, related to convection. However, the same vectorial basis functions can be used, leading to the same structure of the matrix. Also for matrix B we can make a similar reasoning, since it uses the same velocity basis functions; its structure takes the following form

$$B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}.$$

This structure of the matrices involved can be seen also from Figures 2.1 and 2.2, which depict the sparsity patterns of matrices A and B for one of the discretizations that we used: the blocks just described are clearly visible; we can also see a secondary block structure inside the primary blocks, but this characteristic depends heavily on the discretization used and on the numeration of the nodes.

In the case of the Stokes equations, the $(1, 1)$ block corresponds to a discrete Laplacian operator (or diffusion operator), while for the Navier-Stokes equations it represents a discrete convection-diffusion operator. The matrix B^T instead corresponds to a discrete gradient operator and the matrix B to a discrete divergence operator.

We will now define some other matrices that will be useful later: first of all, call n the number of velocity variables and m the number of pressure variables; the velocity-mass matrix $Q_v \in \mathbb{R}^{n \times n}$ is defined as $(Q_v)_{ij} = \int_\Omega \phi_i \phi_j \, d\Omega$. The (pressure-)mass matrix $Q \in \mathbb{R}^{m \times m}$ is defined as $Q_{ij} = \int_\Omega \psi_i \psi_j \, d\Omega$. They are

Figure 2.1: Sparsity pattern of matrix A

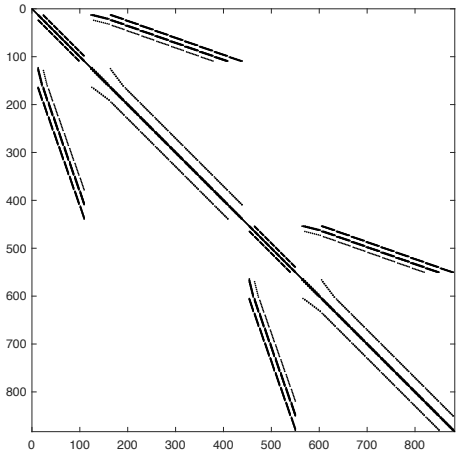
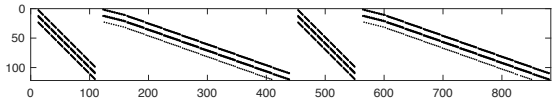


Figure 2.2: Sparsity pattern of matrix B



both square, symmetric and positive definite. Indeed

$$\begin{aligned}\mathbf{v}^T Q_v \mathbf{v} &= \sum_{j=1}^{n}\sum_{i=1}^{n} \mathbf{v}_j (Q_v)_{ij} \mathbf{v}_i \\ &= \sum_{j=1}^{n}\sum_{i=1}^{n} \mathbf{v}_j \Big(\int_\Omega \phi_j \phi_i\Big) \mathbf{v}_i \\ &= \int_\Omega \Big(\sum_{j=1}^{n} \mathbf{v}_j \phi_j\Big)\Big(\sum_{i=1}^{n} \mathbf{v}_i \phi_i\Big) \\ &= \int_\Omega \Big(\sum_{j=1}^{n} \mathbf{v}_j \phi_j\Big)^2 \geq 0\end{aligned}$$

and it is equal to 0 only if $\mathbf{v} = 0$. From the point of view of operators, the mass matrix corresponds to a discrete identity operator.

2.2 Properties of saddle point matrices

Let us underline some of the fundamental properties of the linear system (2.1). Firstly, it is important to identify the dimensions of the matrices: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times n}$, which means that A is a square matrix while B is rectangular. In the case of a Stokes problem, the matrix A is symmetric, which implies that

the whole system is symmetric. Moreover, A is also positive definite; to see this consider a vector $\mathbf{v}$ that corresponds in the finite element space to a function $v = \sum_{j=1}^{n} \mathbf{v}_j \phi_j$, then

$$\begin{aligned}\mathbf{v}^T A\mathbf{v} &= \sum_{j=1}^{n}\sum_{i=1}^{n} \mathbf{v}_j A_{ij} \mathbf{v}_i = \sum_{j=1}^{n}\sum_{i=1}^{n} \mathbf{v}_j \Big(\int_\Omega \nu \nabla\phi_j \cdot \nabla\phi_i\Big)\mathbf{v}_i \\ &= \int_\Omega \nu \Big(\sum_{j=1}^{n} \mathbf{v}_j \nabla\phi_j\Big) \cdot \Big(\sum_{i=1}^{n} \mathbf{v}_i \nabla\phi_i\Big) \\ &= \int_\Omega \nu \nabla v \cdot \nabla v \geq 0.\end{aligned}$$

This proves that A is positive semi-definite. To prove the positive definiteness, we notice that $\mathbf{v}^T A\mathbf{v} = 0$ if and only if $\nabla v = 0$, which means that v is constant in Ω. But v must vanish on the boundary, hence it must be $v \equiv 0$, which implies $\mathbf{v} = \mathbf{0}$.

Thus, we have proved that A is positive definite for the Stokes problem. In the case of the Navier-Stokes problem, the matrix A is no longer symmetric, but its symmetric part, i.e. $\frac{1}{2}(A + A^T)$, is positive definite. Moreover, the matrix A is still "positive definite" (if we accept to extend this concept to non symmetric matrices), i.e. it satisfies $(A\mathbf{v}, \mathbf{v}) > 0$ for all $\mathbf{v} \neq 0$. This can be proved showing that the bilinear form $a(\cdot, \cdot)$ is still coercive even in the case of a convection-diffusion problem (see [16, p. 292]).

Unfortunately, not much can be said about the properties of the saddle point system in the case of the Navier-Stokes equations; we will now underline some of the properties of the system for the Stokes problem.

We just showed that the matrix A is positive definite; moreover, it can be proved that its condition number is proportional to h^{-2}, where h is the dimension of the mesh used. This means that the $(1, 1)$ block becomes more and more ill-conditioned as the mesh is refined. To understand something more about the matrix B, we introduce the concept of *generalized singular values*, i.e. the numbers σ which satisfy the following generalized eigenvalue problem:

$$\begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{q} \end{bmatrix} = \sigma \begin{bmatrix} A & 0 \\ 0 & Q \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{q} \end{bmatrix}.$$

If $\sigma = 0$, the corresponding vectors $\mathbf{v}$ and $\mathbf{q}$ lay in the kernels of B and B^T respectively. Consider $\sigma \neq 0$, then

$$\begin{aligned} B^T\mathbf{q} = \sigma A\mathbf{v} &\implies (\mathbf{v}, B^T\mathbf{q}) = \sigma(\mathbf{v}, A\mathbf{v}) \\ B\mathbf{v} = \sigma Q\mathbf{q} &\implies (\mathbf{q}, B\mathbf{v}) = \sigma(\mathbf{q}, Q\mathbf{q}). \end{aligned}$$

Subtracting the two equations yields

$$(\mathbf{v}, B^T\mathbf{q}) - (\mathbf{q}, B\mathbf{v}) = \sigma[(\mathbf{v}, A\mathbf{v}) - (\mathbf{q}, Q\mathbf{q})],$$

and therefore

$$(B\mathbf{v}, \mathbf{q}) - (\mathbf{q}, B\mathbf{v}) = 0 = \sigma[(\mathbf{v}, A\mathbf{v}) - (\mathbf{q}, Q\mathbf{q})].$$

So,we have discovered that $(\mathbf{v}, A\mathbf{v}) = (\mathbf{q}, Q\mathbf{q})$.

Now, consider again the first equation: $B^T\mathbf{q} = \sigma A\mathbf{v}$. It is equivalent to $A^{-1}B^T\mathbf{q} = \sigma\mathbf{v}$. Let us take the scalar product of both sides with the vector $B^T\mathbf{q}$, which yields

$$(A^{-1}B^T\mathbf{q}, B^T\mathbf{q}) = \sigma(\mathbf{v}, B^T\mathbf{q}).$$

But we already discovered that $(\mathbf{v}, B^T\mathbf{q}) = \sigma(\mathbf{v}, A\mathbf{v})$. Hence

$$(A^{-1}B^T\mathbf{q}, B^T\mathbf{q}) = (BA^{-1}B^T\mathbf{q}, \mathbf{q}) = \sigma^2(\mathbf{v}, A\mathbf{v}) = \sigma^2(\mathbf{q}, Q\mathbf{q}).$$

We have obtained a characterization of σ depending only on $\mathbf{q}$. In the same way we can eliminate $\mathbf{q}$ and obtain a characterization of σ depending only on $\mathbf{v}$. They read

$$\frac{(BA^{-1}B^T\mathbf{q}, \mathbf{q})}{(Q\mathbf{q}, \mathbf{q})} = \sigma^2 = \frac{(B^TQ^{-1}B\mathbf{v}, \mathbf{v})}{(A\mathbf{v}, \mathbf{v})}. \tag{2.2}$$

With these in mind, we can now give an algebraic characterization to the inf-sup constant β in(1.12) (we will now use a slightly different, but equivalent, version of this condition):

$$\inf_{q_h \neq 0} \sup_{\mathbf{v_h} \neq 0} \frac{b(\mathbf{v_h}, q_h)}{\|\nabla \mathbf{v_h}\| \|q_h\|} \geq \beta.$$

Recall that $b(\mathbf{v_h}, q_h) = (q_h, \mathrm{div}\mathbf{v}_h)$, which can be written using the discrete divergence operator B as $(\mathbf{q}, B\mathbf{v})$. Moreover, $\|\nabla \mathbf{v_h}\|^2 = (A\mathbf{v}, \mathbf{v})$ and, due to the fact that Q represents a discrete identity operator, $\|q_h\|^2$ can be written also as $(Q\mathbf{q}, \mathbf{q})$. Therefore, the inf-sup conditions becomes

$$\begin{aligned}
\beta &\leq \inf_{\mathbf{q}\neq 0} \sup_{\mathbf{v}\neq 0} \frac{(\mathbf{q}, B\mathbf{v})}{(A\mathbf{v}, \mathbf{v})^{1/2}(Q\mathbf{q}, \mathbf{q})^{1/2}} \\
&= \inf_{\mathbf{q}\neq 0} \frac{1}{(Q\mathbf{q}, \mathbf{q})^{1/2}} \sup_{\mathbf{v}\neq 0} \frac{(\mathbf{q}, BA^{-1/2}A^{1/2}\mathbf{v})}{(A^{1/2}\mathbf{v}, A^{1/2}\mathbf{v})^{1/2}} \\
&= \inf_{\mathbf{q}\neq 0} \frac{1}{(Q\mathbf{q}, \mathbf{q})^{1/2}} \sup_{\mathbf{w}=A^{1/2}\mathbf{v}\neq 0} \frac{(\mathbf{q}, BA^{-1/2}\mathbf{w})}{(\mathbf{w}, \mathbf{w})^{1/2}} \\
&= \inf_{\mathbf{q}\neq 0} \frac{1}{(Q\mathbf{q}, \mathbf{q})^{1/2}} \sup_{\mathbf{w}\neq 0} \frac{(A^{-1/2}B^T\mathbf{q}, \mathbf{w})}{(\mathbf{w}, \mathbf{w})^{1/2}}.
\end{aligned}$$

The supremum is reached for $\mathbf{w} = A^{-1/2}B^T\mathbf{q}$, hence

$$\begin{aligned}\beta &\leq \inf_{\mathbf{q}\neq 0} \frac{(A^{-1/2}B^T\mathbf{q}, A^{-1/2}B^T\mathbf{q})}{(Q\mathbf{q},\mathbf{q})^{1/2}(A^{-1/2}B^T\mathbf{q}, A^{-1/2}B^T\mathbf{q})^{1/2}} \\ &= \inf_{\mathbf{q}\neq 0} \frac{(A^{-1/2}B^T\mathbf{q}, A^{-1/2}B^T\mathbf{q})^{1/2}}{(Q\mathbf{q},\mathbf{q})^{1/2}} \\ &= \inf_{\mathbf{q}\neq 0} \frac{(BA^{-1}B^T\mathbf{q},\mathbf{q})^{1/2}}{(Q\mathbf{q},\mathbf{q})^{1/2}}. \end{aligned} \tag{2.3}$$

The best possible constant β that satisfies this inequality is exactly $\beta = \inf_{\mathbf{q}\neq 0} \frac{(BA^{-1}B^T\mathbf{q},\mathbf{q})^{1/2}}{(Q\mathbf{q},\mathbf{q})^{1/2}}$ which is equal to the minimum singular value of the matrix B from (2.2): $\beta \equiv \sigma_{\min}$.

The square matrix $BA^{-1}B^T$ is usually called the (pressure-)*Schur complement* and plays a fundamental role in the numerical methods used to solve the saddle point system (2.1). We will now try to understand some of the properties of the Schur complement.

Let us first give a definition:

Definition 2.1. Consider two sequences of symmetric positive definite matrices X_n and Y_n. Assume that all the eigenvalues λ_i of $Y_n^{-1}X_n$ satisfy the relation $c_1 \leq \lambda_i \leq c_2$, with c_1 and c_2 two positive constants, independent of n. Then X_n and Y_n are said to be *spectrally equivalent.*

In our case, the sequence of matrices will be the finite element matrices, indexed by the mesh dimension h used to generate them.

Next, recall the fundamental property of the Rayleigh quotient:

Proposition 2.1. *Given a symmetric positive definite matrix M, the field of values of the Rayleigh quotient $\frac{v^TMv}{v^Tv}$ for all $v \neq 0$ is equal to the spectral interval of M, $[\lambda_{min}, \lambda_{max}]$.*

We can now state the following

Proposition 2.2. *Given two symmetric positive definite matrices A and B and a nonzero vector x, the field of values of the function $R(x) = \frac{x^TAx}{x^TBx}$ is equal to the spectral interval of the matrix $B^{-1}A$.*

Proof. The eigenvalues of $B^{-1}A$ satisfy $Av = \lambda Bv$, for some eigenvector v. Since the matrix B is positive definite, it can be written as $B = C^TC$ with C positive definite. Let us introduce a new variable $y = Cx$. The function $R(x)$ then becomes

$$R(x) = \frac{x^TAx}{x^TC^TCx} = \frac{y^TC^{-T}AC^{-1}y}{y^Ty} = \frac{y^TDy}{y^Ty},$$

where the matrix $D = C^{-T}AC^{-1}$ is still symmetric positive definite, since C is. By means of Proposition 2.1, this implies that the field of values of $R(x)$

is equal to the spectral interval of D. The eigenvalues of D satisfy $Du = \sigma u$. Substituting the definition of D and using the change of variable $w = C^{-1}u$, we obtain

$$C^{-T}AC^{-1}u = \sigma u$$

and therefore

$$Aw = \sigma C^T Cw = \sigma Bw.$$

which is exactly the characterization of the eigenvalues of $B^{-1}A$ that we stated at the beginning of the proof. Hence, the spectral interval of D is the same as the spectral interval of $B^{-1}A$ and therefore the thesis is proved. □

This proposition implies that two sequences of matrices X_n and Y_n are spectrally equivalent if and only if the function

$$R(x) = \frac{x^T X_n x}{x^T Y_n x}$$

is bounded independently of n.

We are now ready for the following

Theorem 2.1. *Consider a problem where $\partial\Omega = \Gamma_D$ (there are no Neumann boundary conditions), discretized using a stable approximation on a shape-regular quasi-uniform subdivision of $\mathbb{R}^2$. Then the Schur complement $BA^{-1}B^T$ is spectrally equivalent to the pressure mass matrix Q:*

$$\beta^2 \leq \frac{(BA^{-1}B^T\mathbf{q}, \mathbf{q})}{(Q\mathbf{q}, \mathbf{q})} \leq 1, \quad \forall \mathbf{q} \neq \mathbf{0}, \mathbf{q} \neq \mathbf{1}$$

where β is the inf-sup constant.

Proof. The lower bound is a direct consequence of (2.3). To prove the upper bound, let us consider the term $|(q_h, \mathrm{div}\mathbf{v}_h)|$ and let us use the Cauchy-Schwarz inequality, which yields

$$|(q_h, \mathrm{div}\mathbf{v}_h)| \leq \|q_h\| \, \|\mathrm{div}\mathbf{v}_h\|.$$

Consider the vectorial identity $\|\nabla \mathbf{v}_h\|^2 = \|\mathrm{div}\mathbf{v}_h\|^2 + \|\nabla \times \mathbf{v}_h\|^2$, which holds because $\mathbf{v}_h$ is in H^1 and vanishes on the boundary. A trivial consequence of this identity is that $\|\nabla \mathbf{v}_h\|^2 \geq \|\mathrm{div}\mathbf{v}_h\|^2$. If we combine this result with the previous one, we obtain

$$|(q_h, \mathrm{div}\mathbf{v}_h)| \leq \|q_h\| \, \|\nabla \mathbf{v}_h\|,$$

and therefore

$$\frac{|(q_h, \mathrm{div}\mathbf{v}_h)|}{\|q_h\| \, \|\nabla \mathbf{v}_h\|} \leq 1.$$

We can now write this expression using the discrete operators, as we did to obtain (2.3). The result is

$$\frac{(BA^{-1}B^T\mathbf{q},\mathbf{q})}{(Q\mathbf{q},\mathbf{q})} \leq 1,$$

which is the upper bound. □

The technical details, e.g. why the boundary must be all of Dirichlet type or why the subdivision must be shape-regular and quasi-uniform, have been left out; for a thorough discussion of this proof see [8, pp. 174-176]. The reason why $\mathbf{q}$ cannot be the vector $\mathbf{1}$ instead will be clear later.

If we combine this theorem with Proposition 2.2, we obtain an important consequence: regardlessly of how much we refine the mesh, the matrix $Q^{-1}BA^{-1}B^T$ will always have its eigenvalues on a very narrow interval. Moreover, it will always be positive semi-definite (as we will see later, $\mathbf{q} = \mathbf{1}$ is in the kernel of the Schur complement, so it is not positive definite) and, since Q is positive definite, also the Schur complement will always be positive semi-definite. Therefore, the matrix $Q^{-1}BA^{-1}B^T$ has a condition number that can be bounded independently of h; this is not true for matrix A, which has a condition number that grows indefinitely as h goes to 0. This fact will play a crucial role in the numerical solution of the Stokes and Navier-Stokes equations.

A similar result holds also if the Neumann boundary Γ_N is non empty; in this case the upper bound is 2. Again, for all further details see [8].

Now that we have understood the properties of the various blocks of the system, we are ready to discover some features of the whole system (2.1). We notice that it admits a factorization of the kind

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -BA^{-1}B^T \end{bmatrix} \begin{bmatrix} I & A^{-1}B^T \\ 0 & I \end{bmatrix}. \tag{2.4}$$

Let us now recall the famous Sylvester's law of inertia:

Theorem 2.2 (Sylvester's law of inertia)**.** *Let M be a symmetric matrix and let X be a nonsingular matrix of the same dimension of M. Consider the symmetric matrix $N = XMX^T$, which is said to be congruent to M. Then, M and N have the same number of positive, zero and negative eigenvalues.*

As can be seen in (2.4), the saddle point matrix is congruent to the matrix

$$D = \begin{bmatrix} A & 0 \\ 0 & -BA^{-1}B^T \end{bmatrix},$$

so they share the same number of positive, zero and negative eigenvalues. The

matrix D is block diagonal, so its eigenvalues are simply the eigenvalues of the diagonal blocks. We know that the matrix A is positive definite, so its eigenvalues are all strictly positive. We also discovered that the Schur complement $BA^{-1}B^T$ is positive semi-definite, so the $2,2-$block of D is negative semi-definite. This means that D has both positive and negative eigenvalues; hence, also the saddle point matrix has both positive and negative eigenvalues and therefore is indefinite. Moreover, as the mesh is refined and h is reduced, the number of both positive and negative eigenvalues increases. This property is sometimes referred to as *strong indefiniteness*.

Sylvester's law of inertia can be applied only when the system is symmetric, so it does not hold for the matrices arising from the Navier-Stokes problem. However, as stated in [1, p. 22], for the Navier-Stokes equations the symmetric part of the saddle point system is indefinite and therefore the whole system has eigenvalues with both positive and negative real part.

2.3 Singularity of the system

We will now derive some results which will show that, under some hypothesis, the saddle point linear system (2.1) is singular. We will then understand that this is not a problem when using a numerical solver like the GMRES method, that will be introduced in the next Chapter.

The model problem that we use has $\partial\Omega = \Gamma_D$ and is an enclosed flow, which means that there are no points on the boundary where the normal component of the velocity is different from zero. Intuitively, this means that the flow can't communicate with the outside of the domain. The Dirichlet boundary conditions impose the velocity solution, but there are no boundary conditions for the pressure. Again by intuition, we can assume that the pressure may not be uniquely defined, since in the original formulation of the equations, the pressure appears only with its derivatives. Thus, the pressure field may be defined only up to a constant in an enclosed flow. Let us try to formally show this concept.

First of all, for an enclosed flow the inf-sup condition is valid in a slightly different form:

$$\inf_{q \neq \text{constant}} \sup_{\mathbf{v} \neq 0} \frac{|(q, \text{div}\mathbf{v})|}{\|\mathbf{v}\|_{H^1} \|q\|_0} \geq \beta > 0 \tag{2.5}$$

where we have used a new norm defined as $\|q\|_0 = \|q - \bar{q}\|_{L^2}$, with $\bar{q}$ being the average of q over Ω, i.e. $\bar{q} = \frac{1}{|\Omega|} \int_\Omega q$ (to be precise, this is not a norm, but a semi-norm).

Let us now try to address the problem of the uniqueness of the solution for the weak formulation of the Stokes equations. To do so, we consider the

homogeneous weak formulation

$$\int_\Omega \nu \nabla \mathbf{u} \cdot \nabla \mathbf{v} \, d\Omega - \int_\Omega p \, \mathrm{div}\mathbf{v} \, d\Omega = 0 \quad \forall \mathbf{v} \in V \tag{2.6}$$

$$\int_\Omega q \, \mathrm{div}\mathbf{u} = 0 \quad \forall q \in Q. \tag{2.7}$$

Equation (2.6) holds for all $\mathbf{v} \in V$, so also for $\mathbf{v} = \mathbf{u}$. Equation (2.7) holds for all $q \in Q$, so also for $q = p$. This immediately leads to the equation

$$\int_\Omega \nu \nabla \mathbf{u} \cdot \nabla \mathbf{u} \, d\Omega = \nu \|\nabla \mathbf{u}\|_{L^2}^2 = 0,$$

which implies that $\mathbf{u} = 0$ due to the Poincarè-Friedrichs inequality (see [8, p. 35]).

Substituting $\mathbf{u} = 0$ in (2.6) yields $\int_\Omega p \, \mathrm{div}\mathbf{v} = 0$ for all $\mathbf{v} \in V$. Moreover, the inf-sup condition (2.5) can be equivalently stated as: for any non constant $q \in Q$, there exists $\mathbf{v} \in V$ such that

$$\frac{(q, \mathrm{div}\mathbf{v})}{\|\mathbf{v}\|_{H^1}} \geq \beta \|q\|_0.$$

Choosing $q = p$, the numerator goes to zero, as we just proved, and this means that $\|p\|_0$ must be zero. In turn, this means that the pressure field is identically equal to the mean pressure field $\bar{p}$, hence it is constant.

Now, suppose there exist two solutions to the same Stokes equations (i.e. with the same physical constants, same forcing term and same boundary conditions), $(\mathbf{u}_1, p_1)$ and $(\mathbf{u}_2, p_2)$. Their difference $(\mathbf{u}_1 - \mathbf{u}_2, p_1 - p_2)$ satisfies the homogeneous equation. We just proved that the only solution to the homogeneous problem is $\mathbf{u} = 0$ and $p =$ constant; thus, it must be $\mathbf{u}_1 - \mathbf{u}_2 = 0$ and $p_1 - p_2 =$ constant. This shows that the velocity field $\mathbf{u}$ is uniquely defined by the boundary conditions, while the pressure field p is defined only up to a constant.

Let us now try to understand how this non uniqueness in the pressure affects the properties of the saddle point system. Of course, we would like to obtain a non singular matrix; to understand whether the system is singular or not, let us study the homogeneous saddle point system, which can be explicitly expressed as

$$\begin{cases} A\mathbf{u} + B^T\mathbf{p} &= 0 \\ B\mathbf{u} &= 0 \end{cases}.$$

The matrix is non singular if and only if this system has only the trivial solution. Premultiply the first equation by $\mathbf{u}^T$ to obtain

$$\mathbf{u}^T A\mathbf{u} + \mathbf{u}^T B^T \mathbf{p} = 0.$$

Premultiply the second equation by $\mathbf{p}^T$, to obtain

$$\mathbf{p}^T B\mathbf{u} = 0.$$

This second equation implies that the first one reduces to $\mathbf{u}^T A\mathbf{u} = 0$. Since A is positive definite, this means that $\mathbf{u} = 0$, which is what we wanted. Therefore, the velocity discrete solution is unique.

Now, substitute $\mathbf{u} = 0$ into the original system, which yields

$$B^T\mathbf{p} = 0.$$

This means that the pressure discrete solution is unique only up to the null space of B^T, which may be non trivial.

To find out the vectors $\mathbf{p}$ in $\ker(B^T)$, we use the reasoning developed before to show that the pressure field is not unique. Indeed, we proved that

$$\int_\Omega p\,\mathrm{div}\mathbf{v} = 0 \quad \forall \mathbf{v} \in V \implies p = \text{constant}.$$

Some care must be put in the next passages: recall that here p represents the scalar pressure field, while $\mathbf{p}$ is the discrete vector containing the values of p in the nodes of the discretization. Moreover, when talking about the continuous formulation, $\mathbf{v}$ represents the vectorial velocity field, while when talking about the saddle point system $\mathbf{v}$ is the discrete vector containing the values of the components of the velocity field in the nodes of the discretization.

Therefore, the condition that we just wrote can be expressed in discrete form as

$$(\mathbf{p}, B\mathbf{v}) = (B^T\mathbf{p}, \mathbf{v}) = 0 \quad \forall \mathbf{v} \in \mathbb{R}^n \implies \mathbf{p} = \text{constant}.$$

The null space of B^T contains only constant vectors, hence $\ker(B^T) = \{\mathbf{1}\}$. In turn, this means that the kernel of the Schur complement $BA^{-1}B^T$ is non trivial and therefore the factorization (2.4) suggests that the saddle point system has a zero eigenvalue and is singular.

We have developed this theoretical result for the Stokes problem, but the matrix B does not change if we solve the Navier-Stokes equations. Thus, if B^T has non trivial kernel for the Stokes problem, it will be the same in the Navier-Stokes case. Therefore, in both the situations the saddle point system is singular.

We are actually able to characterize in a precise way the singularity of a generic saddle point system, with the following

Proposition 2.3. *Consider the generic matrix*

$$M = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix},$$

if $(A\mathbf{v}, \mathbf{v}) > 0$ *for all* $\mathbf{v} \neq 0$*, then*

$$\ker(M) = \left\{ \begin{bmatrix} \mathbf{0} \\ \mathbf{p} \end{bmatrix} \middle| \, \mathbf{p} \in \ker(BA^{-1}B^T) \right\}.$$

Proof. We already saw that a generic vector in the kernel of M has a first block equal to $\mathbf{0}$ and a second block $\mathbf{p}$ which belongs to the kernel of B^T. Therefore, we just need to prove that the kernel of the Schur complement is the same as the one of B^T. Trivially, all the vectors in $\ker(B^T)$ are also in $\ker(BA^{-1}B^T)$.

Suppose that there exists a vector $\mathbf{v}$ in $\ker(BA^{-1}B^T)$ but not in $\ker(B^T)$

$$BA^{-1}B^T\mathbf{v} = 0.$$

Since $B^T\mathbf{v} \neq 0$, call it $\mathbf{u}$; then, since A is invertible, also $A^{-1}\mathbf{u} \neq 0$, call it $\mathbf{w}$. Thus

$$BA^{-1}B^T\mathbf{v} = BA^{-1}\mathbf{u} = B\mathbf{w} = 0.$$

Therefore, $\mathbf{w}$ must be in $\ker(B)$. Basic linear algebra tells us $\ker(B) = \mathrm{im}(B^T)^\perp$; $\mathbf{w}$ is in the kernel of B, $\mathbf{u}$ instead is a generic vector in the range of B^T, therefore $\mathbf{w}$ and $\mathbf{u}$ must be orthogonal:

$$0 = (\mathbf{w}, \mathbf{u}) = (A^{-1}\mathbf{u}, \mathbf{u}).$$

Since $(A\mathbf{v}, \mathbf{v}) \neq 0$ for all $\mathbf{v} \neq 0$, and the same holds also for the inverse A^{-1}, this implies that $\mathbf{u} = 0$, which is a contradiction since we supposed that $\mathbf{v}$ is not in $\ker(B^T)$. □

Now that we know that the system is singular, we want to discover how we can solve such a system. We want the system to be *consistent*, which means that a solution exists. Consider a generic linear system $M\mathbf{x} = \mathbf{b}$; to be sure of the existence of a solution $\mathbf{x}$, it is sufficient to have $\mathbf{b} \in \mathrm{im}(M)$. In our case, the load vector is $\mathbf{b} = \begin{bmatrix} \mathbf{f} & \mathbf{0} \end{bmatrix}^T$.

We know that $\mathrm{im}(M) = \ker(M^T)^\perp$, but M^T is very similar to the original saddle point system, with the only difference that the $(1,1)$ block is transposed. However, the kernel depends only on the matrix B^T and not on the $(1,1)$ block, so in our case $\ker(M^T) = \ker(M) = \begin{bmatrix} \mathbf{0} & \mathbf{1} \end{bmatrix}^T$. The structure of the kernel and of the load vector imply that $\mathbf{b}$ is always orthogonal to $\ker(M^T)$ and hence the system is consistent.

The method that will be used to solve the system is the GMRES method, that will be described in the next Chapter. However, at the moment we want to know only if this method will be able to find a solution for this particular singular system. A result from [6] states that GMRES finds a solution of the system $M\mathbf{x} = \mathbf{b}$ for all load vectors and for all starting vectors, if and only if the null spaces of M and M^T are the same. Here, the matrix M is not exactly the saddle point matrix, but it is the preconditioned matrix, that will be presented later in this work. To check the previously cited condition is not easy, since the preconditioned matrix has a very complicated structure. However, we are not interested in finding out if GMRES converges for all $\mathbf{b}$, but only for those for which the system is consistent. The next theorem, taken again from [6], helps us in this sense:

Theorem 2.3. *Suppose the linear system $M\mathbf{x} = \mathbf{b}$ is consistent, with M a singular square matrix. If $\ker(M) \cap \mathrm{im}(M) = \{\mathbf{0}\}$, then GMRES is able to construct a solution without breaking down.*

Unfortunately, this condition is very difficult to check, due to the great complexity of the structure of the matrix involved. In [8, pp. 396-397], this condition is checked for a preconditioner different from the one that will be used in this work, but that allows simpler computations. In Appendix A, we present a proof of the check of this condition for the preconditioner that we used.

Chapter 3

Iterative methods for linear systems

In this Chapter, we will present the main method that will be used to solve linear systems, GMRES; we will develop the algorithm and understand its main properties about the speed of convergence. Then, we will introduce Multigrid methods and we will try to show that they can be excellent preconditioners for our systems.

3.1 Projection methods

We now address the problem of solving the linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$; there exist many methods to do so, but very few of them are suitable if the dimension of the matrix if very large. Indeed, for a large sparse matrix, it is not possible to exploit methods which factorize the matrix (like Gaussian elimination), since they would destroy the sparsity, producing dense matrices of the same dimension of A. Instead, there are methods that approximate the solution without factorizing the matrix or even without knowing its explicit expression, but only accessing it through matrix-vector products (i.e. the matrix A is not known, but it is known its action on any vector). These methods represent the state-of-the-art in terms of algorithms to solve large linear systems.

To start, let us assume that we want to find an approximation of the solution x in a subspace of $\mathbb{R}^n$, that we will call K. In order to find it easily, let us suppose that the dimension of K is $m \ll n$. m conditions are needed to determine the approximation inside K: these constraints are given imposing that the residual $b - Ax$ is orthogonal to another subspace L of dimension m. This condition is known as *Petrov-Galerkin* condition. So, starting from an initial approximation

x_0, the problem reduces to

$$\text{find } \tilde{x} \in x_0 + K, \quad \text{such that } b - A\tilde{x} \perp L. \tag{3.1}$$

Suppose to know a basis for K and L: $V \in \mathbb{R}^{n\times m}$ contains the m vectors of the basis of K, while $W \in \mathbb{R}^{n\times m}$ the ones for L. The approximation sought can then be written as $\tilde{x} = x_0 + Vy$, for some vector of coordinates $y \in \mathbb{R}^m$ to be found. Let us try to impose the condition of orthogonality: the residual can be written as

$$r = b - A\tilde{x} = b - Ax_0 - AVy$$

and the orthogonality to L is written as

$$0 = W^T r = W^T b - W^T A x_0 - W^T AVy = W^T r_0 - W^T AVy.$$

Therefore, y is found as

$$y = (W^T AV)^{-1} W^T r_0$$

and the approximation searched is

$$\tilde{x} = x_0 + Vy = x_0 + V(W^T AV)^{-1} W^T r_0. \tag{3.2}$$

From (3.2), it is possible to construct an iterative method: take an initial guess x_0, choose the spaces K and L and calculate a basis; then calculate $\tilde{x}$ and evaluate its residual. If the approximation is sufficiently good then stop, otherwise repeat the same steps starting with $x_0 = \tilde{x}$ and enlarging the spaces.

For this method to be well defined, the matrix $W^T AV$ needs to be nonsingular, but this is not always true, even if A is nonsingular. However, the following result holds

Proposition 3.1. *If A is positive definite and $L = K$, or if A is nonsingular and $L = AK$, then the matrix $W^T AV$ is nonsingular for every choice of bases V and W of K and L respectively.*

Let us try to understand whether the approximation given by (3.2) is a good one or not. It turns out that, due to the orthogonal projection involved in the process, this is the best possible approximation, in some precise sense, that can be found in the space K. This result is formalized in the following

Proposition 3.2. *If A is an arbitrary square matrix and $L = AK$, then $\tilde{x}$ is the result of a projection method onto K, orthogonally to L, starting from x_0, if and only if it minimizes the norm of the residual onto $x_0 + K$, i.e.*

$$R(\tilde{x}) = \min_{x \in x_0 + K} R(x) \quad \textit{where } R(x) = \|b - Ax\|_2.$$

A similar result holds also in the case of a symmetric positive definite matrix A and with $L = K$, only that in this case the quantity minimized is the error instead of the residual.

This means that the approximation $\tilde{x}$ is close to the exact value if the spaces K and L are chosen properly and if they get better and better when their dimension increases. It is therefore crucial to choose these spaces appropriately.

3.1.1 Krylov subspaces

A Krylov subspace K_m is defined as

$$K_m(A, v) = span\{v, Av, A^2v, \ldots, A^{m-1}v\}.$$

It is immediately clear that K_m is the space of all the vectors of $\mathbb{R}^n$ that can be written as $p(A)v$, where p is a polynomial of degree at most $m-1$.

Let us define the *minimal polynomial of* v as the nonzero monic polynomial p of lowest degree such that $p(A)v = 0$ and the *grade of* v as the degree of p. Obviously, the grade of v is smaller or equal than the degree of the minimal polynomial of the matrix A, and therefore it is smaller or equal than n.

Let us now see some important properties that will be useful when using Krylov subspaces for projection methods.

Proposition 3.3. *Call μ the grade of v, then*

$$dim(K_m) = \min\{m, \mu\}.$$

Proof. For K_m to have dimension m, the vectors $v, Av, \ldots, A^{m-1}v$ must be linearly independent, which means that, for any choice of m scalars α_i, $i = 0, \ldots, m-1$ not all vanishing, it must be $\sum_{i=0}^{m-1} \alpha_i A^i v \neq 0$. This implies that there does not exist any nonzero polynomial q of degree smaller or equal than $m-1$ such that $q(A)v = 0$, hence v has grade at least m, i.e. $\mu \geq m$. So, if $m \leq \mu$, the space K_m has dimension m.

Let us consider the definition of grade of v: there exists a sequence of coefficients c_i such that $\sum_{i=0}^{\mu} c_i A^i v = 0$, which implies that

$$\sum_{i=0}^{\mu-1} c_i A^i v + c_\mu A^\mu v = 0 \implies A^\mu v = -\sum_{i=0}^{\mu-1} \frac{c_i}{c_\mu} A^i v.$$

The important conclusion is that $A^\mu v$ is a linear combination of vectors in K_μ.

Now, consider a vector $z \in K_{\mu+1}$: it can be written as

$$z = \sum_{i=0}^{\mu} \beta_i A^i v = \sum_{i=0}^{\mu-1} \beta_i A^i v + \beta_\mu A^\mu v.$$

The term with the sum belongs to K_μ following from the definition, while the term $\beta_\mu A^\mu v \in K_\mu$ due to the previous observation. Hence, $z \in K_\mu$, which means that $K_{\mu+1}$ and K_μ are the same space and $K_{\mu+1}$ has dimension μ and not $\mu+1$.

Thus, if $m \leq \mu$, the dimension is m, while if $m > \mu$, the dimension is μ, which is the thesis. □

So, if for example the starting vector v is an eigenvector of A, then any vector of the form $A^i v$ will be parallel to v, yielding a space K_m of dimension 1, regardlessly of m. Indeed, if $Av = \lambda v$, then the minimal polynomial is simply $p(x) = x - \lambda$, which has degree 1. A generalization of this observation is that if the starting vector v is a combination of l eigenvectors of A, then the grade of v and the maximum dimension of any Krylov subspace starting from v will be l.

A projection method where the spaces K and L are Krylov subspaces is called *Krylov subspace method*; the most used methods to solve large linear systems all belong to this class. In order to build such methods, we need a good basis for K_m: due to the properties of power iteration, the vectors $A^i v$ tend to become parallel and therefore the generating vectors of the Krylov subspace are not a good basis to use. Instead, we can try to produce an orthonormal basis, using the Gram-Schmidt procedure, which in this context takes the name of *Arnoldi procedure*. Algorithm 3.1 illustrates the method.

Algorithm 3.1 Arnoldi procedure

1: Choose v_1 such that $\|v_1\| = 1$
2: **for** j=1,. . . , m **do**
3: **for** i=1,. . . , j **do**
4: Compute $h_{ij} = (Av_j)^T v_i$
5: **end for**
6: Compute $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
7: $h_{j+1,j} = \|w_j\|_2$
8: **if** $h_{j+1,j} = 0$, stop
9: $v_{j+1} = w_j / h_{j+1,j}$
10: **end for**

Let us underline some of the fundamental properties of this algorithm. First of all, the outputs of the procedure are two matrices: $V_m \in \mathbb{R}^{n \times m}$, the matrix containing the orthonormal vectors that form the basis for K_m, and $\bar{H}_m \in \mathbb{R}^{(m+1) \times m}$, the matrix built with the coefficients h_{ij}, which is a Hessenberg matrix by construction. Define also the matrix $H_m \in \mathbb{R}^{m \times m}$, obtained from $\bar{H}_m$

by deleting its last row. Following from the orthogonalization procedure, these matrices obey to the following relations:

$$AV_m = V_{m+1}\bar{H}_m \qquad V_m^T A V_m = H_m. \tag{3.3}$$

The following proposition is important to understand Krylov subspace methods:

Proposition 3.4. *Algorithm 3.1 breaks down at step j, meaning that $h_{j+1,j} = 0$ in line 8, if and only if the grade of v_1 is j.*

This means that the Arnoldi procedure is always able to compute a full orthonormal basis for the space K_m, since either the algorithm does not break down and computes all the m vectors, or it stops when the value of j has reached the dimension of the space. Therefore, increasing m it will happen that either the dimension of the space grows (and the Arnoldi procedure will be able to compute a new vector for the basis) or that the maximum dimension is reached (and Arnoldi breaks down). In the first case, the next approximation found will surely be more precise than the previous one, meaning that the residual will not be larger, since it is minimized over a space of larger dimension. In the second case, it can be proved that the space K_m of largest possible dimension is invariant under A (it follows from the proof of Proposition 3.3) and in this case the solution of any projection method is exact. Thus, the solution either improves or is exact. This leads to an iterative procedure, where at each iteration the value of m is increased, until either the exact solution is reached or a sufficiently good approximation is found. Of course, in finite precision arithmetic, the exact solution will never be found and the method will have to be stopped according to a threshold on the residual.

A straightforward property of the method just described is that the computational cost of a single iteration grows as the iterations count proceeds, since every time there is one more vector against which the new vectors must be orthonormalized. There are though some lucky cases, where the properties of the matrix of the system allow for a *short recurrence*: this means that at every iteration, the new vector must be orthogonalized only against some of the previously calculated vectors. The most famous example is the Conjugate Gradient, where a 3-term recurrence holds for matrices that are symmetric positive definite. Also for indefinite symmetric matrices there are methods which allow short recurrence.

Therefore, in some cases, the Arnoldi algorithm may be simplified, leading to a simpler method to solve the linear system. The Faber-Manteuffel theorem characterizes all the matrices for which a short recurrence is possible: this

property is in some sense related to a generalized concept of symmetry of the matrix, meaning that the transpose matrix does not need to be equal to the matrix itself, but it must hold that $A^T = q(A)$, where q is a polynomial of low degree; the smallest the degree of q, the shortest the recurrence will be. This of course includes the case of symmetric matrices, as we already knew. A more detailed discussion of the Faber-Manteuffel Theorem can be found in [18, pp. 206-208].

Therefore, a generic non-symmetric matrix will not benefit of a short recurrence in the Arnoldi procedure and the cost per iteration will grow every time. This fact leads to two classes of iterative methods: the ones which perform the complete Arnoldi procedure and therefore satisfy an optimality property like the one defined in Proposition 3.2, and the ones which perform a variation of the Arnoldi procedure where the maximum number of vectors against which to orthogonalize is fixed, which leads to non-optimality but to a computational cost that does not grow as the iterations proceed. The most known methods of each class are GMRES and BiCGSTAB respectively. In this work we will use only the GMRES method, which is presented in the next Section.

3.2 GMRES

The *Generalized Minimum Residual* method (GMRES) is a projection method that takes $K = K_m$ and $L = AK_m$. As stated in Proposition 3.2, a method that uses such spaces will minimize the residual over K at every iteration. The method starts with an initial guess of the solution, x_0; the first vector for the Krylov space is chosen to be the normalized initial residual $v_1 = r_0/\|r_0\|$.

We will now derive the method using the optimality condition. A generic approximation of the solution x at the $m-$th iteration will belong to the space $x_0 + K_m$. If we suppose to know a basis for K_m, whose vectors are stored in the matrix V_m as columns, then x can be expressed as

$$x = x_0 + V_m y,$$

where y is a vector that represents the coordinates of x in the space K_m. We want to find $y \in \mathbb{R}^m$ such that x minimizes the residual among all the vectors in

$x_0 + K_m$. The residual can be expressed as follows, exploiting relations (3.3)

$$\begin{aligned} b - Ax =& b - A(x_0 + V_m y) \\ =& b - Ax_0 - AV_m y \\ =& r_0 - V_{m+1}\bar{H}_m y. \end{aligned} \tag{3.4}$$

Define β as the norm of the initial residual. If we call e_1 the first vector of the canonical basis, then $v_1 = V_{m+1}e_1$ and so

$$b - Ax = \beta v_1 - V_{m+1}\bar{H}_m y = V_{m+1}(\beta e_1 - \bar{H}_m y).$$

The quantity to be minimized is the norm of the residual, but since the columns of V_{m+1} are orthonormal, it follows that $\|b - Ax\| = \|\beta e_1 - \bar{H}_m y\|$. Therefore, the $m-$th iterate x_m can be found as

$$x_m = x_0 + V_m y_m$$

where y_m is found as

$$y_m = \text{argmin}\|\beta e_1 - \bar{H}_m y\|.$$

The question now is how to determine such y. In order to do so, let us consider the QR decomposition of the matrix $\bar{H}_m$. Recall that any square or rectangular matrix can be decomposed into the product of an orthogonal matrix Q and an upper triangular matrix R. In our case, $\bar{H}_m$ has dimensions $(m+1) \times m$, hence Q will be a matrix $(m+1) \times (m+1)$ and R will be $(m+1) \times m$, with the last row made entirely of zeros. With this in mind, we can say that

$$\beta e_1 - \bar{H}_m y = \beta e_1 - QRy = Q(\beta Q^T e_1 - Ry).$$

Define $g = \beta Q^T e_1$ and R_1 as the $m \times m$ submatrix of R, obtained eliminating the last row, made of zeros. In the same way, call g_1 the first m components of g and $\tilde{g}$ the last component. Then

$$\beta Q^T e_1 - Ry = \begin{bmatrix} g_1 \\ \tilde{g} \end{bmatrix} - \begin{bmatrix} R_1 \\ \mathbf{0} \end{bmatrix} y.$$

The quantity that we want to minimize is $\|\beta e_1 - \bar{H}_m\|$, which is equal to $\|g - Ry\|$, since the matrix Q is orthogonal and does not change the value of the Euclidean norm. The minimum is achieved when $R_1 y = g_1$, since in this way the quantity inside the norm has m components that are zero and only the last

component different from zero. Therefore, the vector that we were looking for is $y = R_1^{-1} g_1$. Moreover, the norm of the residual is just equal to $\tilde{g}$, since all the other components vanish.

All the steps that we have described so far are not computationally expensive: the QR decomposition has a computational cost that grows as $\mathcal{O}(m^3)$, but the value of m is usually very small compared to the dimension of the matrix of the system; the solution of the linear system needed to calculate y is very cheap, since the matrix R_1 is triangular, moreover, this computation is not required at every iteration, since the residual is known even without knowing the approximation x. Thus, the vectors y and x can be calculated only at the last iteration, when the residual is below the required tolerance. The steps that are instead very costly are the ones needed to determine the basis of the Krylov space: indeed, we need to use the Arnoldi procedure, which requires matrix-vector products involving the large matrix of the system. Moreover, the procedure does not need to be repeated every time, since at every iteration we can just update the matrices V and H previously calculated, saving a lot of time.

We will now show that the same method can be derived using equation (3.2). In our case, $V = V_m$ and $W = AV_m$. Hence, the solution is

$$x = x_0 + V_m (V_m^T A^T A V_m)^{-1} V_m^T A^T r_0.$$

This means that the vector y can be written as $(V_m^T A^T A V_m)^{-1} V_m^T A^T r_0$. We now use again relations (3.3), to simplify this expression:

$$y = (\bar{H}_m^T V_{m+1}^T V_{m+1} \bar{H}_m)^{-1} \bar{H}_m^T V_{m+1}^T r_0.$$

The columns of V_{m+1} are orthonormal, therefore the matrix $V_{m+1}^T V_{m+1}$ is equal to the identity matrix of dimension $m+1$. Recall that r_0 can be expressed as βv_1.

$$y = (\bar{H}_m^T \bar{H}_m)^{-1} \bar{H}_m^T V_{m+1}^T \beta v_1.$$

For the same reason as before, $V_{m+1}^T v_1 = e_1$. Hence

$$y = (\bar{H}_m^T \bar{H}_m)^{-1} \bar{H}_m^T \beta e_1.$$

This is a system of normal equations that finds the least square minimizer y of the function $\bar{H}_m y - \beta e_1$. Therefore, we have proved that using the theory of projection methods with spaces $K = K_m$ and $L = AK_m$, we obtain the same method as before.

Having in mind how the method works, we can now try to understand the requirements of Theorem 2.3: indeed, if the kernel and the range of the system

matrix have trivial intersection, then every generating vector of the Krylov subspace is different from zero. To explain this take $A^j v$, which is obviously in $\mathrm{im}(A)$ and thus cannot be in $\ker(A)$; the next vector, $A^{j+1}v$, cannot be zero then, and the sequence is well defined (up to the break down point) even if the matrix is singular.

3.2.1 Properties

As we said in the previous Section, for a generic non symmetric matrix, the GMRES method have a long recurrence, which means that at every iteration it is required to orthonormalize the new vector against all the previous ones. To bound the computational cost of the method, the GMRES is sometimes used in its restarted version: we fix a maximum number of vectors that can be stored, p, and when p iterations are complete, the method deletes all the vectors saved in the memory and restarts using as initial vector the last computed approximation. Obviously, in this way the optimality property is lost, but at least we can bound the computational cost.

We will now present some of the fundamental properties of the GMRES method. As a first guess, we may think that the method can stop too early if the Arnoldi procedure breaks down, leaving us with a poor solution. Instead, the following proposition assures us that this cannot happen.

Proposition 3.5. *Suppose to use the GMRES method on a nonsingular matrix. Then the algorithm breaks down at step j, i.e. $h_{j+1,j} = 0$, if and only if the approximate solution x_j is exact.*

Let us now try to understand the speed of convergence of the GMRES method. First of all, we notice that a generic approximation x, since it belongs to $x_0 + K_m$, can be expressed as

$$x = x_0 + \sum_{j=0}^{m-1} \alpha_j A^j r_0,$$

following from the definition of Krylov subspace. Therefore, also the residual can be expressed in terms of powers of A:

$$r = b - Ax = b - Ax_0 - \sum_{j=0}^{m-1} \alpha_j A^{j+1} r_0 = r_0 - \sum_{j=1}^{m} \alpha_{j-1} A^j r_0.$$

Hence, the residual can be written as $r = \bar{p}(A)r_0$, with $\bar{p}(x)$ a polynomial of degree m such that $p(0) = 1$. Due to the minimization property of GMRES, the

polynomial $\bar{p}(x)$ is the one which minimizes the residual and therefore

$$\|r_m\| = \min_{\substack{p \in P_m \\ p(0)=1}} \|p(A)r_0\|. \tag{3.5}$$

We can now prove the following important result

Theorem 3.1 (Finite termination property)**.** *Consider a nonsingular system with matrix $A \in \mathbb{R}^{n\times n}$; the GMRES method finds the solution within n iterations.*

Proof. Define $p(x) = \det(A - xI)$, the characteristic polynomial of A. We know that $p(x)$ has degree n, $p(0) \neq 0$ since A is nonsingular and $p(A) = 0$ from the Cayley-Hamilton theorem.

Build the polynomial $\bar{p}(x) = \dfrac{p(x)}{p(0)}$, so that $\bar{p}(0) = 1$. From (3.5), it follows that

$$\frac{\|r_n\|}{\|r_0\|} \leq \|q(A)\| \quad \forall q \in P_n \text{ such that } q(0) = 1.$$

Choosing $q = \bar{p}$, we obtain that $\|r_n\| \leq 0$, which implies that $r_n = \mathbf{0}$ and that the approximation x_n is exact. □

Thus, in the worst case the method will converge in a number of iterations equal to the dimension of the matrix. However, if the matrix is very large, this result is not very useful. Luckily, under some hypothesis, the convergence can be much faster.

Suppose that the matrix A is diagonalizable; then, there exist a nonsingular matrix V and a diagonal matrix D such that $A = VDV^{-1}$. Moreover, given a polynomial p, $p(A) = Vp(D)V^{-1}$, because any power of A will have the form $VDV^{-1}VDV^{-1}V \dots V^{-1}VDV^{-1}$, but $V^{-1}V = I$ and what remains is $VDD \dots DV^{-1}$.

Theorem 3.2. *Consider a nonsingular diagonalizable matrix $A = VDV^{-1}$. Let x_k be the kth approximation found by GMRES, then for all $p \in P_k$ such that $p(0) = 1$*

$$\frac{\|r_k\|}{\|r_0\|} \leq \kappa(V) \max_{z \in \sigma(A)} |p(z)| \tag{3.6}$$

where $\sigma(A)$ is the spectrum of A.

Proof. We already know that, for all $p \in P_k$ such that $p(0) = 1$

$$\frac{\|r_k\|}{\|r_0\|} \leq \|p(A)\|.$$

We can estimate the norm of $p(A)$ as follows

$$\|p(A)\| \leq \|V\|\|V^{-1}\|\|p(D)\|.$$

The term $\|V\|\|V^{-1}\|$ is the condition number of the matrix V, $\kappa(V)$.

D is diagonal with entries equal to the eigenvalues of A; it follows from the definition of matrix norm that $\|D^n\|$, for a generic power n, is a convex combination of the $n-$th powers of the eigenvalues of A. Hence, it is smaller than $\max_{z\in\sigma(A)}|z^n|$. Therefore, also $\|p(D)\| \leq \max_{z\in\sigma(A)}|p(z)|$ and the thesis is proved. □

With this result in mind, we can now state the following fundamental result about the speed of convergence of the GMRES method.

Theorem 3.3. *Consider a nonsingular diagonalizable matrix A. If it has only k distinct eigenvalues, then GMRES finds the exact solution in at most k iterations.*

Proof. Let us use equation (3.6). Since it holds for all the polynomials, then it holds also for the minimum over the polynomials:

$$\frac{\|r_k\|}{\|r_0\|} \leq \kappa(V) \min_{\substack{p\in P_k \\ p(0)=1}} \max_{z\in\sigma(A)} |p(z)|.$$

If we can find a suitable polynomial of degree k that is equal to zero if evaluated on all the eigenvalues of A, then $\max_{z\in\sigma(A)}|p(z)| = 0$ and we have proved that $\|r_k\| = 0$ and that the approximation x_k is exact. Such a polynomial certainly exists, since we are looking for a polynomial that passes through the point $(0,1)$ and that passes through another k points of the kind $(\lambda, 0)$, with λ the eigenvalues of A. Thus, this polynomial must satisfy $k+1$ conditions and since it has degree k, such a polynomial is uniquely determined. □

Another similar result states that, if $\mathbf{b}$ is a linear combination of k eigenvectors of A, then GMRES will find the exact solution in at most k iterations. For more details about the speed of convergence of GMRES, see [18, p. 171] and [13, p. 33].

Unfortunately, the eigenvalues of the matrix of the system are not always the best indicator of the speed of convergence of GMRES: indeed, in [10] it is shown that, given a nonincreasing positive sequence of numbers f_k, there always exists a matrix A and a vector r^0 such that $\|r_k\| = f_k$, where r_k is the residual at the $k-$th iteration of GMRES applied to the system $Ax = b$, with initial residual r_0. Moreover, the matrix A can be chosen to have any eigenvalues. Therefore, there are matrices with very nice spectral properties which produce a sequence of residual norms that is slowly decreasing. The problem of finding a better indicator for the speed of convergence is still not well understood; however, for the matrices that we used, the eigenvalues were able to predict accurately the behavior of GMRES.

3.2.2 Preconditioning

We have seen some important properties about the speed of convergence of GMRES. Sometimes however, for some very ill-conditioned systems, this method can become extremely slow or may even fail to converge. In these situations, a commonly used technique to enhance the performance of the method is to *precondition* the matrix. Suppose that we want to solve the extremely complicated system $Ax = b$; instead, we can solve the preconditioned system $P^{-1}Ax = P^{-1}b$, which has the same solution, where the matrix P is called the *preconditioner*.

For some particular choices of P, the matrix $P^{-1}A$ has properties that allow for a much faster solution than the original system. Intuitively, the matrix P should approximate in the best possible way A, but it should also be computationally cheap to compute and to invert. In this way, the condition number of the system is reduced and the system is easier to solve. The more P gets close to A, the more the eigenvalues will cluster around the value 1; in turn, exploiting Theorem 3.3, this means that GMRES will converge faster. Therefore, one of the possible ways to assess the quality of a preconditioner is to evaluate how many eigenvalues are clustered around 1 and how many clusters of eigenvalues are present.

One of the simplest and most frequent choices for preconditioners is given by an incomplete factorization: computing the exact factorization of a large sparse matrix is not feasible, since it would produce dense matrices; it is instead possible to produce an inexact factorization that preserves sparsity. These are called incomplete LU factorization or incomplete Cholesky factorization, they are cheap to compute but approximate well the original matrix. The inverses of these factorizations are easy to compute, since the matrices obtained are triangular. This simple technique unfortunately does not work well in the case of saddle point systems. It is better to use block preconditioners, in which every block of the system is preconditioned in a proper way.

The preconditioned GMRES method has an algorithm that is almost identical to the original one, as it is shown in [18, p. 282]. The preconditioning can be done in two ways, left and right: left preconditioning solves the system $P^{-1}Ax = P^{-1}b$, while right preconditioning solves the system $AP^{-1}y = b$ and then finds x as $x = P^{-1}y$. Algorithm 3.2 presents the left preconditioned GMRES method.

In the next Chapter we will discuss more in detail how to build efficient preconditioners for saddle point problems.

3.2.3 Arnoldi method for eigenvalues

We will now illustrate a method to find a large number of eigenvalues of a matrix, which is based on Krylov subspaces and the Arnoldi procedure. We will use this

Algorithm 3.2 Left-preconditioned GMRES

Compute $r_0 = P^{-1}(b - Ax_0)$, $\beta = \|r_0\|$ and $v_1 = r_0/\beta$
Choose the restart parameter p
for j=1,. . . , p **do**
  Compute $w = P^{-1}Av_j$
  for i=1,. . . , j **do**
    $h_{i,j} = (w, v_i)$
    $w = w - h_{i,j}v_i$
  end for
  Compute $h_{j+1,j} = \|w\|$ and $v_{j+1} = w/h_{j+1,j}$
  Update matrices V and $\bar{H}$
  Compute the minimizer y_j and the approximation $x_j = x_0 + Vy_j$
  If satisfied Stop, else continue
end for
If satisfied Stop, else Restart with $x_0 = x_p$

method to obtain useful information about the spectrum of the matrices of the saddle point systems that we want to solve.

The problem that we want to address is to find an eigenvalue λ and an eigenvector u such that $Au = \lambda u$. We will discuss projection methods, as in the case of the solution of linear systems. Suppose then to have an $m-$dimensional subspace of $\mathbb{R}^n$ and call it K; we want to find an approximate eigenpair $\tilde{\lambda}$, $\tilde{u} \in K$ such that the residual $A\tilde{u} - \tilde{\lambda}\tilde{u}$ is orthogonal to the space K:

$$(A\tilde{u} - \tilde{\lambda}\tilde{u}, v) = 0 \quad \forall v \in K.$$

Suppose to know a basis for K, stored in the columns of the matrix $V = [v_1 \dots v_m]$. The vector $\tilde{u}$ can be expressed as Vy, for some coordinate vector y. Then, the orthogonality condition can be written as

$$(AVy - \tilde{\lambda}Vy, v_j) = 0 \quad \forall j = 1, \dots, m$$

which in a more compact form becomes

$$V^T(AVy - \tilde{\lambda}Vy) = 0.$$

Let us call $B_m = V^T AV$; then, since $V^T V = I$, we obtain an eigenrelation for the matrix B_m

$$B_m y = \tilde{\lambda} y.$$

Therefore, instead of computing the eigenvalues of A, which is a large sparse matrix, we just compute the eigenvalues of B_m, which is a smaller dense matrix. The eigenvectors of A are then found by $\tilde{u} = Vy$. This method is known as

Rayleigh-Ritz procedure. Many convergence results can be found in [19, pp. 97-105].

This technique can be used choosing K to be a Krylov subspace. In this case, the basis V must be computed using the Arnoldi procedure; the matrix B_m does not need to be computed explicitly, since from relations (3.3) it follows that $B_m = H_m$, and so at the end of the iterations of Arnoldi we already know this matrix.

3.3 Multigrid

We now present a class of methods completely different from the Krylov space methods, but with very interesting properties. Indeed, Multigrid methods are among the most efficient solvers or preconditioners known in numerical linear algebra; under suitable assumptions on the problem to be solved and under careful choice of the parameters involved, these methods can produce a convergence rate that does not depend on the dimension of the problem. This means that, if we are trying to solve a linear system arising from PDE discretization with mesh size h, using a Multigrid scheme as preconditioner, then the number of iterations of GMRES will be independent of h. Other solvers (or preconditioners) usually experience an increment of the number of iterations as h becomes smaller and smaller; this leads to a computational cost that grows faster than $\mathcal{O}(n)$. With multigrid methods instead, a linear growth of the computation cost is achievable.

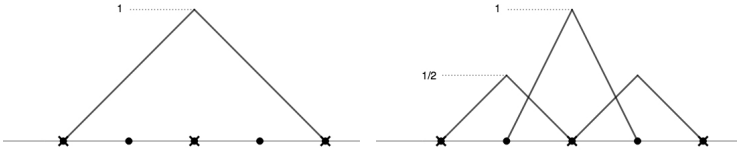
To understand the main idea of Multigrid, consider a problem to be solved on a mesh of size h. If the same problem were to be solved on a mesh of size $2h$, the linear system would be smaller (in two dimensions approximately four times smaller) and thus easier to solve. The idea behind Multigrid methods is to have two grids, a coarse one and a fine one, and to pass from one another using some suitable operators, called prolongation operator (going from the coarse grid to the fine one) and restriction operator; in this way, the problem can be solved on the coarse grid and then taken back to the fine mesh.

Call S_{2h} the space of functions defined on the coarse grid (i.e. the space of linear combinations of basis functions defined on the coarse grid) and S_h the space of functions defined on the fine grid. Obviously, $S_h = S_{2h} + B_h$, where B_h represents the space of functions defined on the nodes that belong to the fine grid but not to the coarse one. Therefore, a general function in S_h has some components that can be represented in the coarse grid and some other that would be lost passing from one grid to the other. Thus, before moving from the fine to the coarse grid, these latter components must be reduced, so that the representation in S_{2h} will be close to the original one in S_h. This is done using a smoothing operator. All these steps will now be explained in detail.

3.3.1 Prolongation and restriction

To understand how prolongation and restriction work, it is important to understand that S_{2h} is included in S_h, meaning that every function in S_{2h} can be represented exactly in S_h. Figure 3.1 explains this idea: the basis function in the coarse grid can be represented as the superposition of three basis functions in the fine grid.

Figure 3.1: Basis function in S_{2h} and S_h.



Define I_{2h}^h as the prolongation operator that maps S_{2h} into S_h. Obviously, for what we just said, if $v_{2h} \in S_{2h}$ then $I_{2h}^h v_{2h} = v_{2h}$. This means that $S_{2h} \subset S_h$. To determine an expression for I_{2h}^h, consider ϕ_j^{2h}, a basis for S_{2h}, and ϕ_j^h, a basis for S_h. Due to the inclusion of the spaces, it must be

$$\phi_j^{2h} = \sum_{i=1}^{n_h} p_{ij} \phi_i^h,$$

where n_h is the dimension of S_h and p_{ij} are coefficients to be determined.

Since $v_{2h} \in S_{2h}$, it can be written as

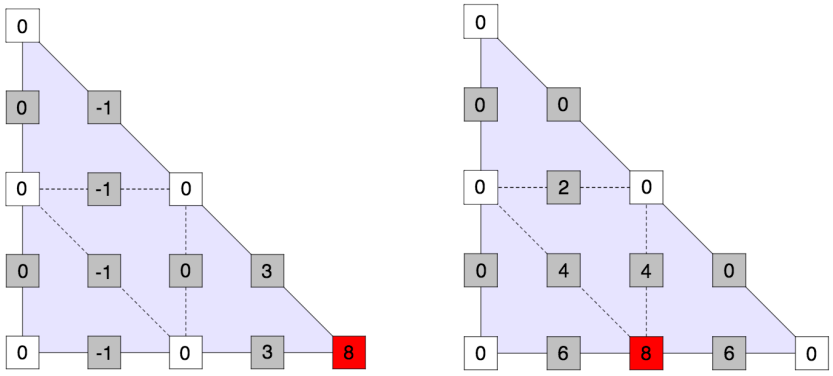
$$\begin{aligned} v_{2h} &= \sum_{j=1}^{n_{2h}} \mathbf{v}_j^{2h} \phi_j^{2h} = \sum_{j=1}^{n_{2h}} \Big(\mathbf{v}_j^{2h} \sum_{i=1}^{n_h} p_{ij} \phi_i^h \Big) = \\ &= \sum_{i=1}^{n_h} \Big(\sum_{j=1}^{n_{2h}} p_{ij} \mathbf{v}_j^{2h} \Big) \phi_i^h = \sum_{i=1}^{n_h} \mathbf{v}_i^h \phi_i^h, \end{aligned} \tag{3.7}$$

where $\mathbf{v}_i^{2h}$ and $\mathbf{v}_i^h$ are the coefficients that represent the function in S_{2h} and S_h respectively. From (3.7) it can be seen that the vector of coefficients $\mathbf{v}^h$ can be expressed as $P\mathbf{v}^{2h}$, where P is the matrix with coefficients p_{ij}. Following from the choice of basis functions in the finite elements setting, the coefficients p_{ij} must be the values of the coarse grid basis functions on the fine grid nodes. In the case of Figure 3.1, the values would be 0, 1/2, 1, 1/2, 0. With the same reasoning, it is easy to understand that the restriction operator I_h^{2h}, that maps S_h into S_{2h}, is described by the matrix $R = P^T$.

As stated in Chapter 1, the FEM discretization chosen for the velocity space is P_2: for this particular choice, the construction of the prolongation matrix P is particularly complex. Indeed, going from the coarse to the fine grid, there

are a lot of nodes which give contribution to the matrix. Figure 3.2 shows the interpolation weights to use to calculate the prolongation matrix (they all have to be divided by 8); the weights depend on the position of the node considered, as it can be a vertex node or a mid-edge node.

Figure 3.2: Weights for P_2 triangular elements (multiplied by 8). In red the reference node; in white the nodes of the coarse grid; in grey the nodes added in the fine grid.



(a) Vertex node

(b) Mid-edge node

Note that this is only one of the possible definitions for prolongation and restriction operators: it is the easiest definition since it arises from the natural inclusion of S_{2h} into S_h, but there are also more complicated ways of defining these operators. However, the natural definition will be the only one used in this work.

In general, the linear system to solve will have the form $A\mathbf{x} = \mathbf{f}$. To use a Multigrid technique, it will be necessary to represent the matrix A and the load vector $\mathbf{f}$ in the coarse grid space. This is easily done exploiting the operators just defined:

$$\mathbf{f}_{2h} = R\mathbf{f}_h = P^T\mathbf{f}_h, \tag{3.8}$$

$$\bar{A} = P^T AP. \tag{3.9}$$

The expression (3.9), called Galerkin coarse grid operator, is only one of the possibilities to represent the matrix A in the coarse grid: indeed, one could just use the exact matrix calculated on the corresponding mesh. In the following, both these techniques will be used, since each of them has advantages and disadvantages, depending on which problem is to be solved.

3.3.2 Smoothing

The second fundamental ingredient of Multigrid is a smoothing operator: a classic choice for this operator is given by a stationary iterative method, like Jacobi or Gauss-Seidel. These methods perform a splitting of the matrix $A = M - N$ and then define the iterates as $M\mathbf{x}^k = N\mathbf{x}^{k-1} + \mathbf{f}$ for k starting from 0. The Jacobi method chooses M as the diagonal of the matrix A, while the Gauss-Seidel method sets M equal to the lower triangular part of A. An alternative version of Jacobi, called damped Jacobi, multiplies the diagonal of A by a parameter θ.

Using these methods, and calling $\mathbf{e}^k$ the error at the k-th iteration, it is easy to see that $\mathbf{e}^k = (I - M^{-1}A)^k\mathbf{e}^0$. It turns out that, after just a couple of iterations of one of these stationary methods, the component of the error vector in the space B_h is almost zero. This means that $\mathbf{e}^k$ can be restricted to the coarse grid with very little loss of information. Of course, in reality it is impossible to work with the error, since the real solution is not known. Instead, the residual must be used: $\mathbf{r}^k = \mathbf{f} - A\mathbf{x}^k = A\mathbf{e}^k$.

The restricted residual will be $\bar{\mathbf{r}} = P^T\mathbf{r}$ and the error will be $\mathbf{e} = P\bar{\mathbf{e}}$. Therefore

$$\bar{\mathbf{r}} = P^T\mathbf{r} = P^TA\mathbf{e} = P^TAP\bar{\mathbf{e}}.$$

This suggests the way in which to proceed: knowing the current approximation $\mathbf{x}$, apply a few iterations of a smoother; compute the residual $\mathbf{r}$ and its restricted version $\bar{\mathbf{r}}$ and then solve the *coarse grid correction* $\bar{\mathbf{r}} = P^TAP\bar{\mathbf{e}}$ to find the vector $\bar{\mathbf{e}}$. This vector then needs to be prolonged and summed to the smoothed approximation, to obtain the next iterate of the Multigrid method. This procedure is summarized in Algorithm 3.3.

Algorithm 3.3 Two-grid cycle

1: Choose $\mathbf{u}_0$
2: **repeat**
3: apply smoother $(I - M^{-1}A)\mathbf{u}_i + M^{-1}\mathbf{f} \to \mathbf{u}_i$
4: restrict residual $\bar{\mathbf{r}} = P^T(\mathbf{f} - A\mathbf{u}_i)$
5: solve coarse grid correction $\bar{\mathbf{r}} = \bar{A}\bar{\mathbf{e}}$
6: prolong error and update $\mathbf{u}_i + P\bar{\mathbf{e}} \to \mathbf{u}_{i+1}$
7: **until** convergence

3.3.3 V and W cycles

The first improvement that can be done to Algorithm 3.3 is to symmetrize it, adding at the end of each iteration another application of the smoother. The two smoothing steps will then be called pre-smoothing and post-smoothing. The next obvious thing to do is to generalize the two grid cycle to a generic number

of grids. This can be done recursively, since the previously formulated algorithm is able to pass from one grid to the next, even amongst a large number of grids.

The easiest implementation of this idea is the V-cycle: starting from the finest grid, the residual is restricted from one grid to the next and the coarse grid correction is solved only when the coarsest grid available is reached. This is summarized in Algorithm 3.4.

Algorithm 3.4 V-cycle

$\mathbf{u}$ =V-cycle $(A, \mathbf{u}_0, \mathbf{f})$
Pre-smooth $(I - M^{-1}A)\mathbf{u} + M^{-1}\mathbf{f} \to \mathbf{u}$
Restrict residual $\bar{\mathbf{r}} = P^T(\mathbf{f} - A\mathbf{u})$
if coarsest level **then**
    Solve coarse grid correction $\bar{\mathbf{r}} = \bar{A}\bar{\mathbf{e}}$
else
    Recursion $\bar{\mathbf{e}}$ =V-cycle $(\bar{A}, \mathbf{0}, \bar{\mathbf{r}})$
end if
Prolong error and correct $\mathbf{u} + P\bar{\mathbf{e}} \to \mathbf{u}$
Post-smooth $(I - M^{-1}A)\mathbf{u} + M^{-1}\mathbf{f} \to \mathbf{u}$

A variation of this algorithm, called W-cycle, can be obtained with a slight modification: when applying the recursion, instead of doing it only one time, it is performed two times. The names for these algorithms are clearly understandable once they are represented in a diagram: suppose that a dot represents a grid and a line represent a prolongation or restriction. Then the V-cycles and W-cycles for a system of 3 grids are drawn in Figure 3.3 and for 4 grids in Figure 3.4; the fine grid is the one on top, while the coarse grid is on the bottom.

Figure 3.3: V and W cycles for 3 grids.

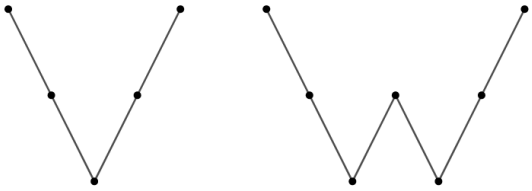
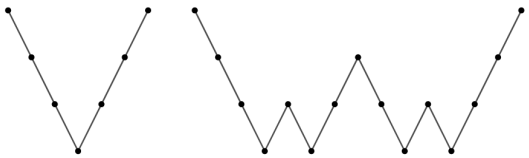


Figure 3.4: V and W cycles for 4 grids.



Intuitively, the W-cycle will be more costly from the computational point of

view, but it will also be more accurate: indeed, this cycle spends definitely more time in the coarser grids, solving the coarse grid correction multiple times.

3.3.4 Properties

We will now prove some important results for Multigrid methods applied to a standard Poisson problem (i.e. a problem which involves only diffusion) with a two-grid cycle. Then, we will discuss how to generalize the method in order to get the same properties for a convection-diffusion problem, like the one that is involved in solving the Navier-Stokes equations. These properties will all rely on some common assumptions: the grids used are quasi-uniform (see Definition 1.1) and are made of shape regular elements (Definition 1.2); moreover, the Poisson problem considered is H^2 regular (Definition 1.3). This last requirement is achieved, for instance, if the boundary of the domain is C^2, i.e. it is locally the graph of a C^2 function.

Let us start by stating some useful results to prove the main theorems: the problem to be solved is of the form $-\Delta u = f$, with $f \in L^2$. $\mathbf{u}$ and $\mathbf{f}$ are the coefficient vectors that represent the function u and f in the finite element space. The following results hold

Lemma 3.1. *If linear finite elements are used to solve a Poisson problem and u is regular enough, then there exists a constant C such that*

$$\|\nabla(u_h - u)\|_{L^2} \leq Ch\|u\|_{H^2}.$$

Due to H^2 regularity, this lemma implies that the L^2 norm of $\nabla(u_h - u)$ can be controlled with the L^2 norm of f.

Lemma 3.2. *The norm of $\mathbf{f}$ can be controlled by the norm of f with a constant C as follows*

$$h\|f\|_{L^2} \leq C\|\mathbf{f}\|.$$

With these two results, it is now possible to state the following important properties.

Theorem 3.4 (Approximation property)**.** *Under the previously cited hypothesis, the approximation property holds:*

$$\|(A^{-1} - P\bar{A}^{-1}P^T)\mathbf{y}\|_A \leq C\|\mathbf{y}\| \qquad \forall \mathbf{y} \in \mathbb{R}^n \tag{3.10}$$

where $\|\cdot\|_A$ is the norm induced by the matrix A and the constant C is independent of the grid size h.

Proof. Consider $u_h \in S_h$ and its coefficient vector $\mathbf{u}$, that satisfies $\mathbf{u} = A^{-1}\mathbf{f}$. The same solution in S_{2h} is u_{2h}; its coefficient vector $\bar{\mathbf{u}}$ satisfies $\bar{\mathbf{u}} = \bar{A}^{-1}\mathbf{f}_{2h} = \bar{A}^{-1}P^T\mathbf{f}$. Now, it can be proved that there is a correspondence between the A-norm and the bilinear form $a(\cdot,\cdot)$: $\|\mathbf{x}\|_A^2 = a(x,x)$, where $\mathbf{x}$ is the coefficient vector of the function x.

Consider (3.10) with $\mathbf{y} = \mathbf{f}$:

$$\|(A^{-1} - P\bar{A}^{-1}P^T)\mathbf{f}\|_A = \|\mathbf{u} - P\bar{\mathbf{u}}\|_A = a(u_h - I_{2h}^h u_{2h}, u_h - I_{2h}^h u_{2h})^{1/2},$$

but $I_{2h}^h u_{2h} = u_{2h}$, so

$$\|(A^{-1} - P\bar{A}^{-1}P^T)\mathbf{f}\|_A = a(u_h - u_{2h}, u_h - u_{2h})^{1/2}.$$

Now, remember that $a(u,u) = \int_\Omega (\nabla u)^2\, d\Omega = \|\nabla u\|^2$; sum and subtract the exact solution u and use the triangular inequality to obtain

$$\|(A^{-1} - P\bar{A}^{-1}P^T)\mathbf{f}\|_A = \|\nabla u_h - \nabla u_{2h}\| \leq \|\nabla(u_h - u)\| + \|\nabla(u - u_{2h})\|.$$

Using Lemma 3.1 and H^2 regularity, we can bound these terms as follows

$$\begin{aligned}\|(A^{-1} - P\bar{A}^{-1}P^T)\mathbf{f}\|_A &\leq \|\nabla(u_h - u)\| + \|\nabla(u - u_{2h})\| \\ &\leq C_1 h\|u\|_{H^2} + C_1 2h\|u\|_{H^2} \\ &\leq 3C_1 h\|u\|_{H^2} \leq 3C_1 hC_2\|f\|.\end{aligned}$$

Finally, we use Lemma 3.2. Hence

$$\|(A^{-1} - P\bar{A}^{-1}P^T)\mathbf{f}\|_A \leq 3C_1 hC_2\|f\| \leq 3C_1C_2C_3\|\mathbf{f}\|,$$

which proves the theorem, since the constant $3C_1C_2C_3$ is independent of h. □

Theorem 3.5 (Smoothing property)**.** *Consider a smoother with $M = \theta I$, $\theta \in \mathbb{R}$ and with the eigenvalues of $I - M^{-1}A$ in the interval $[-\sigma, 1]$, with $0 \leq \sigma < 1$, independently of h. Then the approximation property holds*

$$\|A(I - M^{-1}A)^k\mathbf{y}\| \leq \eta(k)\|\mathbf{y}\|_A \qquad \forall \mathbf{y} \in \mathbb{R}^n \tag{3.11}$$

where $\eta(k) \to 0$ as $k \to \infty$

Proof. Consider the matrix $I - M^{-1}A = I - A/\theta$, which is symmetric, since A is; its orthogonal eigenvectors are denoted by $\mathbf{z}_i$, with corresponding eigenvalues

λ_i. Therefore, every $\mathbf{y}$ can be written as $\mathbf{y} = \sum_i c_i \mathbf{z}_i$. From the definition of $\mathbf{z}_i$, it follows that

$$(I - A/\theta)\mathbf{z}_i = \lambda_i \mathbf{z}_i \implies A\mathbf{z}_i = \theta(1-\lambda_i)\mathbf{z}_i$$

and so

$$\begin{aligned}
\|A(I - M^{-1}A)^k \mathbf{y}\|^2 &= \Big\|\sum_i c_i A(I - A/\theta)^k \mathbf{z}_i\Big\|^2 = \Big\|\sum_i c_i \lambda_i^k A\mathbf{z}_i\Big\|^2 \\
&= \Big\|\sum_i c_i \lambda_i^k \theta(1-\lambda_i)\mathbf{z}_i\Big\|^2 = \sum_i c_i^2 \lambda_i^{2k}\theta^2(1-\lambda_i)^2(\mathbf{z}_i, \mathbf{z}_i) \\
&\leq \max_{\lambda_i \in [-\sigma,1]} \big(\lambda_i^{2k}(1-\lambda_i)\big)\theta \sum_i c_i^2 \theta(1-\lambda_i)(\mathbf{z}_i, \mathbf{z}_i). \qquad (3.12)
\end{aligned}$$

The quantity $\lambda^{2k}(1-\lambda)$ as a function of λ has derivative $2k\lambda^{2k-1} - (2k+1)\lambda^{2k}$ and, in the interval $[-\sigma, 1]$ is maximized either for $\lambda = -\sigma$ or for $\lambda = 2k/(2k+1)$, i.e. the point where the derivative vanishes. For this second value

$$\lambda^{2k}(1-\lambda) = \Big(\frac{2k}{2k+1}\Big)^{2k} \frac{1}{2k+1} = \frac{1}{2k\big(1+\frac{1}{2k}\big)^{2k+1}} \leq \frac{1}{2ke},$$

since for $k \to \infty$, $(1+\frac{1}{2k})^{2k+1} \to e$ monotonically. Hence

$$\max_{\lambda \in [-\sigma,1]} \lambda^{2k}(1-\lambda) \leq \max\Big[\frac{1}{2ke}, \sigma^{2k}(1+\sigma)\Big].$$

Now, recall (3.12), which then becomes

$$\begin{aligned}
\|A(I - M^{-1}A)^k \mathbf{y}\|^2 &\leq \max\Big[\frac{1}{2ke}, \sigma^{2k}(1+\sigma)\Big]\theta \sum_i c_i^2 \theta(1-\lambda_i)\langle \mathbf{z}_i, \mathbf{z}_i\rangle \\
&= \max\Big[\frac{\theta}{2ke}, \theta\sigma^{2k}(1+\sigma)\Big] \sum_i c_i^2 \theta(1-\lambda_i)\langle \mathbf{z}_i, \mathbf{z}_i\rangle \\
&= \max\Big[\frac{\theta}{2ke}, \theta\sigma^{2k}(1+\sigma)\Big] \langle A\mathbf{y}, \mathbf{y}\rangle \\
&= \max\Big[\frac{\theta}{2ke}, \theta\sigma^{2k}(1+\sigma)\Big] \|\mathbf{y}\|_A^2.
\end{aligned}$$

This proves the theorem with $\eta(k) = \big(\max[\frac{\theta}{2ke}, \theta\sigma^{2k}(1+\sigma)]\big)^{1/2}$, which goes to 0 as $k \to \infty$, since $0 \leq \sigma < 1$. □

Theorem 3.5 can be generalized to other smoothers, but the proof is more complicated. For a precise discussion of this theorem, see [21] and [22].

Theorems 3.4 and 3.5 allow us to prove the fundamental result of Multigrid methods, which is the following

Theorem 3.6. *If both the approximation and smoothing property hold, then the two-grid cycle described by Algorithm 3.3 converges and the contraction rate is independent of h.*

Proof. Consider the steps in Algorithm 3.3 and remember that $\mathbf{f} = A\mathbf{u}$, with $\mathbf{u}$ being the exact solution. To study convergence, we are interested in the behavior of the error $\mathbf{e}_{i+1} = \mathbf{u} - \mathbf{u}_{i+1}$. Define as $\mathbf{u}_s$ the result of k smoothing steps on the current iterate $\mathbf{u}_i$. Then, the next iterate is equal to

$$\mathbf{u}_{i+1} = \mathbf{u}_s + P\bar{\mathbf{e}} = \mathbf{u}_s + P\bar{A}^{-1}\bar{\mathbf{r}} = \mathbf{u}_s + P\bar{A}^{-1}P^T(\mathbf{f} - A\mathbf{u}_s). \tag{3.13}$$

Let us try to calculate the term $\mathbf{f} - A\mathbf{u}_s = A\mathbf{u} - A\mathbf{u}_s$ for a generic number of smoothing steps k. Define as $\mathbf{u}_{s-}$ the result of $k-1$ smoothing steps to $\mathbf{u}_i$; then clearly

$$\mathbf{u}_s = (I - M^{-1}A)\mathbf{u}_{s-} + M^{-1}\mathbf{f}.$$

Hence

$$\begin{aligned}
\mathbf{u} - \mathbf{u}_s &= \mathbf{u} - [(I - M^{-1}A)\mathbf{u}_{s-} + M^{-1}A\mathbf{u}] \\
&= \mathbf{u} - (I - M^{-1}A)\mathbf{u}_{s-} - M^{-1}A\mathbf{u} \\
&= (I - M^{-1}A)\mathbf{u} - (I - M^{-1}A)\mathbf{u}_{s-} \\
&= (I - M^{-1}A)(\mathbf{u} - \mathbf{u}_{s-}).
\end{aligned}$$

Iterating this procedure k times yields

$$\mathbf{u} - \mathbf{u}_s = (I - M^{-1}A)^k(\mathbf{u} - \mathbf{u}_i) = (I - M^{-1}A)^k\mathbf{e}_i,$$

$$\mathbf{u}_s = \mathbf{u} - (I - M^{-1}A)^k\mathbf{e}_i.$$

Let us now go back to (3.13)

$$\begin{aligned}
\mathbf{u}_{i+1} &= \mathbf{u}_s + P\bar{A}^{-1}P^TA(\mathbf{u} - \mathbf{u}_s) \\
&= \mathbf{u} - (I - M^{-1}A)^k\mathbf{e}_i + P\bar{A}^{-1}P^TA(I - M^{-1}A)^k\mathbf{e}_i \\
&= \mathbf{u} - (I - P\bar{A}^{-1}P^TA)(I - M^{-1}A)^k\mathbf{e}_i.
\end{aligned}$$

So, it is now possible to calculate the error of the next iterate

$$\begin{aligned}
\mathbf{e}_{i+1} = \mathbf{u} - \mathbf{u}_{i+1} &= (I - P\bar{A}^{-1}P^TA)(I - M^{-1}A)^k\mathbf{e}_i \\
&= (A^{-1} - P\bar{A}^{-1}P^T)A(I - M^{-1}A)^k\mathbf{e}_i.
\end{aligned} \tag{3.14}$$

Equation (3.14) shows that the error can be expressed using the matrices involved in the Theorems 3.4 and 3.5. Therefore, exploiting the approximation

and smoothing properties

$$\begin{aligned}\|\mathbf{e}_{i+1}\|_A &= \|(A^{-1} - P\bar{A}^{-1}P^T)A(I - M^{-1}A)^k\mathbf{e}_i\|_A \\ &\leq C\|A(I - M^{-1}A)^k\mathbf{e}_i\|_A \\ &\leq C\eta(k)\|\mathbf{e}_i\|_A.\end{aligned}$$

Since $\eta(k) \to 0$ as $k \to \infty$, there exists a minimal number of smoothing steps $\bar{k}$ such that $\|\mathbf{e}_{i+1}\|_A \leq \gamma\|\mathbf{e}_i\|_A$, where the factor γ is smaller than one and independent of h, meaning that the method converges and with a contraction rate independent of h. □

This important result relies on the hypothesis that the eigenvalues of the smoothing matrix are bounded away from -1 independently of h. It is known that stationary methods converge if and only if the spectral radius of the iteration matrix is less that 1, but this is not enough to satisfy the hypothesis of Theorem 3.5; indeed, one eigenvalue could tend to -1 while always remaining in modulus smaller than 1. To avoid this phenomenon, the smoother must be tuned carefully: in particular, for damped Jacobi smoothing (i.e. a stationary method where M is the diagonal of A multiplied by θ), the optimal parameter θ is $9/8$.

If we manage to choose the correct parameters for the smoother, then we obtain a method to solve or precondition a linear system that will always converge in the same number of iterations, regardlessly of the dimension of the problem. This means that solving the same problem in a finer mesh, will produce the same number of iterations and therefore the growth of the computational time will be linear with respect to the number of nodes of the mesh. This property of a numerical method is referrer to as *scalability*.

The results derived in this Section for the simple two-grid cycle are still valid also for the more complex V and W cycles, but the proofs are more difficult and technical.

3.3.5 Multigrid for convection-diffusion problems

All the results derived in the previous Sections, although they can be generalized to a generic V or W cycle and with other smoothers, hold in the case of a Poisson problem. The process of solving (or preconditioning) the Navier-Stokes equations involves solving linear systems related to the discretization of a convection-diffusion operator. This fact changes the way in which the Multigrid method should be used, in order to obtain the desired scalability.

When solving a Stokes problem, there is no convection involved, hence the Multigrid is indeed applied only to a diffusion problem and it can be used

as described up to this point. However, when it comes to the Navier-Stokes equations, there are some major changes to make. These modifications involve essentially three features of the method: the coarse grid approximation, the smoother and the cycle to use. Let us now discuss how all these aspects need to be treated.

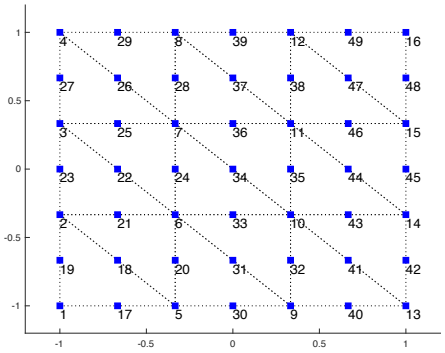
Concerning the coarse grid approximation, we should avoid the use of the Galerkin coarse grid operator: indeed, we will start from a very fine grid, for which the mesh Péclet number is sufficiently small to avoid the use of a stabilization technique, like streamline diffusion; but, when we will get to the coarser grids, their Pe will be much larger and will require some stabilization. Using the Galerkin coarse grid operator to generate the matrices for all the grids does not add any stabilization terms and this will lead to a very unstable problem in the coarse grids. This means that the coarse grid correction will be solved in a poor way, leading to poor results also in the fine grids. The best way to proceed is to directly compute the matrices for all the grids, i.e. calculate the matrices using the finite elements routine and not approximate them with the Galerkin operator. In this way, it is possible to add the stabilization term for the meshes that have a large Péclet number.

The next fundamental problem to address is the choice of the smoother; let us first review some basics about the iterative schemes used as smoothers. Both Jacobi and Gauss-Seidel methods start from an initial vector and progressively update the components in order to get closer to the exact solution of the linear system. The main difference is that for Jacobi-like methods, every component is updated considering only the old values of the approximation. Gauss-Seidel, instead, updates the components following an order and for each one considers also the new values already computed. So, the first component will be updated considering all the components of the old approximation. The second component will take into account also the new value of the first component and so on. The k-th component will then have $k-1$ newly calculated components to exploit in order to update the value.

This implies that the Gauss-Seidel method is more accurate; moreover, while Jacobi updates the components "all at the same time", Gauss-Seidel updates them following a specific pattern. This means also that the properties of the smoother can vary according to which pattern is used. Indeed, for convection dominated problems, some patterns work better than others, and some simply do not work at all; in particular, the smoothing scheme should follow the convection process, meaning that the components of the solution should be updated in an order that follows the direction of the main flow.

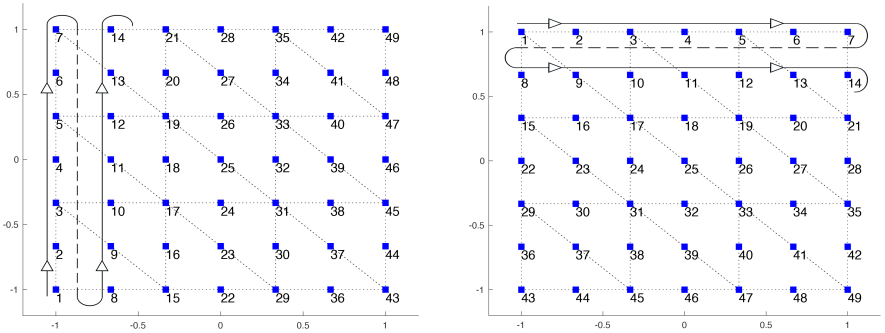
In a problem with a simple structure, this is easy to achieve: e.g. if the flow flows unidirectionally from the left to the right of the domain, then the smoother

Figure 3.5: Standard numeration of the nodes.



should update first the nodes on the left and then proceed towards the right. In more complicated problems, where the flow recirculates, like the lid-driven cavity problem that is considered in this work, there are two options, as suggested in [8]: build a pattern that follows the circulating flow (which is extremely complicated to achieve), or apply multiple simple patterns in succession, in order to capture in the best possible way the features of the convective flow. This means that every smoothing step will consist of multiple Gauss-Seidel iterations, each one sweeping through the domain in a different direction. This approach has already been tested, for instance in [17].

Figure 3.6: Ordering of the nodes for the Gauss-Seidel smoothing.



(a) First iteration

(b) Second iteration

To better understand how this works, consider Figure 3.5 and Figure 3.6: the first one shows the node ordering that arises from the natural construction of the mesh; Gauss-Seidel applied using this pattern will be referred to as Simple-Gauss-Seidel and will definitely not follow the direction of the flow. Figure 3.6 instead shows two patterns, one sweeping through the nodes from left to right

and one from top to bottom. This approach tries to follows the flow, but is inevitably more computationally expensive. We could also perform Gauss-Seidel with four directions of sweeping, making the method more accurate but even more time consuming. It is important to remember that, in order to maintain the symmetry in a Multigrid solver or preconditioner, the various sweeping of Gauss-Seidel in the pre-smoothing must be applied in the opposite order in the post-smoothing.

The last feature of Multigrid to modify is the cycle: it has been noticed that using the W-cycle instead of V-cycle produces a method that is more stable and performs better, even though it requires more computational time. This was already noticed in [14], where it was applied to a different kind of discretization.

To summarize, the important features to modify in the standard Multigrid method in order to use it for convection-diffusion problems are the following:

- calculate directly all the matrices involved, using stabilization when needed, and not use the Galerkin approach;
- use a smoother that follows the flow, like Gauss-Seidel with proper ordering, and not Jacobi-like methods;
- use a W-cycle instead of a V-cycle.

Chapter 4

Preconditioning techniques

We will now describe the preconditioner that have been used in this work. We will show some of its properties and how it can be implemented. Then, we will go into the details of how to precondition the various blocks of the system and in particular, how the Schur complement preconditioner changes passing from the Stokes equations to the Navier-Stokes equations.

4.1 Constraint preconditioner

The preconditioner that we used is known as *constraint preconditioner*. This names comes from the fact the this particular preconditioner keeps in its structure both the blocks B^T and B, which in optimization are related to the constraint of the problem; saddle point systems indeed appear in numerous applications, including constrained optimization. This particular preconditioner can be used in all these applications, since it does not exploit any underlying structure of the blocks.

The constraint preconditioner P takes the form

$$P = \begin{bmatrix} P_A & B^T \\ B & 0 \end{bmatrix}, \tag{4.1}$$

where P_A is a preconditioner for the $(1,1)$ block A. The application of this preconditioner will require its inversion, which can be done in various ways, as we will see later. For the moment, we focus on evaluating the eigenvalues of the preconditioned matrix $P^{-1}M$, where M is the matrix of the saddle point system in (2.1). Indeed, we discovered that the number of different eigenvalues, or clusters of eigenvalues, is fundamental to determine the speed of convergence of a method like GMRES. At the moment, we will discuss this preconditioner

under the assumptions that A and P_A are symmetric and nonsingular and that B has full rank.

4.1.1 Eigenvalues distribution

Let us now try to determine the eigenvalues of the preconditioned matrix, i.e. the numbers λ and the vectors that satisfy

$$\begin{bmatrix} P_A & B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix}.$$

Instead of solving this problem, which requires the inverse of P, we will study the following generalized eigenvalue problem, which has the same eigenvalues:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} P_A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4.2}$$

Let us start by taking the QR factorization of the matrix B^T

$$B^T = \begin{bmatrix} Y & Z \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $Y \in \mathbb{R}^{n\times m}$, $Z \in \mathbb{R}^{n\times(n-m)}$, $R \in \mathbb{R}^{m\times m}$ and the zero block has dimensions $(n-m)\times m$. The next proposition will be useful in the following steps:

Proposition 4.1. *The columns of Z form a basis for the null space of B.*

Proof. Consider a generic orthogonal matrix $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$. Due to its orthogonality, $\mathrm{im}(Q_1) = \mathrm{im}(Q_2)^\perp$.

The matrix B can be written as

$$B = \begin{bmatrix} R^T & 0^T \end{bmatrix} \begin{bmatrix} Y^T \\ Z^T \end{bmatrix}.$$

A generic vector $y \in \ker(B)$ then satisfies

$$By = 0 \implies R^T Y^T y = 0 \implies Y^T y = 0.$$

The last implication comes from the fact that R has full rank, since B has full rank.

Therefore y is in the kernel of Y^T, which is also the orthogonal complement to the range of Y. But $\mathrm{im}(Y)^\perp = \mathrm{im}(Z)$, for what has been said at the beginning of the proof. Therefore, y belongs to the range of Z and so $\ker(B) = \mathrm{im}(Z)$, which proves the thesis. □

Let us now pre-multiply the left-hand side of equation (4.2) by the matrix

$$\begin{bmatrix} Z^T & 0 \\ Y^T & 0 \\ 0 & I \end{bmatrix} \tag{4.3}$$

which is nonsingular since the first two blocks form an orthogonal matrix; the identity matrix is of dimension $m \times m$ to get the correct dimensions. We then obtain

$$\begin{bmatrix} Z^T A & Z^T B^T \\ Y^T A & Y^T B^T \\ B & 0 \end{bmatrix}.$$

Now, we make the following change of variable

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} Z & Y & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_z \\ x_y \\ y \end{bmatrix},$$

so that $x = Zx_x + Yx_y$. In this way, the matrix in the left-hand side of (4.2) gets post-multiplied by the transpose of matrix (4.3), to obtain

$$\begin{bmatrix} Z^T AZ & Z^T AY & Z^T B^T \\ Y^T AZ & Y^T AY & Y^T B^T \\ BZ & BY & 0 \end{bmatrix}.$$

Exploiting Proposition 4.1, we can say that $BZ = 0$. Moreover, from the QR decomposition of B^T, it follows that $B^T = YR$ and so, since Y has orthogonal columns, $R = Y^T B^T$. With this in mind, the previous matrix can be simplified. The same procedure can be carried out for the right-hand side of equation (4.2), yielding the following problem

$$\begin{bmatrix} Z^T AZ & Z^T AY & 0 \\ Y^T AZ & Y^T AY & R \\ 0 & R^T & 0 \end{bmatrix} \begin{bmatrix} x_z \\ x_y \\ y \end{bmatrix} = \lambda \begin{bmatrix} Z^T P_A Z & Z^T P_A Y & 0 \\ Y^T P_A Z & Y^T P_A Y & R \\ 0 & R^T & 0 \end{bmatrix} \begin{bmatrix} x_z \\ x_y \\ y \end{bmatrix}.$$

Exchanging row and column blocks, the previous matrices can be put into triangular form

$$\begin{bmatrix} R^T & 0 & 0 \\ Z^T AY & Z^T AZ & 0 \\ Y^T AY & Y^T AZ & R \end{bmatrix} \begin{bmatrix} x_y \\ x_z \\ y \end{bmatrix} = \lambda \begin{bmatrix} R^T & 0 & 0 \\ Z^T P_A Y & Z^T P_A Z & 0 \\ Y^T P_A Y & Y^T P_A Z & R \end{bmatrix} \begin{bmatrix} x_y \\ x_z \\ y \end{bmatrix}.$$

Therefore, the preconditioned matrix $P^{-1}M$, is similar to the following matrix, up to a permutation of rows and columns

$$\begin{bmatrix} I & 0 & 0 \\ N & (Z^T P_A Z)^{-1}(Z^T A Z) & 0 \\ S & T & I \end{bmatrix}. \tag{4.4}$$

The structure of matrices N, S and T is not relevant, since we are only interested in finding the eigenvalues of this matrix. The three diagonal blocks are of dimensions $m \times m$, $(n-m) \times (n-m)$ and $m \times m$ respectively. Therefore, we can say that the preconditioned matrix has $2m$ eigenvalues equal to 1 and $n-m$ eigenvalues given by the eigenvalues of the matrix $(Z^T P_A Z)^{-1}(Z^T A Z)$. We summarize the results of these last pages in the following

Theorem 4.1. *Consider a symmetric nonsingular matrix $A \in \mathbb{R}^{n\times n}$ and a matrix $B \in \mathbb{R}^{m\times n}$ of full rank which form a saddle point system. Assume $Z \in \mathbb{R}^{n\times(n-m)}$ is a basis for the kernel of B. If a preconditioner like* (4.1) *is used, with P_A symmetric, then the preconditioned matrix has a unit eigenvalue, with multiplicity $2m$, and $n-m$ eigenvalues defined by the generalized eigenvalue problem $Z^T A Z x_z = \lambda Z^T P_A Z x_z$.*

We can thus appreciate how the constraint preconditioner is able to cluster $2m$ eigenvalues at 1. If the matrix B does not have full rank, like in the driven cavity problem that we will solve, then in the QR decomposition the matrix R will have the last rows made of zeros, while matrix Z will have larger dimension. The rest of the reasoning remains the same though, so a large part of the eigenvalues will still be clustered at 1.

We now want to understand more about the $n-m$ eigenvalues that are not 1. In order to do so, let us suppose that the preconditioner P_A is positive definite. In this case we can use the following

Theorem 4.2 (Cauchy's interlace theorem)**.** *Consider $T \in \mathbb{R}^{n\times n}$ symmetric and $H \in \mathbb{R}^{m\times m}$, with $m < n$ and H a principal submatrix of T. Consider the eigenvalues of these matrices labeled as*

$$Tz_i = \alpha_i z_i, \quad i = 1, \dots, n, \quad \alpha_1 \le \alpha_2 \le \cdots \le \alpha_n,$$

$$Hy_i = \lambda_i y_i, \quad i = 1, \dots, m, \quad \lambda_1 \le \lambda_2 \le \cdots \le \lambda_m.$$

Then

$$\alpha_k \le \lambda_k \le \alpha_{k+n-m}, \quad k = 1, \dots, m.$$

Now, recall that $Q = \begin{bmatrix} Y & Z \end{bmatrix}$ is an orthogonal matrix and consider the following two eigenvalue problems

$$Q^T A Q v = \alpha Q^T P_A Q v, \tag{4.5}$$

$$Z^T A Z w = \lambda Z^T P_A Z w. \tag{4.6}$$

Since P_A is positive definite, also $Q^T P_A Q$ is, and so we can write its Cholesky decomposition

$$Q^T P_A Q = \begin{bmatrix} Z^T P_A Z & Z^T P_A Y \\ Y^T P_A Z & Y^T P_A Y \end{bmatrix} = \begin{bmatrix} L & 0 \\ R & S \end{bmatrix} \begin{bmatrix} L^T & R^T \\ 0 & S^T \end{bmatrix} = N N^T.$$

It is easy to check that $LL^T = Z^T P_A Z$, $R = Y^T P_A Z L^{-T}$ and $SS^T = Y^T P_A Y - RR^T$. We can therefore rewrite problems (4.5) and (4.6) as

$$Q^T A Q v = \alpha N N^T v, \qquad Z^T A Z w = \lambda L L^T w.$$

We use the change of variable $u = N^T v$ and $z = L^T w$, to obtain

$$N^{-1} Q^T A Q N^{-T} u = \alpha u, \qquad L^{-1} Z^T A Z L^{-T} z = \lambda z.$$

If we pre-multiply the matrix $N^{-1} Q^T A Q N^{-T}$ by N^{-T} and post-multiply by N^T, we obtain $P_A^{-1} Q^T A Q$, so the two are similar; moreover, $Q^T A Q$ is trivially similar to A, hence $N^{-1} Q^T A Q N^{-T}$ is similar to $P_A^{-1} A$ and so the α_i are also the eigenvalues of $P_A^{-1} A$. In the same way, $L^{-1} Z^T A Z L^{-T}$ is similar to $(Z^T P_A Z)^{-1}(Z^T A Z)$ and therefore the λ_i are exactly the non unitary eigenvalues of the preconditioned saddle point system. We can now use Theorem 4.2 to say that $\alpha_k \leq \lambda_k \leq \alpha_{k+m}$, $k = 1, \ldots, n - m$. In turn, this means that the eigenvalues λ_i are all bounded by the extremal eigenvalues of $P_A^{-1} A$; so, if P_A is a good preconditioner for A, then the λ_i will be clustered and the constraint preconditioner will produce a preconditioned matrix with eigenvalues very well clustered around 1.

These results hold under some strong hypothesis, which will not be verified for the system arising from the Navier-Stokes equations, that is not even symmetric; however, these properties are useful to show how powerful the constraint preconditioner can be in the best case scenario.

Also the distribution of eigenvectors can be analyzed in a similar way, as it is done in [12]; the important result to notice is that, in general, the number of linearly independent eigenvectors of the preconditioned saddle point linear system is not $n + m$ and thus the matrix is not diagonalizable. This implies

that Theorem 3.3 does not hold and so at this point we cannot be sure that the clustering of eigenvalues corresponds to a rapid convergence of the GMRES method.

Luckily, GMRES still converges in a number of iterations related to the number of eigenvalues, as stated in the following

Theorem 4.3. *Consider a symmetric nonsingular matrix $A \in \mathbb{R}^{n\times n}$ and a matrix $B \in \mathbb{R}^{m\times n}$ of full rank that form a saddle point system M. Assume $Z \in \mathbb{R}^{n\times(n-m)}$ is a basis for the kernel of B. Suppose a preconditioner P like (4.1) is used, with P_A symmetric. If the matrix $(Z^T P_A Z)^{-1}(Z^T A Z)$ has k distinct eigenvalues λ_i, with multiplicity μ_i, then the Krylov subspace $K_d(P^{-1}M, b)$ has dimension at most $k+2$ for any b.*

Proof. (sketch, the full proof is available in [12])

The dimension of a Krylov subspace is bounded by the degree of the minimal polynomial of the vector b, which in turn is bounded by the degree of the minimal polynomial of the matrix of the system. Let us start by writing the characteristic polynomial of $P^{-1}M$, which is easily derived from Theorem 4.1:

$$(P-I)^{2m}\prod_{i=1}^{k}(P-\lambda_i I)^{\mu_i}.$$

This polynomial has degree $n+m$. Let us now consider the polynomial of degree $k+1$

$$(P-I)\prod_{i=1}^{k}(P-\lambda_i I).$$

If we substitute the structure of the preconditioner from (4.4) and we expand all the terms, we get to a matrix with only the $(3,1)$ block different from zero. Multiplying again by $(P-I)$ yields the zero matrix, suggesting that the minimal polynomial, in the worst case, is

$$(P-I)^2\prod_{i=1}^{k}(P-\lambda_i I),$$

which has degree $k+2$. □

Since we know that GMRES terminates when the Krylov subspace reaches its maximum dimension, we can be sure that the number of iterations required to converge will be at most $k+2$. This result is almost as powerful as the one in Theorem 3.3, since in that case convergence would occur in $k+1$ iterations, only one less.

4.1.2 Implementation

We will now present different techniques that allow us to efficiently implement the constraint preconditioner. The first one is known as *Exact Constraint Preconditioner* (ECP) and it consists in applying the preconditioner exactly as it is presented at the beginning of this Chapter. In Algorithm 3.2, we can see that, at some point, we will need to take the inverse of the preconditioner or, in other words, to solve a system like $Py = r$:

$$\begin{bmatrix} P_A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \tag{4.7}$$

which corresponds to the equations

$$\begin{cases} P_A y_1 + B^T y_2 = r_1 \\ B y_1 = r_2 \end{cases}.$$

The first equation yields $y_1 = P_A^{-1}(r_1 - B^T y_2)$, that can be substituted into the second equation to give $BP_A^{-1}(r_1 - B^T y_2) = r_2$. It is thus possible to find y_2, solving the system

$$BP_A^{-1}B^T y_2 = BP_A^{-1}r_1 - r_2 \tag{4.8}$$

and then y_1 comes from the system

$$P_A y_1 = r_1 - B^T y_2. \tag{4.9}$$

To find the solution, it is now fundamental to solve these two linear systems, which have as matrices P_A and $P_S = BP_A^{-1}B^T$, the Schur complement of the preconditioner P. The techniques that we will present all differ in how to solve these two linear systems, and in particular in how system (4.8) is solved.

The first possibility that we considered for the Stokes problem, was to use as P_A the incomplete Cholesky factorization of matrix A; this is not possible for the Navier-Stokes problem, since the matrix in no longer symmetric. Thus $P_A = LL^T$ and the application of P_A^{-1} is very cheap, since it is obtained through the solution of two triangular linear systems. In [4], it is explained that the preconditioned conjugate gradient (PCG) can be used successfully even if the system is indefinite, provided that the initial residual has a particular structure. Therefore, we decided to use the PCG; moreover, we chose to use the PCG also to solve system (4.8), since the Schur complement is symmetric positive semi-definite. Therefore, we have two PCG cycles: each one needs a routine to apply the matrix of the system and a routine to apply the preconditioner. For the outer PCG, we used the exact constraint preconditioner just described;

for the inner PCG cycle, we applied the matrix simply by multiplying by all its constitutive blocks ($P_S = BL^{-T}L^{-1}B^T$) and we used as preconditioner the mass matrix Q. Indeed, Theorem 2.1 assures us that the mass matrix is a good preconditioner for the Schur complement, since all the eigenvalues are well clustered in a narrow interval.

The ECP technique therefore has a complicated structure, since it has nested PCG iterations and it requires multiple external routines to apply the matrices and the preconditioners. The advantage is that this procedure solves exactly the problem and therefore achieves all the theoretical properties that have been derived in the previous Section. However, the exact solution of system (4.8) at every iteration is in general very expensive and affects badly the performance of the whole method.

For the exact constraint preconditioner, we tried to apply Theorem 2.3: we calculated the exact structure of the preconditioned matrix and tried to find the vectors in the kernel and in the range. The computations are shown in Appendix A.

Another possibility is to approximate P_S in order to solve system (4.8) quickly, for instance by means of an incomplete Cholesky factorization; this approach is known as *Inexact Constraint Preconditioner* (ICP). Clearly, the eigenvalues distribution that we found for the ECP does not hold anymore in the case of the ICP, but still the preconditioner is robust and performs much better than a standard preconditioner based on incomplete factorization of the whole system. A particular case of inexact constraint preconditioner is obtained considering two different preconditioners for the $(1,1)$ block, P_A and $\tilde{P}_A$: the first is used to solve system (4.9), while the second one is used to build the Schur complement preconditioner, i.e. $P_S = B\tilde{P}_A^{-1}B^T$. This technique goes under the name of *Mixed Constraint Preconditioner* (MCP).

In this work we will mainly use an inexact constraint preconditioner, so let us try to understand its properties. As we did with the saddle point matrix, also the preconditioner can be factorized in the following way

$$P = \begin{bmatrix} I & 0 \\ BP_A^{-1} & I \end{bmatrix} \begin{bmatrix} P_A & 0 \\ 0 & -P_S \end{bmatrix} \begin{bmatrix} I & -P_A^{-1}B^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} P_A & B^T \\ B & BP_A^{-1}B^T - P_S \end{bmatrix}.$$

If we use an exact constraint preconditioner, then $P_S = BP_A^{-1}B^T$ and we recover preconditioner (4.1); if instead P_S is approximated, then the $2,2-$block of the preconditioner is no longer zero, but it is equal to $BP_A^{-1}B^T - P_S$. This preconditioner can be applied in the same way described before for the ECP, with the difference that now the matrix for the system (4.8) is different.

To summarize, suppose that to apply the preconditioner we need to solve a

system like (4.7), using an approximation of the Schur complement P_S, then the main steps are shown in Algorithm 4.1.

Algorithm 4.1 Application of ICP

1: Compute x: $P_A x = r_1$
2: Compute f: $f = Bx - r_2$
3: Compute y_2: $P_S y_2 = f$
4: Compute g: $g = r_1 - B^T y_2$
5: Compute y_1: $P_A y_1 = g$.

We would now like to understand the distribution of the eigenvalues of the inexact constraint preconditioner. To do so, we will continue to suppose that A is symmetric and we will also consider P_A and P_S to be symmetric positive definite. Then, we can define the following preconditioned matrices

$$A_P = P_A^{-1/2} A P_A^{-1/2}, \qquad S_P = P_S^{-1/2} B A^{-1} B^T P_S^{-1/2}.$$

We will denote their eigenvalues as follows: α_A and α_S will be the minimum eigenvalues in modulus of A_P and S_P, while β_A and β_S will be their maximum eigenvalues in modulus.

Let us now make the following assumptions

$$0 < \alpha_A < 1 < \beta_A,$$
$$0 < \alpha_S < 1 < \beta_S.$$

These conditions are often fulfilled; to see this, let us first notice that A_P has the same eigenvalues as $P_A^{-1} A$: indeed, the eigenvalues and eigenvectors of A_P are defined as

$$P_A^{-1/2} A P_A^{-1/2} v = \lambda v.$$

If we pre-multiply by $P_A^{-1/2}$, we get

$$P_A^{-1} A P_A^{-1/2} v = \lambda P_A^{-1/2} v,$$

and changing the variable to $w = P_A^{-1/2} v$, we obtain

$$P_A^{-1} A w = \lambda w.$$

Therefore, $P_A^{-1} A$ and A_P share the same eigenvalues but not the same eigenvectors. Thus, if P_A is a good preconditioner for A, meaning that is clusters the eigenvalues around 1, then the same holds for the eigenvalues of A_P. Hence, it is reasonable to assume that α_A and β_A lay on different sides of unity. The same holds for S_P.

We will also make the further assumption that $\beta_A < 2$; then, the following theorem holds

Theorem 4.4. *Assuming the previously stated hypothesis, the real eigenvalues of an ICP preconditioned saddle point system satisfy*

$$\min\Big\{\alpha_A, \frac{\alpha_S}{\beta_A}\Big\} \leq \lambda \leq \max\{\beta_A, (2-\alpha_A)\beta_S\}.$$

The complex eigenvalues $\lambda = \lambda_R + i\lambda_I$ satisfy

$$\frac{\alpha_A + \alpha_S(2-\beta_A)}{2} \leq \lambda_R \leq \frac{\beta_A + \beta_S(2-\alpha_A)}{2},$$

$$|\lambda_I| \leq \sqrt{\beta_S}\max\{1-\alpha_A, \beta_A - 1\}.$$

The proof of this theorem can be found in [3], together with some numerical tests confirming that these bounds are tight, especially for the real eigenvalues. This result is a generalization of what we found previously: indeed, we discovered that for ECP, the non unitary eigenvalues are confined in the interval $[\alpha_A, \beta_A]$; in this case this interval is possibly enlarged by the presence of some other terms.

In any case, if P_A and P_S are good preconditioners for A and $S = BA^{-1}B^T$, then Theorem 4.4 assures us that the eigenvalues of the preconditioned matrix will be clustered around 1. The important question now it how to find a scalable preconditioner: recall that a preconditioner is scalable if, as the mesh is refined, the computational cost to solve the system grows linearly. This means that the eigenvalues and the condition number of the matrix should not change much from one matrix to the next, so that the only added computational cost is given by the larger dimension of the problem. Thus, the eigenvalues of the preconditioned matrix need to be uniformly bounded, i.e. they should not grow too much or tend to zero as the mesh is refined. If we are able to find scalable preconditioners for the $(1,1)$ block and the Schur complement, then α_A, β_A, α_S and β_S will not depend on the mesh size h and so, due to Theorem 4.4, also the eigenvalues of the preconditioned saddle point system will be bounded independently of h, meaning that the ICP will be scalable.

4.1.3 Relaxation

The inexact constraint preconditioner can be accelerated by means of a relaxation parameter ω, as it is shown in [5]; the new preconditioner obtained is built in

this way

$$P(\omega) = \begin{bmatrix} I & 0 \\ BP_A^{-1} & I \end{bmatrix} \begin{bmatrix} P_A & 0 \\ 0 & -\omega P_S \end{bmatrix} \begin{bmatrix} I & -P_A^{-1}B^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} P_A & B^T \\ B & BP_A^{-1}B^T - \omega P_S \end{bmatrix}.$$

For some values of ω, this preconditioner will produce better convergence rates than the standard inexact constraint preconditioner. Indeed, the following theorem holds

Theorem 4.5. *Under the same hypothesis of Theorem 4.4, the real eigenvalues of $P(\omega)^{-1}M$ satisfy*

$$\min\Big\{\alpha_A, \frac{\omega\alpha_S}{2}\Big\} \leq \lambda \leq \max\{\beta_A, 2\omega\beta_S\}.$$

The complex eigenvalues $\lambda = \lambda_R + i\lambda_I$ satisfy

$$\frac{\alpha_A}{2} \leq \lambda_R \leq \beta_A + \frac{\omega\beta_S}{2}, \qquad |\lambda_I| \leq \sqrt{\omega\beta_S}\max\{1, \beta_A - 1\}.$$

The proof of this theorem can be found in [5], where it is also noticed that the speed of convergence of this preconditioner is influenced by the ratio between the largest and smallest real eigenvalue of the preconditioned matrix. Therefore, the optimal value of ω is the one which minimizes this ratio; calling $R_{\max}$ and $R_{\min}$ the largest and smallest real eigenvalues of the preconditioned matrix, their ratio can be estimated according to Theorem 4.5:

$$\kappa = \frac{R_{\max}}{R_{\min}} \leq \frac{\max\{\beta_A, 2\omega\beta_S\}}{\min\{\alpha_A, \frac{1}{2}\omega\alpha_S\}}.$$

It is computationally cheap to calculate the largest eigenvalues, but it is very expensive to compute the smallest ones; therefore, the next result gives an approximation of the optimal ω based only on the knowledge of the largest eigenvalues.

Theorem 4.6. *Let $\omega = \frac{\beta_A}{\beta_S}$, then the ratio between the largest and smallest eigenvalue of the preconditioned matrix is bounded by*

$$\frac{R_{max}}{R_{min}} \leq \max\{2c_A, 4c_S\},$$

where $c_A = \frac{\beta_A}{\alpha_A}$ and $c_S = \frac{\beta_S}{\alpha_S}$ are the condition numbers of A_P and S_P respectively.

Proof. Considering the results of the previous theorems, the following bounds

hold

$$R_{\max} \leq \max\{\beta_A, 2\omega\beta_S\} = \max\{\beta_A, 2\beta_A\} = 2\beta_A,$$

$$R_{\min} \geq \min\Big\{\alpha_A, \frac{\omega\alpha_S}{2}\Big\} = \min\Big\{\alpha_A, \frac{\alpha_S\beta_A}{2\beta_S}\Big\} = \min\Big\{\alpha_A, \frac{\beta_A}{2c_S}\Big\}.$$

Therefore

$$\frac{R_{\max}}{R_{\min}} \leq \max\Big\{\frac{2\beta_A}{\alpha_A}, 2\beta_A\frac{2c_S}{\beta_A}\Big\} = \max\{2c_A, 4c_S\}.$$

□

Therefore, without having to compute the smallest eigenvalues, we are able to obtain a value of ω for which the value of κ is close to the optimal one. This specific value $\omega = \frac{\beta_A}{\beta_S}$ has a particular interpretation: the matrix ωS_P has the largest eigenvalue equal to $\omega\beta_S = \beta_A$, which is the spectral radius of A_P. Then, this relaxation technique can be seen as a shift in the spectral interval of S_P, in order to match the spectral radii of these two preconditioned blocks. This method can produce an actual acceleration in the convergence rate only if there is a significant shift between the two spectra of A_P and S_P, and will instead have no effect if one spectrum is fully contained in the other or if the two largest eigenvalues are very close. In some cases, the relaxation can also have negative effects, since it can push some eigenvalues closer to zero, worsening the condition number of the problem.

4.2 Other preconditioners

Besides the constraint preconditioner, many other preconditioners for saddle point systems have been developed. We will now present a few of them, showing some of their basic properties.

In [8, p. 195], a *block diagonal preconditioner* was used to solve the Stokes problem; its structure is the following

$$P = \begin{bmatrix} P_A & 0 \\ 0 & T \end{bmatrix},$$

where P_A is a preconditioner for the $(1,1)$ block and T is an approximation of the Schur complement. In some particular cases, the preconditioned matrix has only three eigenvalues $\lambda = 1, 1/2 \pm \sqrt{5}/2$. With some more practical choices of the matrices involved, the eigenvalues lay in the union of three narrow intervals, with bounds that are independent of the mesh size. This means that the convergence will happen fast and the number of iterations will not grow as the problem becomes larger. However, for the Navier-Stokes problem, the choice of a block diagonal preconditioner is not particularly good; keeping in the structure one

or both the blocks B, B^T yields better clustering of the eigenvalues and thus a better performance of the solver.

In [20], a *triangular block preconditioner* (TBP) was used to solve the Navier-Stokes problem and in [3] some eigenvalue bounds were developed; the structure is

$$P = \begin{bmatrix} P_A & 0 \\ 0 & -P_S \end{bmatrix} \begin{bmatrix} I & P_A^{-1}B^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} P_A & B^T \\ 0 & -P_S \end{bmatrix},$$

which is derived from the constraint preconditioner previously presented. The real eigenvalues then satisfy

$$\min\Big\{\alpha_A, \frac{\alpha_S}{\beta_S + \alpha_S}\Big\} \leq \lambda \leq \beta_S + \beta_A.$$

These bounds are similar to the ones for the inexact constraint preconditioner, in the sense that they still depend only on the spectral properties of the preconditioners used for the $(1,1)$ block and the Schur complement. Computationally, its application is slightly cheaper that the one of ICP: Algorithm 4.2 indeed shows that we need to solve one system with matrix P_A, one with matrix P_S and perform a matrix-vector product. In Algorithm 4.1 instead, we need to solve two systems with matrix P_A, one with matrix P_S and perform two matrix-vector products. This means that surely the computational time per iteration of GMRES with TBP will be smaller than the one with ICP; however, TBP is likely to produce a larger number of iterations, since it embeds less information about the matrix in its structure. It is impossible to tell a priori which one of these effects will prevail and so which of the two preconditioners will perform better. In the next Chapter, we present the results of a comparison between ICP and TBP.

Algorithm 4.2 Application of TBP

1: Compute y_2: $P_S y_2 = -r_2$
2: Compute g: $g = r_1 - B^T y_2$
3: Compute y_1: $P_A y_1 = g$.

A completely different approach is given in [2] and [9]: the preconditioner used is essentially the same triangular block preconditioner as before, only that now the linear system solved is different. The original system is replaced with

$$\begin{bmatrix} A + \gamma B^T W^{-1} B & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix},$$

where W is usually chosen to be equal to Q and γ is a constant with value close to the intensity of the wind. This system has the same solution as the original one, since in the $(1,1)$ block we have added something that vanishes if multiplied by u. The advantage in using this *augmented Lagrangian* approach is that the

Schur complement preconditioner to be used with any block preconditioner is extremely simple: indeed, the Schur complement now takes the form

$$S = B(A + \gamma B^T Q^{-1} B)^{-1} B^T,$$

which can be simplified using the matrix inversion lemma. The final result is

$$S^{-1} = (BA^{-1}B^T)^{-1} + \gamma Q^{-1},$$

and therefore, since the scaled mass matrix is a preconditioner for the term $BA^{-1}B^T$, then a good preconditioner for the Schur complement of the augmented system is

$$\tilde{S}^{-1} = (\nu + \gamma) Q^{-1},$$

which can even become a diagonal matrix if Q is replaced by a diagonal approximation. Thus, there is no need to find a suitable approximation for the Schur complement, which is, as we will see later, one of the main challenges when preconditioning the Navier-Stokes equations. The price to pay is that the preconditioner for the $(1,1)$ block is an extremely complicated Multigrid scheme, very different from what has been presented in the previous Chapter; this is because the $(1,1)$ block now contains a term with a large null space, since matrix B has a large kernel.

Moreover, as analyzed in [9], this preconditioner works well only for some particular discretization techniques, which may need to be developed precisely for this task and its extension to three dimensional problems is even more complicated.

4.3 Preconditioners for the blocks

Previously, we showed that a block preconditioner for a saddle point linear system, like the inexact constraint preconditioner, is only as good as the preconditioners used to approximate the $(1,1)$ block and the Schur complement. Therefore, we now try to find scalable preconditioners for these two blocks.

Let us start from the $(1,1)$ block: we have seen, with Theorem 3.6, that the Multigrid method is able to solve a linear system involving a Poisson problem in a number of iterations that does not depend on the mesh size; we have then tried to generalize this property for a convection-diffusion problem. The question now might be how to use such a technique as a preconditioner: indeed, we are not able to write down explicitly the matrix associated to a Multigrid scheme, but it turns out that to apply a preconditioner we only need the action of its inverse. This means that we do not have to know P_A^{-1} in explicit form, but we just need

to have a routine that applies P_A^{-1} to a vector. This is easily done with the algorithms that were presented when talking about V and W cycles. Therefore, when in Algorithm 4.1 we need to solve a linear system with matrix P_A, we will just employ a Multigrid scheme, without actually knowing the matrix.

The big problem is to choose the right Multigrid scheme to use, since only some of them are scalable. For the Stokes problem, it is sufficient to use the easiest version of Multigrid, since the $(1,1)$ block represents a standard Poisson problem; we will then use a damped Jacobi smoother, we will use the Galerkin coarse grid operator and we will apply a V-cycle. For the Navier-Stokes problem, the $(1,1)$ block represents a convection-diffusion operator and so we will need to take this into account; hence, we will employ a Gauss-Seidel smoother and we will try different possibilities for the ordering of the nodes. We will calculate all the matrices involved exactly, without using the Galerkin approach, and we will use a W-cycle. We hope that all these changes will produce a scalable preconditioner, but there are no theoretical results like the ones for the Poisson problem, so we will not be sure about the behavior of the preconditioner until we actually test it. For very low viscosities, it is possible that even the more complicated Multigrid scheme that we can produce will not be sufficient to assure scalability.

4.3.1 Schur complement preconditioner

Let us now focus in finding a scalable preconditioner for the Schur complement, P_S. We will start from the Stokes problem. In this case, we can exploit a result that we proved previously: Theorem 2.1 states the fact that the matrix $Q^{-1}BA^{-1}B^T$ has eigenvalues that are bounded independently of the mesh size; moreover the upper bound is 1, so the spectrum will be clustered close to 1. This condition obviously holds as long as the inf-sup constant β is different from zero, which is the case since our choice of the finite element spaces is stable.

Therefore, using $P_S = Q$ is a choice that assures scalability; we then expect that a constraint preconditioner using a Multigrid scheme for the $(1,1)$ block and the mass matrix Q for the Schur complement, applied to the Stokes problem, will be scalable. In Section 5.2 we checked this condition, both with the eigenvalues distribution and with the iterations count. The mass matrix is actually such a good preconditioner for the Schur complement, that the relaxation has no effect at all, since the two spectra are very tight.

Solving a linear system with the mass matrix is very cheap, since in most cases it has a particular structure (tridiagonal, pentadiagonal or similar) and it is symmetric positive definite: this means that we can either solve directly the system with the backslash operator of Matlab, which is extremely efficient in

exploiting such particular structures, or we can perform a Cholesky factorization. Another possibility is to substitute the full mass matrix Q with a diagonal approximation D, i.e. a diagonal matrix that approximates, in some sense, the mass matrix. The easiest choice is to use exactly the diagonal of Q, but there are more sophisticated techniques like the mass lumping. In [8, p. 199], it is proved that choosing for D the diagonal of Q produces a preconditioner for the Schur complement that is spectrally equivalent to the mass matrix. Thus, we can use this D to simplify the computations even more, but without losing the scalability of the preconditioner. As it turns out, this method does not yield better results, as it is shown again in Section 5.2.

The question now is how to find a suitable P_S for the Navier-Stokes problem. The first guess would be to use again the mass matrix, but unfortunately this method is not scalable anymore. For high viscosities (i.e. for problems close to the Stokes one), the results are still acceptable, but for lower viscosities Q does not represent anymore a viable choice. The first improvement that can be done is to use the scaled mass matrix, i.e. the mass matrix divided by the viscosity: this improves the results, but still it is not a scalable preconditioner, as it is shown in Section 5.3. We then need to find different strategies to precondition the Schur complement.

Let us start with an intuitive explanation of why the mass matrix works so well for the Stokes problem. The Schur complement $BA^{-1}B^T$ represent the discrete version of the action of three operators: a gradient, given by B, the inverse of a Laplacian, given by A^{-1}, and a divergence, given by B^T. We can then think that the Schur complement is related to the operator $(\operatorname{div}\Delta^{-1}\nabla)$; recalling that Δ acts as $(\operatorname{div}\nabla)$, it is intuitively clear that the Schur complement is equivalent to an identity operator. The mass matrix, following from its definition, represents a discrete identity operator and so it is the perfect candidate as preconditioner for the Schur complement.

In the Navier-Stokes problem, however, the matrix A is no longer a Laplace operator, but it represents a convection-diffusion operator and hence this reasoning does not hold anymore. Nonetheless, we can proceed with a slightly different reasoning, to produce a suitable preconditioner for the Schur complement in the Navier-Stokes case. In [11], all the technical details of the derivation of this preconditioner are carried out exactly, using Green's functions and Fourier transforms; here we will just give the intuitive interpretation. The main idea is given by two points: firstly, if the convective processes tend to zero (and so the problem tends to a Stokes problem), the preconditioner should default to the mass matrix, which is the optimal preconditioner in the Stokes case; secondly, we should build the preconditioner considering that A now represents an operator of the kind $-\nu\Delta + \mathbf{w}\cdot\nabla$, where $\mathbf{w}$ is the local wind, that in the

Oseen problem is given by the solution of the previous iteration. Therefore, the Schur complement is now equivalent to the operator $\mathrm{div}\,(-\nu\Delta + \mathbf{w}\cdot\nabla)^{-1}\nabla$. Its inverse can be viewed as the application of $-\nu\Delta + \mathbf{w}\cdot\nabla$ together with the inverse of both the divergence and the gradient; we can think of these last two operators as the action of the inverse of a Laplacian. Thus, an approximation to the inverse of the Schur complement is given by the operator $(-\nu\Delta + \mathbf{w}\cdot\nabla)\Delta^{-1}$. In matrix notation, this is written as $A_p L_p^{-1}$, where A_p has the same structure of matrix A, but it is build using the basis functions of the pressure space, and L_p is the Laplacian matrix built on the pressure space. We need to consider the pressure version of these matrices in order to match the dimensions to those of the Schur complement. $A_p L_p^{-1}$ does not default to Q^{-1} as $\mathbf{w} \to 0$, so we just pre-multiply by Q^{-1}. The final preconditioner is then $P_S^{-1} = Q^{-1} A_p L_p^{-1}$. Since this preconditioner uses a convection-diffusion operator built on the pressure space, it is usually called *pressure convection-diffusion* (PCD) preconditioner.

PCD has been successfully tested in a wide range of situations and it showed mesh-independent convergence rates for moderate values of ν; moreover, the dependence of the convergence rate on ν is improved in comparison to the scaled mass matrix. However, when the viscosity gets low, the convergence rates can deteriorate even for the simplest flows. The matrices involved A_p and L_p are not readily available from the finite elements discretization, but they must be calculated separately; moreover, they need the proper boundary conditions for the preconditioner to work at its best. The issue of choosing the right boundary conditions for these matrices is not yet well understood, since for many situations they have to be different from the boundary conditions prescribed on the underlying differential problem, they may depend on the viscosity and on the type of convective flow and they may even differ on different portions of the boundary. In [15], a theorem is proved stating that the eigenvalues of the PCD preconditioned Schur complement are uniformly bounded.

Let us now derive another Schur complement preconditioner, originally proposed in [7]. This formulation starts from the following simple result of linear algebra.

Proposition 4.2. *Consider two matrices G and K of dimensions $m \times n$, $n > m$, of full rank. The matrix*

$$K^T (G K^T)^{-1} G \tag{4.10}$$

represents an identity operator over range(K^T).

Proof. Take a vector $x \in \mathrm{range}(K^T)$; it can be written as $x = K^T y$, for some

vector $y \in \mathbb{R}^m$. The application of $K^T(GK^T)^{-1}G$ to x yields

$$\begin{aligned} K^T(GK^T)^{-1}Gx &= K^T(GK^T)^{-1}GK^Ty \\ &= K^Ty = x. \end{aligned}$$

Therefore, $K^T(GK^T)^{-1}G$ is an operator that maps range(K^T) into itself and moreover it is the identity operator. □

To apply this result, assume that B has full rank and consider $G = BA^{-1}$ and $K = B$. They are both of dimension $m \times n$ and so we can apply the previous proposition. The operator (4.10) then becomes $B^T(BA^{-1}B^T)^{-1}BA^{-1}$ and we can say that

$$B^T(BA^{-1}B^T)^{-1}BA^{-1} = I \qquad \text{on range}(B^T),$$

which means that

$$B^T(BA^{-1}B^T)^{-1}BA^{-1}x = x \quad \forall x \in \text{range}(B^T).$$

Equivalently, considering a vector $y = A^{-1}x$, we can say that

$$B^T(BA^{-1}B^T)^{-1}By = Ay \quad \forall y \in \text{range}(A^{-1}B^T). \tag{4.11}$$

Let us now make the following assumption

$$\text{range}(B^T) \subset \text{range}(A^{-1}B^T). \tag{4.12}$$

Consider a vector $u \in \text{range}(B^T)$; due to assumption (4.12), u is also in range$(A^{-1}B^T)$ and so we can use equation (4.11) with $y = u$:

$$B^T(BA^{-1}B^T)^{-1}Bu = Au \quad \forall u \in \text{range}(B^T).$$

Since u is in the range of B^T, we can write it as $u = B^Tv$, for some vector $v \in \mathbb{R}^m$

$$B^T(BA^{-1}B^T)^{-1}BB^Tv = AB^Tv \quad \forall v \in \mathbb{R}^m.$$

We can now pre-multiply this expression by B, to get

$$BB^T(BA^{-1}B^T)^{-1}BB^Tv = BAB^Tv \quad \forall v \in \mathbb{R}^m,$$

or equivalently

$$(BA^{-1}B^T)^{-1}BB^Tv = (BB^T)^{-1}(BAB^T)v \quad \forall v \in \mathbb{R}^m.$$

We can now change variable to $w = BB^T v$ and obtain

$$(BA^{-1}B^T)^{-1}w = (BB^T)^{-1}(BAB^T)(BB^T)^{-1}w \quad \forall w \in \mathbb{R}^m.$$

This expression implies that $(BB^T)^{-1}(BAB^T)(BB^T)^{-1}$ and $(BA^{-1}B^T)^{-1}$ are the same matrix: indeed, if the expression holds for every vector w, in particular it holds for the canonical basis. Using the $k-$th vector of the canonical basis yields the $k-$th column of the matrix, which means that every column of the first matrix is equal to the corresponding column of the second matrix. Thus, the two matrices are equal.

Therefore, we have found an expression for the inverse of the Schur complement that does not involve the inverse of matrix A. We can then say that

$$P_S^{-1} = (BB^T)^{-1}BAB^T(BB^T)^{-1} \tag{4.13}$$

is a good preconditioner for the Schur complement. In the original formulation, matrix A was called F and so this method took the name of *BFBt preconditioner*, since if involved the term BFB^T.

Since this preconditioner represents the exact inverse of the Schur complement, it would produce a preconditioned matrix with all eigenvalues equal to 1. In practice, it is not possible to invert exactly the matrix BB^T and moreover assumption (4.12) does not hold perfectly. In [7], there is an example of a problem for which the assumption hold, but for generic scenarios with generic boundary conditions this will not be the case.

Preconditioner (4.13) can be modified in a simple way to improve its performance; the final form of the BFBt preconditioner is

$$P_S^{-1} = (BQ_v^{-1}B^T)^{-1}BQ_v^{-1}AQ_v^{-1}B^T(BQ_v^{-1}B^T)^{-1}, \tag{4.14}$$

where Q_v is either the exact velocity mass matrix or a diagonal approximation.

The application of (4.14) requires the solution of two systems with matrix $BQ_v^{-1}B^T$; this can be done either with an inexact factorization or reformulating this term using the pressure Laplacian matrix. Besides this, the application requires three matrix-vector products and the solution of two linear systems involving Q_v, which are easy to solve. Therefore, it seems that the computational cost of a single application of this preconditioner is higher in comparison to the PCD preconditioner. However, there are some advantages: this technique is build using only the matrices that are already present in the original problem; for simple flows, the robustness with respect to ν is improved and mesh size independence is observed; there is no more the necessity to impose the proper boundary conditions on the matrices, which was a difficult problem to deal

with. Unfortunately, for complicated flows, like the one that we will use, this preconditioner shows some dependence on ν and on the mesh size h.

In Section 5.3, we present the results of the use of the BFBt preconditioner, together with various Multigrid schemes, for the Navier-Stokes problem. The results shown are again related to the spectral properties and the number of iterations of GMRES. We chose to use the BFBt preconditioner in our application and not the PCD, so that we did not have to deal with the unknown boundary conditions to be applied in the latter. Moreover, we wanted to compare the BFBt preconditioner to another preconditioner that is derived directly from it and that we are now going to introduce.

The BFBt preconditioner can be thought, neglecting the mass matrix scaling, as the discrete version of the operator

$$\Delta_p^{-1}\mathrm{div}\,(-\nu\Delta + \mathbf{w}\cdot\nabla)\nabla\Delta_p^{-1},$$

where we have indicated with Δ_p the operator related to a pressure Laplace problem. In [15], some critical issues of this approach are underlined: they derive mainly from the difference between the boundary conditions that such an operator is able to apply, in comparison to the boundary conditions required for the underlying problem. The suggested solution to this problem is to commute the operators ∇ and div with Δ_p^{-1}; after adjusting the operators so that the dimensions match, the result is

$$\mathrm{div}\,\Delta^{-1}(-\nu\Delta + \mathbf{w}\cdot\nabla)\Delta^{-1}\nabla,$$

where this time we have used the standard velocity Laplace problem. The preconditioner that comes out of this operator is given by

$$P_S^{-1} = Q^{-1}BL^{-1}AL^{-1}B^TQ^{-1}, \tag{4.15}$$

where the matrix L is the Laplacian matrix, that corresponds to the matrix A used in the Stokes problem. We have also introduced a scaling given by Q^{-1}, as was done in the case of the BFBt preconditioner.

The commutation that was performed is formally correct only for some particular cases with special boundary conditions; however, arguments based on inexact commutators are often used in the derivation of both PCD and BFBt preconditioners. Thus, we do not expect it to be a problem.

Let us see some of the advantages of this formulation:

1. Preconditioner (4.15) is built using the matrices already available from the finite element routines; indeed, matrix L corresponds to the diffusive part

of A.

2. There is no more the issue of inverting the matrix BB^T: this could have been done using a Multigrid scheme, but it would have required a different technique, since this matrix acts on the pressure space, which has linear basis functions and therefore the prolongation and restriction operators would have been different from the ones used in the main Multigrid routine. Moreover, such a method requires to impose some boundary conditions on the pressure Laplacian and again we have the problem of which boundary conditions to apply.

3. The action of L^{-1} instead can be performed using the same Multigrid that we use for matrix A, since it is an operator that works on the velocity space.

4. None of the matrices involved require the application of specially developed boundary conditions, which was a problem with the PCD preconditioner.

5. For a wider set of cases, this method showed convergence independent of the mesh size and of the viscosity, as shown by the results in [15].

On the other hand, there are still some critical points:

1. For complicated flows, like the one that we will investigate, some dependence on ν was observed.

2. This preconditioner is not particularly effective for diffusion dominated flows; indeed, for the limit case in which A becomes exactly L (which is the Stokes problem), then the preconditioned Schur complement becomes just $(Q^{-1}S)^2$. We already know that in this case the optimal preconditioner is Q^{-1} and this means that this approach produces a preconditioned matrix with a condition number that is the square of what we would get in the optimal case. This in turn means that the number of iterations will nearly double.

3. Compared with the BFBt preconditioner, we still have to perform three matrix-vector products and solve a couple of systems involving Q; however, now we have to solve two systems with matrix L, that compared to BB^T has a much larger dimension.

For preconditioner (4.15), the following theorem holds

Theorem 4.7. *Consider a quasi uniform discretization and suppose that the bilinear forms involved in the finite elements approximation satisfy the assumptions*

of Theorem 1.1. Then, the eigenvalues λ *of the Schur complement, preconditioned using* (4.15)*, satisfy*

$$c \leq \lambda \leq C,$$

where c *and* C *are positive constants independent of the mesh size* h*.*

The proof of this theorem is extremely technical and can be found in [15]. Notice that the constants c and C could still depend on other parameters, like the viscosity. However, we hope that this result will allow us to obtain a scalable preconditioner for the Schur complement.

Since this preconditioner is obtained from the BFBt one, by means of a commutation, it will be referred to as *BFBt-c preconditioner*. The results of its application are shown in Section 5.3.

In Appendix B we report the various algorithms that have been used to apply the constraint preconditioner.

Chapter 5

Numerical results

In this last Chapter, we will show the results of the numerical experiments performed; the spectral properties of the matrices will be used to predict the behavior of a certain technique and we will then compare the predictions with the actual results. Some plots will be used to compare the spectra of different preconditioners and to show convergence profiles of GMRES.

5.1 Problem description

The discretization chosen for the finite elements is of second order for the velocity and first order for the pressure. Therefore, calling n the number of intervals on each side of the squared domain, the number of degrees of freedom for the pressure variables is $(n+1)^2$, while for the velocity we must take into account also the midpoints of the edges of the triangles; moreover, for every node, there are two unknowns, since the velocity is a vectorial quantity. Therefore, the number of degrees of freedom for the velocity is $2(2n+1)^2$. Making the sum, the total variables are $9n^2+10n+3$. Considering a refinement of the mesh with a number of intervals $2n$, the number of variables becomes $36n^2+20n+3$. It is important to notice that the ratio between the number of degrees of freedom in these two cases approaches 4 from below as n increases.

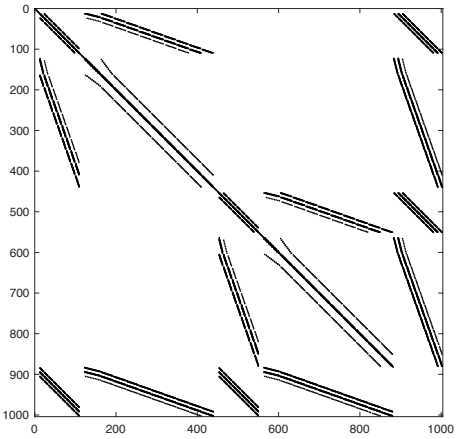
The matrices used for the numerical tests are generated using values of n from 10 to 320. In particular, Table 5.1 illustrates all the matrices used, showing their dimension and number of nonzero entries, while Figure 5.1 shows the sparsity pattern the matrix M10.

It can be seen that the number of nonzero entries per row is varying between 14 and 18 and the density increases as n grows. The ratio of the dimensions goes from 3.8 for the small matrices to 3.99 for the bigger ones, which agrees with what has been said before. The smaller matrices M10 and M20 will be used

Table 5.1: Properties of the matrices

Matrix	Dimension	NNZ
M10	1 003	14 280
M20	3 803	62 008
M40	14 803	258 268
M80	58 403	1 054 076
M160	232 003	4 258 388
M320	924 803	17 118 310

Figure 5.1: Sparsity pattern of matrix M10



only in the layer of the Multigrid preconditioner, while the other will be actually used as matrices for the saddle point system.

Concerning the Navier-Stokes equations, the matrices used are obtained after five iterations of a Picard scheme, which ensures that the nonlinear phenomena are well represented. The value of the viscosity is set to 1 for the Stokes problem. For the Navier-Stokes equations, it will assume the values 10^{-1}, 10^{-2}, $5 \cdot 10^{-3}$ and 10^{-3}. Lowering the viscosity, the system will become more asymmetric, which is expected to worsen the properties of the preconditioner.

The numerical results that will be presented will regard firstly the Stokes problem, that will asses if the basic Multigrid scheme works as expected. Then, different strategies to precondition the Navier-Stokes equations will be investigated, analyzing both the spectral properties and the number of iterations required for the GMRES to converge. The notation will be the following: α_A and β_A represent the maximum and minimum eigenvalues in modulus of the preconditioned $(1,1)$ block; α_S and β_S represent the maximum and minimum eigenvalues in modulus of the preconditioned Schur complement (obviously, the zero eigenvalue of the Schur complement is not considered, since it is al-

ways present); ω^* represents the optimal value of the relaxation parameter, as formulated in Theorem 4.6.

Some details on implementation

All the results have been obtained using codes written in Matlab. The finite elements have been implemented using a four-point Gaussian quadrature scheme. The eigenvalue estimates are obtained using the Arnoldi method. The solver used is left-preconditioned GMRES with a tolerance of 10^{-10} and without restart.

The preconditioner used is the inexact constraint preconditioner (ICP), unless otherwise specified, sometimes used with relaxation. The coarsest mesh for the Multigrid schemes is always the one corresponding to the matrix M10. So, if we are solving the problem with matrix M160, the Multigrid will have 5 levels, relative to the matrices M160, M80, M40, M20, M10, and the system will be solved exactly only in the coarsest level. To solve this small system, we used the backslash command of Matlab.

We noticed that an enormous quantity of time was lost to extract the proper matrices in the smoothing phase (diagonal or triangular parts); so, we decided to extract all these matrices in advance, store them and rewrite the smoothers so that they choose the correct matrix to use. This requires some preprocessing time, which varies from some hundredths of a second (M40) to 2-3 seconds (M320). Anyway, this time grows linearly with respect to the dimension of the matrix, so it will never represent a problem.

All the times reported are in seconds.

5.2 Stokes problem

The Stokes problem is symmetric, so we used a simple Multigrid V-cycle with one iteration of pre and post smoothing, using a damped Jacobi smoother. The preconditioner for the Schur complement is either the mass matrix (Q) or its diagonal (diagQ). These two preconditioners for the $(1,1)$ block and for the Schur complement have been analyzed and studied, even if using a different block preconditioner, in [8, pp. 189-211].

The spectral properties of the preconditioned matrices are reported in Table 5.2, together with the optimal value of the parameter ω for both cases.

Table 5.2: Spectral properties of the preconditioned matrices for the Stokes problem.

Matrix	α_A	β_A	α_S(Q)	β_S(Q)	β_S(diagQ)	$\omega^*(Q)$	ω^*(diagQ)
M40	0.8606	1.0002	0.0968	0.9921	1.7290	1.0082	0.5809
M80	0.8496	1.0000	0.1333	0.9941	1.7690	1.0060	0.5652

It is clear that the Multigrid works as expected, since the eigenvalues are clustered around 1, in a tight interval. Moreover, the eigenvalues do not degrade refining the mesh. The same holds for the Schur complement preconditioner, in both cases. Therefore we expect a number of iterations almost constant and a computational time that grows linearly as the mesh is refined. This is confirmed by Table 5.3, which also shows that the mass matrix is a better preconditioner than its diagonal.

Table 5.3: Results for the Stokes problem.

Matrix	Q		diagQ	
	iterations	time	iterations	time
M40	31	5.57	46	8.31
M80	33	18.79	47	29.37
M160	34	93.29	48	148.60

Since both the Multigrid preconditioner and the Schur complement preconditioner work very well, we expect not to gain anything using the relaxation parameter. Indeed, Table 5.4 proves this fact.

Table 5.4: Results for the Stokes problem with relaxation.

Matrix	ω	iter with Q	iter with diagQ
M40	1	31	46
	1.0082	31	–
	0.5809	–	46
M80	1	33	47
	1.0060	33	–
	0.5652	–	47
M160	1	34	48
	1.01	34	–
	0.55	–	47

We solved the Stokes problem also using the exact constraint preconditioner (ECP) presented in Algorithm B.1, in order to evaluate its performance relative to the inexact constraint preconditioner. The results with ECP are shown in Table 5.5.

This preconditioner is clearly not scalable; for the matrix M40, ECP is much faster that ICP, but then the number of iterations grows rapidly and the computational time consequently becomes much grater than the one required to the ICP. Therefore, it is clear that ICP is superior to ECP.

Now that we have assessed that the preconditioners behave as expected in the well known Stokes problem, we can try to solve the Navier-Stokes equations.

Table 5.5: Results for the Stokes problem using ECP, with inner tolerance 10^{-6}, outer tolerance 10^{-10}, droptol 10^{-3}.

Matrix	iter	time
M20	7	0.08
M40	40	1.52
M80	97	28.19
M160	>200	>423

5.3 Navier-Stokes problem

The Navier-Stokes equations involve a system that is no more symmetric, due to the presence of the convection. We therefore expect the Jacobi smoother to behave badly, leading to a non-scalable preconditioner. Moreover, the mass matrix is no longer a viable preconditioner for the Schur complement. Table 5.6 shows the eigenvalues of the Schur complement preconditioned with the scaled mass matrix (mass matrix divided by the viscosity) for a low viscosity (0.005). It is clear that all the nice properties that the mass matrix had in the Stokes problem are lost: the spectral radius increases as the mesh refines and the eigenvalues are no longer well clustered around one.

Table 5.6: Spectral properties of the Schur complement, preconditioned with the scaled mass matrix, for the Navier-Stokes problem with $\nu = 0.005$.

Matrix	α_S	β_S
M40	0.3753	12.1965
M80	0.2423	16.8846
M160	0.2431	22.9679

It is therefore fundamental to find different preconditioners. We tested three Multigrid schemes, using Jacobi and Gauss-Seidel smoothers, and two preconditioners for the Schur complement, namely the BFBt preconditioner and the BFBt-c preconditioner. We used Algorithms B.3 and B.4. The following Sections present the various combinations of the two preconditioners.

5.3.1 Jacobi Multigrid / BFBt

The combination of Multigrid with damped Jacobi smoothing and BFBt to solve the Navier-Stokes equations will be referred to as NS-J-BFBt. The Multigrid is executed using 5 iterations of a V-cycle with one pre and post smoothing step; the BFBt preconditioner is implemented using an ILU factorization with threshold 10^{-5}. This combination has not been studied a lot since, as it will be clear, it is not a good choice. The Multigrid is the same as for the Stokes

problem, while the Schur complement preconditioner has been analyzed in [7].

The spectral properties of the preconditioned matrices are reported in Table 5.7. The value ω^* is approximately 0.05.

Table 5.7: Spectral properties of the preconditioned matrices for the NS-J-BFBt problem.

Matrix	Viscosity	α_A	β_A	α_S	β_S
M40	0.1	0.4256	1.261	0.6926	25.86
	0.01	0.4251	1.905	0.4746	21.87
M80	0.1	0.3555	1.442	0.2107	26.62
	0.01	0.3588	2.911	0.1494	25.92
M160	0.1	0.3371	1.700	0.0557	39.41
	0.01	0.3383	5.013	0.1744	28.97

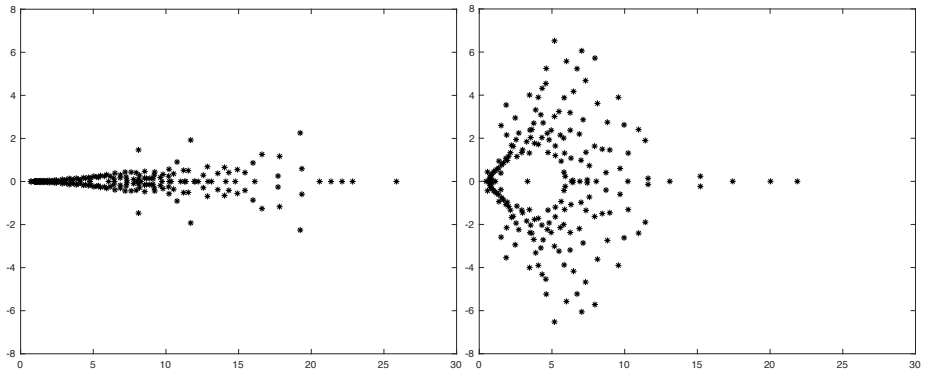
The eigenvalues of the preconditioned $(1,1)$ block show a slight dependence on the mesh size (α_A is decreasing, β_A is increasing) even for the highest viscosity, which represents a problem similar to the Stokes one. The smallest eigenvalue α_A does not seem to be affected by the value of the viscosity, while the biggest one β_A grows substantially with the viscosity, getting in the worst case to a value larger than 5. The fact that the spectral properties deteriorate as the viscosity grows is perfectly in line with the fact that this Multigrid preconditioner was built for a symmetric problem, while lowering the viscosity makes the problem more and more asymmetric. For an even lower viscosity (0.005), the eigenvalues become negative, leading to a complete failure of the preconditioner.

The situation is not better for the Schur complement preconditioner: even in this case, the spectral interval widens as the mesh is refined. Moreover, the eigenvalues are less clustered around 1, as the spectral radius is always above 20.

Figure 5.2 illustrates the spectrum of the preconditioned Schur complement for various values of the viscosity. Every dot represents an eigenvalue in the complex plane (only the first 100 eigenvalues found by Arnoldi are represented). These spectra do not depend on the Multigrid scheme used, so they would be the same even using a different Multigrid. It is clear that, as the viscosity gets lower, the system becomes more asymmetric, and as a consequence the imaginary part of the eigenvalues becomes bigger. Moreover, as ν decreases, most of the eigenvalues tend to have a smaller real part; this leads to a better clustering of the eigenvalues around 1 when the viscosity is low.

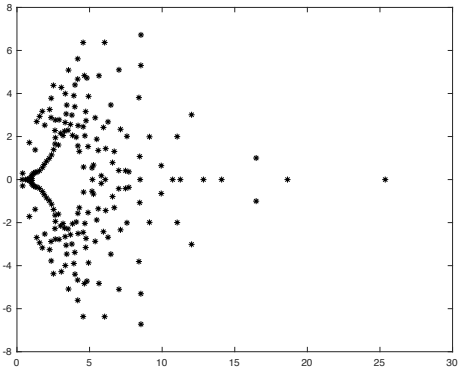
The poor spectral properties lead us to think that the NS-J-BFBt preconditioner will not be scalable. The evident difference in the spectrum of the $(1,1)$ block and of the Schur complement hints that the relaxation may produce a benefit; however, since the spectra have very different width, its effect will not

Figure 5.2: Spectra of the BFBt-preconditioned Schur complement for different values of viscosity.



(a) $\nu = 0.1$

(b) $\nu = 0.01$



(c) $\nu = 0.005$

be too significant.

Table 5.8 reports the results obtained with this preconditioners, in terms of iterations and computational time for various values of the relaxation parameter ω between 0.05 and 1.

The results are not surprising: the preconditioner is not scalable, indeed the number of iterations grows vary rapidly as the mesh is refined. The iterations count increases also as the viscosity gets lower, showing a lack of robustness in this sense. The relaxation produces some small benefits, however the optimal ω appears not to coincide with the estimated ω^*: this is due to the fact that the system is no longer symmetric and hence the theoretical results previously developed may not hold anymore in the same way.

Using a lower viscosity of 0.005, the matrix M40 fails to converge within 300 iterations, emphasizing the lack of robustness of the Jacobi-BFBt combination. In order to improve the performance of the preconditioner, we implemented a Multigrid scheme using simple-Gauss-Seidel as smoother, keeping the same

Table 5.8: NS-J-BFBt results for various values of the relaxation parameter.

Matrix	ω	$\nu = 0.1$		$\nu = 0.01$	
		iter	time	iter	time
M40	1	67	4.10	110	6.89
	1/2	62	3.82	107	6.55
	1/4	61	3.73	105	6.41
	1/10	59	3.63	107	6.64
	1/20	60	3.67	127	7.85
M80	1	97	14.01	195	29.34
	1/2	88	12.58	183	27.95
	1/4	86	12.26	180	26.99
	1/10	83	11.84	177	26.97
	1/20	81	11.52	184	27.83
M160	1	141	79.23	336	233.8
	1/2	135	75.10	332	233.7
	1/4	133	74.92	329	228.0
	1/10	130	72.56	324	223.6
	1/20	128	71.48	325	225.0

Schur complement preconditioner.

5.3.2 Simple-GS Multigrid / BFBt

The combination of Multigrid with simple-Gauss-Seidel and BFBt preconditioners will be referred to as NS-sGS-BFBt. All the details of the implementation of the preconditioner (number of V-cycle...) remain the same as before.

Simple-Gauss-Seidel smoothing performs the Gauss-Seidel method keeping the original numeration of the nodes, hence without following the convection flow. This is expected to produce better results than the Jacobi Multigrid preconditioner, but still it should not be scalable as it does not follow the direction of the flow. Table 5.9 shows the spectral properties of the preconditioned $(1,1)$ block and Schur complement.

There is a great improvement in the eigenvalue bounds of the preconditioned $(1,1)$ block for the highest viscosity: indeed the spectral interval is extremely narrow and focused around 1. However, this nice property disappears for the lowest viscosity, leading even to negative eigenvalues for the matrix M80. That is exactly what we expected: for high viscosities, the convection phenomena are not relevant, hence there is not much difference between smoothers that follow or not the flow. Instead, for low viscosities, the convection process must be taken into account to produce a suitable smoother.

It can be noticed that the eigenvalue bounds for the preconditioned Schur

Table 5.9: Spectral properties of the preconditioned matrices for the NS-sGS-BFBt problem.

Matrix	Viscosity	α_A	β_A	α_S	β_S
M40	0.1	1.0000	1.002	0.6966	25.44
	0.01	0.9784	1.006	0.4527	21.83
	0.005	0.6591	1.422	0.2522	25.15
M80	0.1	1.0000	1.003	0.2159	26.47
	0.01	0.9984	1.015	0.1290	25.82
	0.005	<0	19.35	0.0349	27.95

complement are slightly different from the previous ones: indeed, in order to apply the Arnoldi method to the Schur complement, there is the necessity to use a Multigrid scheme to produce an approximation of the inverse of the $(1,1)$ block. These differences however are very small, since a change in the Multigrid affects the eigenvalue estimates of the Schur complement far less than the eigenvalues of the $(1,1)$ block.

Table 5.10 shows the results obtained for some values of the relaxation parameter.

Table 5.10: NS-sGS-BFBt results for various values of the relaxation parameter.

Matrix	ω	$\nu = 0.1$		$\nu = 0.01$	
		iter	time	iter	time
M40	1	64	3.90	107	6.75
	1/2	62	3.84	105	6.68
	1/4	61	3.80	104	6.57
M80	1	87	12.92	183	28.36
	1/2	86	12.60	178	28.58
	1/4	84	12.11	176	27.97
M160	1	135	80.40	334	250.8
	1/2	133	79.91	328	244.0
	1/4	131	79.12	323	237.8

There is a slight improvement in the number of iterations, but overall the preconditioner remains not scalable. The computational time slightly grows, since the application of the Gauss-Seidel method is more costly in comparison to the Jacobi smoother (indeed the first requires the solution of a triangular system, while the second of a diagonal system). The relaxation parameter again shows some effectiveness, but the reduction is of just a couple of iterations.

In order to find a Multigrid scheme that allows for scalability, we tried to perform the smoothing in two directions that follow the flow.

5.3.3 Two directions GS Multigrid / BFBt

We implemented a smoother that performs two sweeps of Gauss-Seidel in two orthogonal directions, trying to follow the direction of the convective processes, as shown in Figure 3.6. This technique has been proposed in [8, pp. 314-317]. Moreover, we tried to apply the Multigrid using a W-cycle instead of a V-cycle, since it may yield more accurate results, as shown in [14]. These two modifications will produce an increment in the computational time per iteration, i.e. a single application of the preconditioner will be far more time consuming, but hopefully they will make the Multigrid scalable. This combination will be referred to as NS-W2dGS-BFBt. The implementation details are the same as before, except for the number of W-cycles used, which in this case is just 2. Table 5.11 shows the spectral properties of the preconditioned $(1,1)$ block and Schur complement.

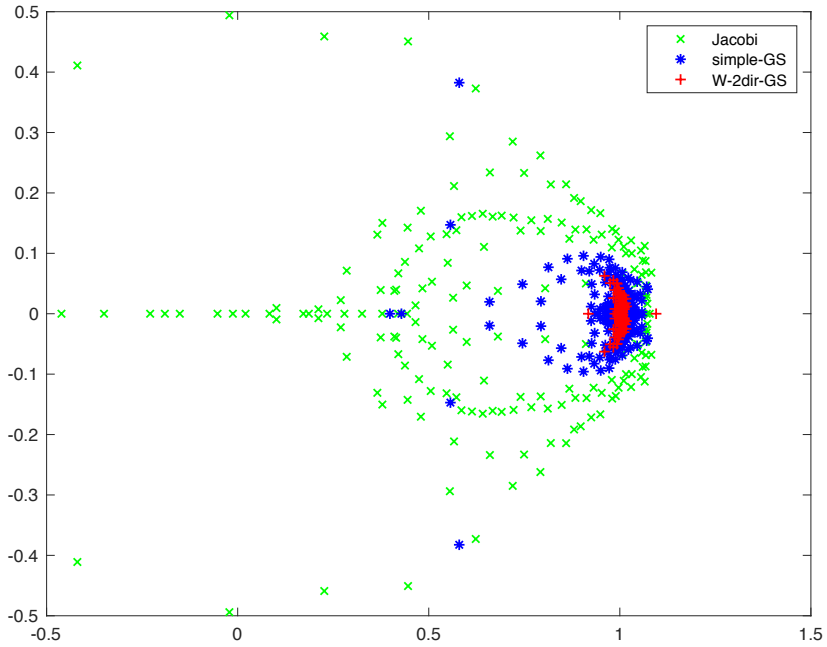
Table 5.11: Spectral properties of the preconditioned matrices for the NS-W2dGS-BFBt problem.

Matrix	Viscosity	α_A	β_A	α_S	β_S
M40	0.1	0.9907	1.022	0.6973	25.46
	0.01	0.9912	1.133	0.4899	21.84
	0.005	0.9153	1.291	0.1962	25.20
M80	0.1	0.9907	1.027	0.2159	26.55
	0.01	0.9889	1.032	0.1242	25.83
	0.005	0.9619	1.130	0.0350	25.25
M160	0.1	0.9908	1.036	0.0554	27.20
	0.01	0.9874	1.036	0.0315	26.68
	0.005	0.9859	1.050	0.0283	26.46

The results are promising, at least for the Multigrid preconditioner: the smallest eigenvalue α_A is very close to 1 and does not suffer from refinement of the mesh or reduction in the viscosity. The spectral radius β_A is slightly above 1 and again does not deteriorate too much changing the properties of the problem: there is a little growth when the viscosity is reduced and there is a decrease when the mesh is refined (this can be explained noticing that the convective phenomena are better represented when the matrix is larger, therefore improving the efficiency of this Multigrid scheme which is based on these phenomena).

A spectral interval so narrow and so stable for the Multigrid preconditioner suggests that this method might produce a scalable preconditioner. To see the improvements with respect to the previous choices, Figure 5.3 shows the spectra of the $(1,1)$ block preconditioned with three Multigrid schemes, using damped Jacobi, simple Gauss-Seidel and the final smoother, for the matrix M40 and viscosity 0.005. It is immediately clear how the last choice of smoother keeps

Figure 5.3: Spectra of various Multigrid schemes for the matrix M40 and $\nu = 0.005$.



almost all the eigenvalues very well clustered around 1 even for a low value of the viscosity. This task is instead failed by the other two possibilities.

In order to understand why the W-cycle is useful, we present the spectral properties obtained with the same $(1,1)$ block preconditioner, only using a V-cycle instead of W. The results are shown in Table 5.12.

Table 5.12: Spectral properties of the $(1,1)$ block without W-cycle.

Matrix	Viscosity	α_A	β_A
M40	0.1	0.9916	1.029
	0.01	0.7948	1.330
	0.005	0.8967	2.577
M80	0.1	0.9912	1.049
	0.01	0.5789	1.491
	0.005	0.9548	4.637
M160	0.1	0.9891	1.090
	0.01	0.3433	1.730
	0.005	0.9163	8.074

It is clear how, without the W-cycle, the preconditioner does not work properly: the spectral radius grows rapidly and is sensible both to mesh refinements and to viscosity changes. It is therefore fundamental to use a W-cycle in order

to get a scalable $(1,1)$ block preconditioner.

We have found a Multigrid scheme that is likely to be scalable, but unfortunately the BFBt preconditioner shows poor spectral properties. From Table 5.11, we can notice that the spectral radius β_S does not deteriorate much, but the minimum eigenvalue in modulus α_S approaches 0 as the mesh is refined. This behavior will surely lead to non-scalability of the whole preconditioner.

Table 5.13: NS-W2dGS-BFBt results for various values of the relaxation parameter.

Matrix	ω	$\nu = 0.1$		$\nu = 0.01$		$\nu = 0.005$	
		iter	time	iter	time	iter	time
M40	1	64	5.90	107	10.25	118	11.43
	1/2	62	5.92	104	9.83	115	10.98
	1/4	61	5.88	102	9.80	111	10.64
	1/10	59	5.60	100	9.61	112	10.50
	1/20	59	5.53	113	10.82	134	12.89
M80	1	88	22.88	181	48.74	253	69.37
	1/2	86	22.23	177	47.31	249	67.28
	1/4	85	22.08	174	46.51	246	65.82
	1/10	82	21.28	171	45.45	242	64.67
	1/20	81	21.17	174	46.02	265	71.95
M160	1	136	126.87	330	346.33	> 400	–
	1/2	135	124.55	328	345.82	> 400	–
	1/4	133	122.58	325	338.70	> 400	–
	1/10	129	118.47	321	331.83	> 400	–
	1/20	128	118.50	319	323.42	> 400	–

Table 5.13 presents the results using the NS-W2dGS-BFBt preconditioner, with various values of the relaxation parameter. It is immediately clear that this combination is not scalable, since the results are almost exactly the ones obtained previously. For the matrix M160 and viscosity 0.005, convergence is not reached within 400 GMRES iterations. The relaxation continues to produce a slight improvement.

It can also be observed a clear growth in the computational time per iteration: with the previous preconditioner, for the M160 matrix, the time per iteration was between 0.6s and 0.7s. In this case it reaches the value of 1s, emphasizing how the smoother used is far more complex.

5.3.4 Two directions GS Multigrid / BFBt-c

The Multigrid scheme using two sweeps of Gauss-Seidel seems to be the good choice to precondition the $(1,1)$ block. In order to obtain a fully scalable method, we implemented the BFBt-c preconditioner, which might behave better

in situations with recirculating flows. This preconditioner has been analyzed in [15]. This combination will be referred to as NS-W2dGS-BFBt-c. The details of the Multigrid implementation are the same as before; moreover, the same Multigrid scheme is used inside the Schur complement preconditioner, to produce the inverse of the Laplacian matrix; the only difference is that 4 W-cycles are used instead of 2 for this task. The spectral properties are shown in Table 5.14; the eigenvalues of the preconditioned $(1,1)$ block are the same as before, since the Multigrid scheme did not change, so they are not reported.

Table 5.14: Spectral properties of the preconditioned matrices for the NS-W2dGS-BFBt-c problem.

Matrix	Viscosity	α_S	β_S
M40	0.1	0.0178	1.0592
	0.01	0.0177	1.0732
	0.005	0.0177	1.5929
M80	0.1	0.0178	1.1514
	0.01	0.0177	1.0732
	0.005	0.0175	1.7966
M160	0.1	0.0178	1.3970
	0.01	0.0177	1.0750
	0.005	0.0176	1.8484

These eigenvalue bounds are promising: the smallest one is constant independently of the mesh size and of the viscosity. The spectral radius shows a slight dependence on these factors, but overall the behavior is definitely better than the BFBt preconditioner. Such a method is likely to produce scalable results. To see the improvement with respect to the previous choice, Figure 5.4 shows the spectra in the complex plane of the BFBt and BFBt-c preconditioned Schur complement, for the matrix M40 and viscosity 0.005.

However, we expect the time per iteration to grow substantially, since various Multigrid iterations are required also inside the Schur complement preconditioner. This means that, if the method is scalable, there will be a gain in the total computational time only if the gain in the number of iterations is large. Moreover, the current situation suggests that the relaxation will not have any effect on the speed of convergence, since the two spectra now have almost the same spectral radius and the preconditioned $(1,1)$ block spectrum is fully contained in the preconditioned Schur complement spectrum. This fact is illustrated in Figure 5.5, which shows the two spectra overlapping, for the matrix M40 and $\nu = 0.005$.

The results obtained are shown in Table 5.15, for values of the viscosity up to 0.005 and also for matrix M320, which is the largest one that we used.

These results confirm what we already anticipated: the NS-W2dGS-BFBt-c

Figure 5.4: Comparison of the spectra of the preconditioned Schur complement for BFBt and BFBt-c preconditioners, for matrix M40 and $\nu = 0.005$.

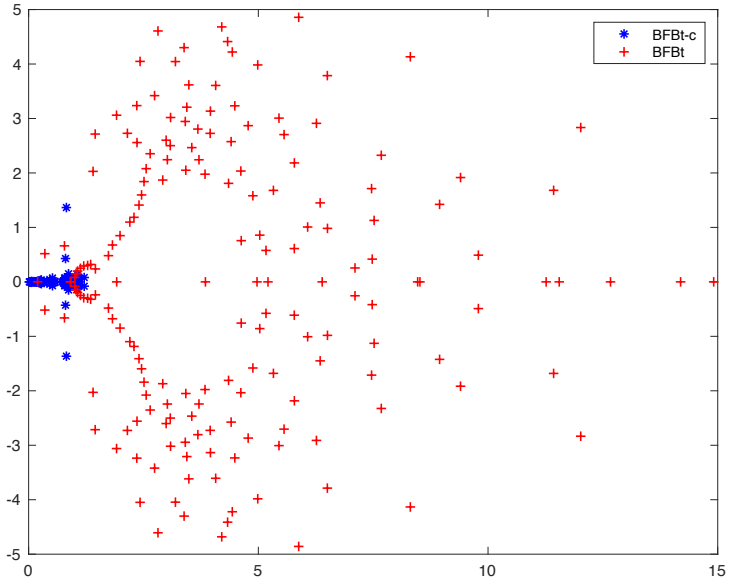
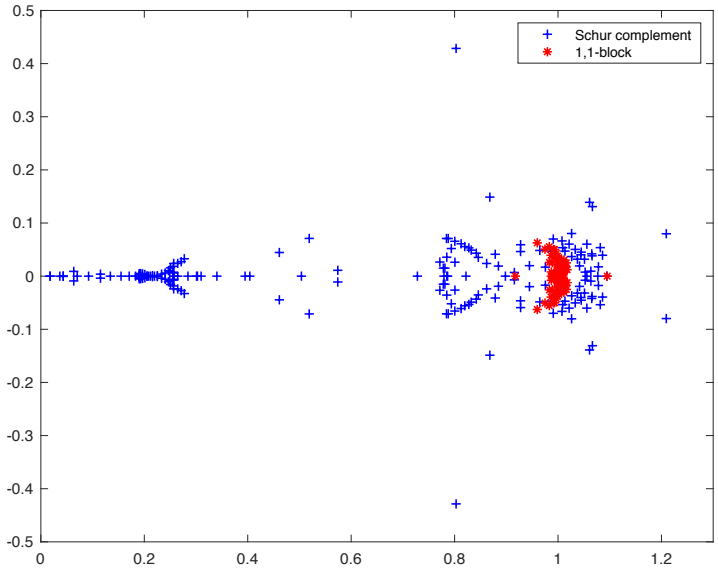


Figure 5.5: Spectra of the preconditioned $(1,1)$ block and Schur complement.



preconditioner is scalable. The number of iterations increases very slowly and the computational time grows linearly for all the viscosities. Moreover, the speed of convergence is just slightly dependent on the viscosity, showing good robustness in this sense.

The time per iteration is definitely larger than before; for the matrix M160,

Table 5.15: NS-W2dGS-BFBt-c results without relaxation.

Matrix	$\nu = 0.1$		$\nu = 0.01$		$\nu = 0.005$	
	iter	time	iter	time	iter	time
M40	52	16.49	62	20.09	68	21.91
M80	57	47.88	65	54.13	70	57.87
M160	61	173.08	66	189.71	77	217.99
M320	64	677.26	69	729.35	79	830.51

it has risen to more than 2.80 seconds per GMRES iteration. Concerning the matrix M40, the final computational time is more than the double of the one obtained with the previous preconditioner, since the gain in the number of iterations is not sufficiently high; with the matrix M80, some time advantage begins to arise with the lowest viscosity; for the matrix M160 the advantage is clear, since the iterations count goes from more than 400 to just 77. The results for the matrix M320 emphasize how the nice behavior of this preconditioner is preserved refining the mesh even further.

Figure 5.6: Computational time with respect to number of unknowns in the system, for various viscosity values, using the scalable preconditioner.

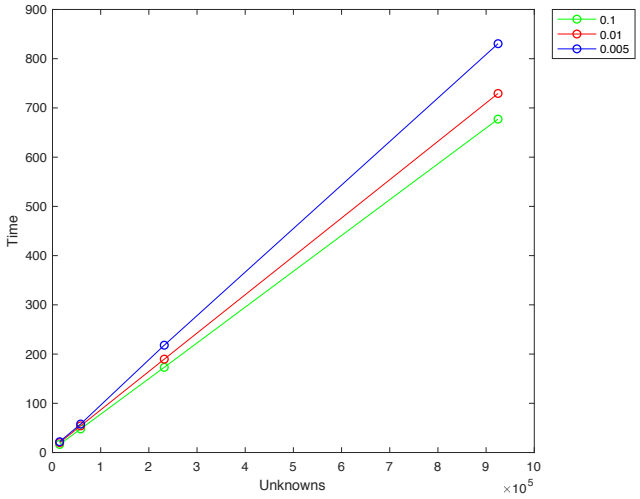


Figure 5.6 plots the computational time needed against the number of unknowns of the system for the scalable preconditioner. It is clear that the growth is linear and the viscosity does not affect too much the results. Figure 5.7 instead plots a comparison of the behavior of the computational time in the case of scalable and non scalable preconditioners; all the three viscosities are shown, for matrices up to M160. We can clearly see that for the non scalable preconditioner, the computational time grows quadratically; for the highest viscosity, there is

not much difference between the behavior in the scalable and non scalable case, since this situation is the closest one to the Stokes problem. We report also the convergence profile of GMRES: Figure 5.8 shows the absolute residual for matrix M40 and different values of viscosity. Figure 5.9 instead shows the convergence profile for matrices M40, M80 and M160 with the lowest viscosity.

Figure 5.7: Comparison of the growth rate of the computational time for the scalable and non scalable preconditioners, for various viscosities and matrices up to M160.

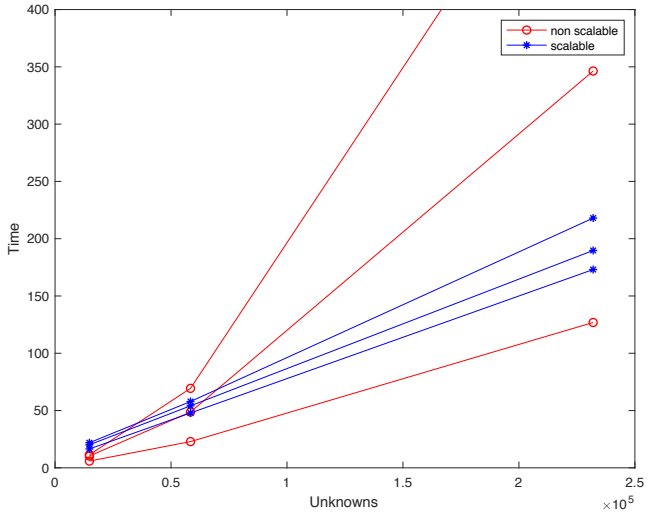
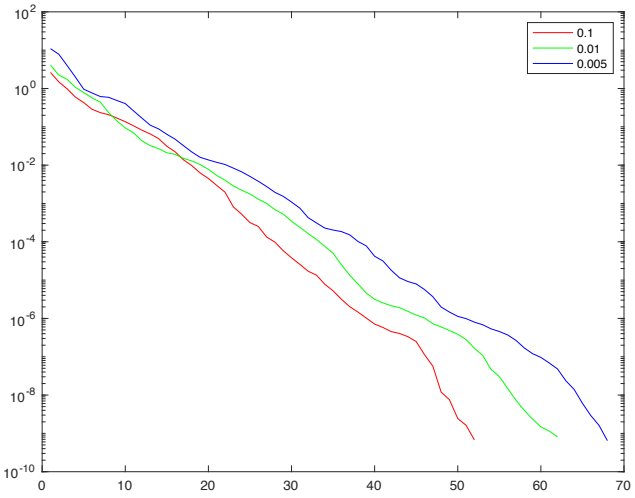
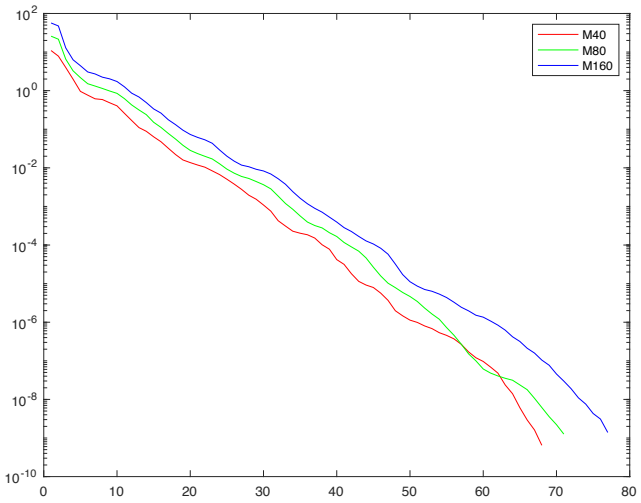


Figure 5.8: Convergence profile for the matrix M40 with different viscosities.



We also checked whether the relaxation had any effect on the speed of convergence, but we could not find any value of ω able to produce an improvement. This fact, again, is in accordance with what we expected, given the spectral

Figure 5.9: Convergence profile for various matrices and viscosity 0.005.



properties of the preconditioner used.

We then proceeded to test the NS-W2dGS-BFBt-c preconditioner with viscosity 0.001, to see if the scalability is preserved even for lower viscosities. Unfortunately, the spectral properties deteriorate enormously, as shown in Table 5.16

Table 5.16: Spectral properties of the preconditioned matrices for the NS-sGS-BFBt-c problem with viscosity 0.001.

Matrix	α_A	β_A	α_S	β_S
M40	0.3812	24.11	0.0176	3.252
M80	0.7775	465.1	0.0178	40.37

It is clear how the eigenvalues are completely different from the ones found for previous viscosities: the Multigrid preconditioner fails completely for the matrix M80 and the spectral radius goes beyond 400. The Schur complement preconditioner does not behave too badly: indeed the eigenvalue with minimum modulus remains the same as before, while the spectral radius grows. However, this growth may be dictated by the fact that, in order to apply the Arnoldi procedure to the Schur complement, a Multigrid scheme is used to invert the $(1,1)$ block. Therefore, it may be that the Multigrid works so badly in this case that it affects the eigenvalues of the Schur complement found with the Arnoldi method.

Thus, the best guess that we can make is that from a viscosity between 0.005 and 0.001, the Multigrid scheme suddenly fails completely to follow the convective flux, leading to poor spectral properties, which in turn produce a non scalable

preconditioner. To overcome this problem, we tried different strategies: we improved the number of W-cycles applied at any iteration, but the improvement was very slight, definitely not enough to produce scalability; we then tried to use more smoothing steps, but this did not produce any progress; we recalculated the matrices, using more stabilization for the convection-diffusion operator, but this still was not showing progresses; we then modified the smoother so that the Gauss-Seidel sweeps were four and not two, hoping to follow better the flow, but this still did not improve the results.

Table 5.17: Comparison of inexact constraint preconditioner and triangular block preconditioner for viscosity 0.005.

Matrix	ICP		TBP	
	iter	time	iter	time
M40	68	21.91	79	21.18
M80	70	57.87	83	57.86
M160	77	217.99	85	198.02
M320	79	830.51	82	734.62

We then proceeded to compare the inexact constraint preconditioner (ICP) to the triangular block preconditioner (TBP) presented in [20] and implemented using Algorithm B.5, in order to assess which one behaves better. We used the same Multigrid scheme and Schur complement preconditioner that we tested in the NS-W2dGS-BFBt-c case. Since the preconditioner structure is simpler, we expect a smaller time per iteration, but a higher iteration count, thus it is not possible to tell a priori which preconditioner will behave better. For this test, we used a viscosity of 0.005 and matrices up to M320. The results are shown in Table 5.17.

It is clear that the number of iterations for the constraint preconditioner is smaller, but the TBP seems to be more scalable, since the iteration count is more stable, leading to a computational time significantly smaller for the larger matrices. However, also the TBP fails to provide good results for a viscosity of 0.001, since the preconditioners for the blocks are the same as before.

Conclusion

Concerning the goals that we set in the introduction of this work, we have been able to develop a smoother that follows the convective flux, at least up to values of the viscosity of 0.005; we also realized that such a smoother is fundamental for the scalability of the whole process, since standard smoothers present bad spectral properties already with a viscosity of 0.01. We have also discovered that the preconditioner for the Schur complement changes completely from the Stokes to the Navier-Stokes problem, due to the presence of convection; for the Navier-Stokes equations, we have been able to build a preconditioner for the Schur complement that remains scalable even for viscosities below 0.005.

For values of ν lower than 0.005, unfortunately we could not find a Multigrid scheme capable of maintaining scalability: the problem is likely to be related to the smoothing phase, however using a different discretization strategy or employing a more complicated stabilization technique for the convective operator might lead to more promising results; in case a more robust smoother cannot be found, the augmented Lagrangian approach presented in [2] has been tested successfully for very low viscosities and surely represents the most promising approach, despite the extreme complexity of the algorithms involved.

We also showed that for very large problems, the inexact approach is far more efficient than the exact one, since the requirement of solving exactly the Schur complement system is a task so complicated that it destroys the performance of the method. Moreover, we came to the conclusion that for this problems, the relaxation has almost no effect and the small benefit that sometimes appears is surely not enough to justify the expensive procedure of computing the eigenvalues.

Appendix A

Application of Theorem 2.3

We will prove that the kernel and the range of the preconditioned matrix have trivial intersection. We will consider the right-preconditioned matrix, since the computations are easier. Let us start recalling that the matrix used is

$$M = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}.$$

The preconditioner instead is the Exact Constraint Preconditioner

$$P = \begin{bmatrix} I & 0 \\ BP_A^{-1} & I \end{bmatrix} \begin{bmatrix} P_A & 0 \\ 0 & -P_S \end{bmatrix} \begin{bmatrix} I & P_A^{-1}B^T \\ 0 & I \end{bmatrix},$$

with $P_S = BP_A^{-1}B^T$.

First of all, we need to compute the inverse of the preconditioner. The inverse of the diagonal block is easy, let us try to invert the triangular blocks. We suppose that the inverse is still block-triangular; moreover, the diagonal blocks must remain identities. To find out the remaining block of the inverse, that we will call X, we proceed as follows

$$\begin{bmatrix} I & 0 \\ BP_A^{-1} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} = \begin{bmatrix} I & 0 \\ BP_A^{-1} + X & I \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}.$$

Therefore it follows that $X = -BP_A^{-1}$. So, the inverse of the preconditioner is

$$\begin{aligned} P^{-1} &= \begin{bmatrix} I & -P_A^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} P_A^{-1} & 0 \\ 0 & -P_S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -BP_A^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} P_A^{-1} - P_A^{-1}B^T P_S^{-1} BP_A^{-1} & P_A^{-1}B^T P_S^{-1} \\ P_S^{-1}BP_A^{-1} & -P_S^{-1} \end{bmatrix}. \end{aligned}$$

The preconditioner P_S is singular, so it is not correct to consider its inverse. Let us define the space $\mathcal{S}^\perp$ as the orthogonal complement of the vector $\mathbf{1}$. We can show that P_S is a well-defined nonsingular operator from $\mathcal{S}^\perp$ to itself. In this way, we can define P_S^{-1} onto $\mathcal{S}^\perp$.

Proposition A.1. *The Schur complement $S = BA^{-1}B^T$ and its exact preconditioner $P_S = BP_A^{-1}B^T$ are nonsingular operators from $\mathcal{S}^\perp$ to itself.*

Proof. We already know that the kernel of S and P_S is of dimension one and contains only the vector $\mathbf{1}$. Therefore we just need to show that the range of S and P_S does not contain the vector $\mathbf{1}$.

Suppose there is a vector $v \in \mathcal{S}^\perp$ such that

$$BA^{-1}B^T v = \mathbf{1}.$$

Since v cannot be the vector $\mathbf{1}$, then $B^T v$ is a nonzero vector; thus, also $A^{-1}B^T v$ is a nonzero vector. Define $w = A^{-1}B^T v$, then

$$Bw = \mathbf{1},$$

but this is a contradiction: indeed, the range of B is the orthogonal complement to the kernel of B^T, and thus cannot contain $\mathbf{1}$.

The same holds similarly for P_S. □

We can therefore define P_S^{-1} as a nonsingular operator that maps $\mathcal{S}^\perp$ to itself and $\mathbf{1}$ to 0.

We can now compute the preconditioned matrix MP^{-1}:

$$\begin{aligned} MP^{-1} &= \begin{bmatrix} AP_A^{-1} - AP_A^{-1}B^T P_S^{-1} BP_A^{-1} + B^T P_S^{-1} BP_A^{-1} & AP_A^{-1}B^T P_S^{-1} - B^T P_S^{-1} \\ BP_A^{-1} - BP_A^{-1}B^T P_S^{-1} BP_A^{-1} & BP_A^{-1}B^T P_S^{-1} \end{bmatrix} \\ &= \begin{bmatrix} AP_A^{-1} + (I - AP_A^{-1})B^T P_S^{-1} BP_A^{-1} & (AP_A^{-1} - I)B^T P_S^{-1} \\ (I - P_S P_S^{-1})BP_A^{-1} & P_S P_S^{-1} \end{bmatrix}. \end{aligned}$$

Due to the nonstandard definition of P_S^{-1} that we saw before, we cannot say that $P_S P_S^{-1} = I$; however, we can state the following

Proposition A.2.

$$(I - P_S P_S^{-1})B = 0.$$

Proof. Let us analyze the action of $(I - P_S P_S^{-1})$ on a generic vector $x \in \mathbb{R}^m$: if $x \neq \mathbf{1}$, then P_S can be inverted and $P_S P_S^{-1} x = x$, so $(I - P_S P_S^{-1})x = 0$. If instead $x = \mathbf{1}$, then $P_S^{-1} x = 0$ and so $(I - P_S P_S^{-1})x = x$.

Take now a vector $y \in \mathbb{R}^n$ and consider By: as we saw before, $\mathbf{1}$ is not in the range of B, so By is surely a vector different from $\mathbf{1}$. Thus, is we consider

the expression $(I - P_S P_S^{-1})By$, we can be sure that it will be well defined and the result will be the zero vector, for any $y \in \mathbb{R}^n$. This in turn proves that $(I - P_S P_S^{-1})B$ is the zero matrix. □

Therefore we can simplify the preconditioned matrix, which becomes

$$MP^{-1} = \begin{bmatrix} AP_A^{-1} + (I - AP_A^{-1})B^T P_S^{-1} B P_A^{-1} & (AP_A^{-1} - I)B^T P_S^{-1} \\ 0 & P_S P_S^{-1} \end{bmatrix}.$$

Define $F = AP_A^{-1} + (I - AP_A^{-1})B^T P_S^{-1} B P_A^{-1}$. Let us look for the kernel of MP^{-1}, i.e. the vectors v and q such that

$$\begin{bmatrix} F & (AP_A^{-1} - I)B^T P_S^{-1} \\ 0 & P_S P_S^{-1} \end{bmatrix} \begin{bmatrix} v \\ q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The second block of equations reads

$$P_S P_S^{-1} q = 0.$$

This is satisfied if either $P_S^{-1} q = 0$ or $P_S^{-1} q = \mathbf{1}$. Since $\mathbf{1}$ is not in the range of P_S^{-1}, it must be $P_S^{-1} q = 0$ and thus $q = \mathbf{1}$.

The first block of equations instead reads

$$Fv + (AP_A^{-1} - I)B^T P_S^{-1} q = 0.$$

Substituting $q = \mathbf{1}$ yields

$$Fv = 0.$$

So v lays in the kernel of F, which we do not need to characterize. Therefore the kernel of the preconditioned matrix is

$$\ker(MP^{-1}) = \left\{ \begin{bmatrix} v \\ q \end{bmatrix} \middle| v \in \ker(F),\, q = \mathbf{1} \right\}.$$

Now, we want to understand if these vectors can lay also in the range of MP^{-1}: let us look for vectors w and s such that

$$MP^{-1} \begin{bmatrix} w \\ s \end{bmatrix} = \begin{bmatrix} v \\ q \end{bmatrix}.$$

The second block of equations reads

$$P_S P_S^{-1} s = q.$$

Suppose that $s \neq \mathbf{1}$, since otherwise the result would be 0. Then, for such a vector s, it holds that $P_S P_S^{-1} s = s$, since P_S is invertible on $\mathcal{S}^{\perp}$. This means that $s = q$, but since $q = \mathbf{1}$, this is a contradiction.

Therefore, there cannot exist vectors that lay both in the kernel and in the range of MP^{-1}.

Appendix B

Algorithms

We will now report the algorithms that have been used. They apply the exact constraint preconditioner (Algorithm B.1), the inexact constraint preconditioner (Algorithms B.2, B.3 and B.4) and the triangular block preconditioner (Algorithm B.5). They are slightly different from the ones presented before, since they have been optimized to reduce the number of operations. In all the algorithms, we suppose that the application of the preconditioner requires the solution of the linear system

$$\begin{bmatrix} P_A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}.$$

Algorithm B.1 Application of ECP

$A \approx LL^T$ using incomplete Cholesky factorization
$L\,x_1 = r_1$ using a direct solver
$L^T x_2 = x_1$ using a direct solver
$f = B\,x_2 - r_2$
$(B(LL^T)^{-1}B^T)y_2 = f$ using PCG with Q as preconditioner
$g = B^T y_2$
$L\,w_1 = g$ using a direct solver
$L^T w_2 = w_1$ using a direct solver
$y_1 = x_2 - w_2$

Algorithm B.2 Application of ICP with Multigrid and mass matrix

$P_A x = r_1$ using Multigrid
$f = Bx - r_2$
$Q\,y_2 = f$ using a direct solver
$g = B^T y_2$
$P_A w = g$ using Multigrid
$y_1 = x - w$

Algorithm B.3 Application of ICP with Multigrid and BFBt

$P_A x = r_1$ using Multigrid
$f = Bx - r_2$
$P_S y_2 = f$ using BFBt:
  $(BQ_v^{-1}B^T)y_{2,1} = f$ using factorization or Multigrid
  $y_{2,2} = B^T y_{2,1}$
  $Q_v y_{2,3} = y_{2,2}$ using a direct solver
  $y_{2,4} = A\, y_{2,3}$
  $Q_v y_{2,5} = y_{2,4}$ using a direct solver
  $y_{2,6} = B\, y_{2,5}$
  $(BQ_v^{-1}B^T)y_{2,7} = y_{2,6}$ using factorization or Multigrid
  $y_2 = \omega\, y_{2,7}$
$g = B^T y_2$
$P_A w = g$ using Multigrid
$y_1 = x - w$

Algorithm B.4 Application of ICP with Multigrid and BFBt-c

$P_A x = r_1$ using Multigrid
$f = Bx - r_2$
$P_S y_2 = f$ using BFBt-c:
  $Q\, y_{2,1} = f$ using a direct solver
  $y_{2,2} = B^T y_{2,1}$
  $L\, y_{2,3} = y_{2,2}$ using Multigrid
  $y_{2,4} = A\, y_{2,3}$
  $L\, y_{2,5} = y_{2,4}$ using Multigrid
  $y_{2,6} = B\, y_{2,5}$
  $Q\, y_{2,7} = y_{2,6}$ using a direct solver
  $y_2 = \omega\, y_{2,7}$
$g = B^T y_2$
$P_A w = g$ using Multigrid
$y_1 = x - w$

Algorithm B.5 Application of TBP with Multigrid and BFBt-c

$P_S y_2 = r_2$ using BFBt-c:
  $Q\, y_{2,1} = r_2$ using a direct solver
  $y_{2,2} = B^T y_{2,1}$
  $L\, y_{2,3} = y_{2,2}$ using Multigrid
  $y_{2,4} = A\, y_{2,3}$
  $L\, y_{2,5} = y_{2,4}$ using Multigrid
  $y_{2,6} = B\, y_{2,5}$
  $Q\, y_{2,7} = y_{2,6}$ using a direct solver
  $y_2 = \omega\, y_{2,7}$
$g = r_1 + B^T y_2$
$P_A y_1 = g$ using Multigrid

Bibliography

[1] Michele Benzi, Gene H Golub, and Jorg Liesen. "Numerical Solution of Saddle Point Problems". In: *Acta Numerica* 14 (2005), pp. 1–137.

[2] Michele Benzi and Maxim Olshanskii. "An augmented Lagrangian-based approach to the Oseen problem". In: *SIAM Journal on Scientific Computing* 28 (2006), pp. 2095–2113.

[3] Luca Bergamaschi. "On eigenvalue distribution of constraint-preconditioned symmetric saddle point matrices". In: *Numerical Linear Algebra with Applications* 194 (2010), pp. 754–772.

[4] Luca Bergamaschi, Massimiliano Ferronato, and Giuseppe Gambolati. "Novel preconditioners for the iterative solution to FE-discretized coupled consolidation equations". In: *Computer Methods in Applied Mechanics and Engineering* 196 (2007), pp. 2647–2656.

[5] Luca Bergamaschi and Angeles Martinez. "RMCP: Relazed Mixed Constraint Preconditioners for saddle point linear systems arising in geomechanics". In: *Computer Methods in Applied Mechanics and Engineering* 221-222 (2012), pp. 54–62.

[6] Peter Brown and Homer Walker. "GMRES on (nearly) singular systems". In: *SIAM Journal on Matrix Analysis and Applications* 18 (1997), pp. 37–51.

[7] Howard Elman. "Preconditioning For The Steady-State Navier-Stokes Equations With Low Viscosity". In: *SIAM Journal on Scientific Computing* 20 (2001), pp. 1299–1316.

[8] Howard Elman, David Sylvester, and Andy Wathen. *Finite Elements and Fast Iterative Solvers*. 2nd ed. Oxford University Press, 2014.

[9] Patrick Farrell, Lawrence Mitchell, and Florian Wechsung. "An augmented Lagrangian preconditioner for the 3D stationary incompressible Navier-Stokes equations at high Reynolds number". In: (2019). arXiv:1810.03315v2 [math.NA].

[10] Anne Greenbaum, Vlastimil Ptak, and Zdenek Strakos. "Any nonincreasing convergence curve is possible for GMRES". In: *SIAM Journal on Matrix Analysis and Applications* (1996), pp. 465–469.

[11] David Kay, Daniel Loghin, and Andrew Wathen. "A Preconditioner for the Steady-State Navier-Stokes Equations". In: *SIAM Journal on Scientific Computing* 24 (2002), pp. 237–256.

[12] Carsten Keller, Nicholas Gould, and Andrew Wathen. "Constraint Preconditioning for Indefinite Linear Systems". In: *SIAM Journal on Matrix Analysis and Application* 21.4 (2000), pp. 1300–1317.

[13] Tim Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1995.

[14] Maxim Olshanskii and Arnold Reusken. "Convergence analysis of a multigrid method for a convection-dominated model problem". In: *SIAM Journal on Numerical Analysis* 42 (2004), pp. 1261–1291.

[15] Maxim Olshanskii and Yuri Vassilevski. "Pressure Schur Complement Preconditioners for the Discrete Oseen Problem". In: *SIAM Journal on Scientific Computing* 29.6 (2007), pp. 2686–2704.

[16] Alfio Quarteroni. *Numerical Models for Differential Problems*. 2nd ed. Vol. 8. Springer, 2012.

[17] Alison Ramage. "A multigrid preconditioner for stabilised discretisations of advection-diffusion problems". In: *Journal of Computational and Applied Mathematics* 110 (1999), pp. 187–203.

[18] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. 2nd ed. PWS, 1996.

[19] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. 2nd ed. SIAM, 2011.

[20] David Silvester et al. "Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow". In: *Journal of Computational and Applied Mathematics* 128 (2001), pp. 261–279.

[21] Ulrich Trottenberg, Cornelis Oosterlee, and Anton Schuller. *Multigrid*. Academic Press, 2001.

[22] Pieter Wesseling. *An Introduction to Multigrid Methods*. John Wiley and Sons, 1992.