algorithms

MDPI

Article

Scalable Block Preconditioners for Linearized Navier-Stokes Equations at High Reynolds Number

Filippo Zanetti [1] and Luca Bergamaschi [2,*]

[1] School of Mathematics, University of Edinburgh, EH9 3FD Edinburgh, UK, F.Zanetti@sms.ed.ac.uk
[2] Department of Civil Environmental and Architectural Engineering, University of Padua, Padova 35122, Italy
* Correspondence: luca.bergamaschi@unipd.it

Received: 21 July 2020; Accepted: 14 August 2020; Published: 16 August 2020

Abstract: We review a number of preconditioners for the advection-diffusion operator and for the Schur complement matrix, which, in turn, constitute the building blocks for Constraint and Triangular Preconditioners to accelerate the iterative solution of the discretized and linearized Navier-Stokes equations. An intensive numerical testing is performed onto the driven cavity problem with low values of the viscosity coefficient. We devise an efficient multigrid preconditioner for the advection-diffusion matrix, which, combined with the commuted BFBt Schur complement approximation, and inserted in a 2×2 block preconditioner, provides convergence of the Generalized Minimal Residual (GMRES) method in a number of iteration independent of the meshsize for the lowest values of the viscosity parameter. The low-rank acceleration of such preconditioner is also investigated, showing its great potential.

Keywords: scalable preconditioners; Navier-Stokes equations; GMRES method; low-rank updates; multigrid

1. Introduction

The task of numerically solving 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 beginning 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 preconditioners in the framework of iterative 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 [1–5]; we choose to use the Constraint Preconditioner, already analyzed in all its forms in [6–8], and the more popular, in the Fluid Dynamics community, Block Triangular Preconditioner, employed e.g., in [9,10]. We also mention the comprehensive review [11] of preconditioners for saddle point linear systems, and the references therein.

We use the GMRES method and 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, in order to obtain an efficient solver. 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 also involves convective processes. The generalization of Multigrid preconditioners to these kind of situations requires a robust smoother, which can be built using a stationary method involving a pattern that follows the convective flow. This approach has already been tested in [2,12,13].

With regard to the preconditioner for the Schur complement, we will mainly use the preconditioner developed in [9] and improved in [14]. These techniques are based on the assumption 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 well-known 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.
- Review and compare a number of Schur complement preconditioners available in the literature.
- Use information on the spectrum of the preconditioned $(1,1)$ and Schur complement blocks to improve the overall preconditioner and possibly improve scalability in the case of dominating advection.

The rest of the paper is structured as follows: In Section 2 , we describe the continuous problem and its weak formulation, while Section 3 is devoted to the development of a stable Mixed Finite Element formulation, the Picard linearization process, and the properties of the saddle point linear systems to be solved at each nonlinear iteration. In Section 4 , we describe two block preconditioners for saddle point systems; Section 5 analyzes a number of Multigrid preconditioners for the $(1,1)$ block and some approximations to the Schur complement matrix. In Section 6, we describe how a low-rank matrix can be used to accelerate the previously described preconditioners. Section 7 provides a thorough numerical testing of the block preconditioners described in the previous sections onto a the challenging lid driven cavity problem, showing the almost perfect scalability of the two most sophisticated variants for the lowest value of the viscosity parameter. Section 8 concludes the paper.

Notation. Throughout the paper, we will write vectors (and vector functions) in boldface. We will use the symbol M to denote a preconditioner which approximates a given coefficient matrix $A (M \approx A)$, while the symbol P will refer to the preconditioner in its inverse form ($P \approx A^{-1}$).

2. Problem Setting and Discretization

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

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

where $\Omega \subset \mathbb{R}^3$ is the domain in 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 usually used in a different form: using a unit density, we obtain the following version of the Navier-Stokes equations

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \mathbf{f}, & \mathbf{x} \in \Omega, t > 0 \\ \nabla \cdot \mathbf{u} = 0, & \mathbf{x} \in \Omega, t > 0 \end{cases} \tag{1}$$

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 $\nabla \cdot \mathbf{u} = 0$ 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) 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) 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) without the time derivative of $\mathbf{u}$.

2.1. 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), then 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) become a linear system of equations, known as the stationary Stokes equations:

$$\begin{cases} -\nu \Delta \mathbf{u} + \nabla p = \mathbf{f}, & \mathbf{x} \in \Omega, t > 0 \\ \nabla \cdot \mathbf{u} = 0, & \mathbf{x} \in \Omega, t > 0 \end{cases} \tag{2}$$

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 (2) have the great advantage of being linear, which makes them easier to work with, both analytically and numerically.

2.2. Weak Formulation

The general formulation of Equations (1) and (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

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

and the Sobolev Space

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

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 definesd 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

$$\int_\Omega \nu \nabla \mathbf{u} \cdot \nabla \mathbf{v} \, d\Omega + \int_\Omega [(\mathbf{u} \cdot \nabla)\mathbf{u}] \cdot \mathbf{v} \, d\Omega - \int_\Omega p \, \nabla \cdot \mathbf{v} \, d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} \, d\Omega + \int_{\partial\Omega} \left(\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n}\right) \cdot \mathbf{v} \, dS \quad \forall \mathbf{v} \in V. \quad (3)$$

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

$$\int_\Omega q \, \nabla \cdot \mathbf{u} \, d\Omega = 0 \quad \forall q \in Q. \quad (4)$$

Accordingly, an alternative version of the Navier–Stokes equations is: find $\mathbf{u} \in V$ and $p \in Q$ such that (3) and (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 of the integrals are well defined (see [15]).

3. Finite Element Method

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

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

$$\begin{cases} c(\mathbf{u},\mathbf{u},\mathbf{v}) + a(\mathbf{u},\mathbf{v}) + b(\mathbf{v},p) = (\mathbf{f},\mathbf{v}) & \forall \mathbf{v} \in V \\ b(\mathbf{u},q) = 0 & \forall q \in Q \end{cases} \tag{5}$$

where $a\colon V \times V \mapsto \mathbb{R}$ and $b\colon V \times Q \mapsto \mathbb{R}$ are bilinear forms, while $c\colon V \times V \times V \mapsto \mathbb{R}$ is a trilinear form, defined as

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) &= \int_\Omega \nu \nabla \mathbf{u} \cdot \nabla \mathbf{v}\, d\Omega, \\ b(\mathbf{u},q) &= -\int_\Omega q\, \nabla \cdot \mathbf{u}\, d\Omega \\ c(\mathbf{u},\mathbf{u},\mathbf{v}) &= \int_\Omega [(\mathbf{u}\cdot\nabla)\mathbf{u}] \cdot \mathbf{v}\, d\Omega. \end{aligned}$$

and

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

3.1. Galerkin Approximation

Equations (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 that is 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 becomes:

$$\begin{cases} a(\mathbf{u}_h,\mathbf{v}_h) + c(\mathbf{u}_h,\mathbf{u}_h,\mathbf{v}_h) + b(\mathbf{v}_h,p_h) = (\mathbf{f},\mathbf{v}_h) & \forall \mathbf{v}_h \in V_h \\ b(\mathbf{u}_h,q_h) = 0 & \forall q_h \in Q_h \end{cases}. \tag{6}$$

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 that are 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 (6) 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. Accordingly, the method can be formalized, as follows

$$\begin{cases} a(\mathbf{u}_h^{(k+1)},\mathbf{v}_h) + c(\mathbf{u}_h^{(k)},\mathbf{u}_h^{(k+1)},\mathbf{v}_h) + b(\mathbf{v}_h,p_h^{(k+1)}) = (\mathbf{f},\mathbf{v}_h) & \forall \mathbf{v}_h \in V_h \\ b(\mathbf{u}_h^{(k+1)},q_h) = 0 & \forall q_h \in Q_h \end{cases} \tag{7}$$

for $k = 0, 1, \ldots$ 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 $\mathbf{u}_h^{(-1)} = \mathbf{0}$. This formulation of the Navier–Stokes problem is known as Oseen problem and it is the method that will be used in this work. We refer e.g., to the book [16] for more details.

3.2. Stabilization of the Convection-Diffusion Term

The first terms in the Oseen problem (7) 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 when 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 it will be at least 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 approaches include the Streamline upwind Petrov–Galerkin (SUPG), the Algebraic subgrid scale (ASGS) stabilized Finite Elements, and the Galerking Least-Square (GLS) methods, but their theory and implementation are far more complicated, and they are beyond the scope of this paper (see [15] for more details). 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. Subsequently, 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 $Pe_h \geq 1$

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

This choice of the stabilization parameter is taken from [2].

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

3.3. Choice of the Finite Element Spaces

The choice of Finite Element spaces V_h and Q_h must satisfy the well-known inf-sup (or LBB) condition [2]. The lowest order pair 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 shows the two kind of elements used for the spaces Q_h and V_h.

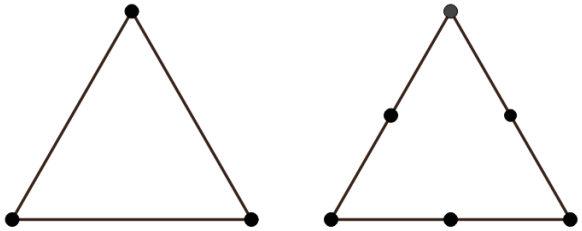


Figure 1. Linear and quadratic triangular elements, used in the Taylor-Hood approximation.

3.4. Algebraic Formulation

Let us now denote with $\boldsymbol{\phi}_i,\ i = 1, \dots, N_u$ a basis of V_h, and with $\psi_i,\ i = 1, \dots, N_p$ a basis for Q_h. The unknown functions $\mathbf{u}_h^{(k+1)}$ and $p_h^{(k+1)}$ in (7) can be written as a linear combination of these basis functions, as

$$\mathbf{u}_h^{(k+1)} = \sum_{i=1}^{N_u} \boldsymbol{\phi}_i u_i, \qquad p_h^{(k+1)} = \sum_{i=1}^{N_p} \psi_i p_i.$$

At each iteration $k+1$ of the Picard method, a linear system has to be solved, which takes the following form

$$\begin{bmatrix} \nu A + N & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix}. \tag{8}$$

where $\mathbf{u}$ and $\mathbf{p}$ 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. Linear systems with this particular block structure are usually called saddle point systems.

In the following, $(\mathbf{x} : \mathbf{y})$ represents the element-wise product of two vectors $\mathbf{x}$ and $\mathbf{y}$, while $(\mathbf{x} \cdot \mathbf{y})$ represents the Euclidian scalar product. The matrix A is the discrete vector Laplacian

$$A = (a_{ij}), \qquad a_{ij} = \int_\Omega \nabla \boldsymbol{\phi}_i : \nabla \boldsymbol{\phi}_j, \qquad i, j = 1, \dots, N_u,$$

the matrix B is called the divergence matrix and is defined as

$$B = (b_{kj}), \qquad b_{kj} = -\int_\Omega \psi_k \nabla \cdot \boldsymbol{\phi}_j, \qquad j = 1, \dots, N_u, \quad k = 1, \dots, N_p.$$

The matrix N is the discrete vector convection matrix, which depends on the current estimate of the velocity $\mathbf{u}_h^{(k)}$ and has the following definition

$$N = (n_{ij}), \qquad n_{ij} = \int_\Omega \left(\mathbf{u}_h^{(k)} \cdot \nabla \boldsymbol{\phi}_j \right) \cdot \boldsymbol{\phi}_i, \qquad i, j = 1, \dots, N_u.$$

The $(1,1)$ block of (8) will be referred to as matrix $F = \nu A + N$.

Notation. In the sequel, we will denote as $n \equiv N_u$ the size of the $(1,1)$ block in (8), while $m \equiv N_p$ will indicate the number of rows of B.

We will now define some other matrices that will be useful later: the velocity-mass matrix $\hat{Q} \in \mathbb{R}^{n \times n}$ and the (pressure-)mass matrix $Q \in \mathbb{R}^{m \times m}$ as

$$\hat{Q} = (\hat{q}_{ij}), \qquad \hat{q}_{ij} = \int_\Omega \boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_j \, d\Omega \tag{9}$$

$$Q = (q_{ij}), \qquad q_{ij} = \int_\Omega \psi_i \psi_j \, d\Omega. \tag{10}$$

They are both square, symmetric, and positive definite. From the point of view of operators, the mass matrix corresponds to a discrete identity operator.

3.5. Properties of Saddle Point Matrices

Theorem 1. *Consider a problem where $\partial\Omega = \Gamma_D$ (no Neumann boundary conditions), discretized using a stable approximation on a shape-regular quasi-uniform subdivision of $\mathbb{R}^2$. Subsequently, 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. See [2]. □

This theorem has an important consequence: regardless 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 $\mathbf{q} = \mathbf{1}$ is in the kernel of the Schur complement, so it is not positive definite, being singular) and, because Q is positive definite, also the Schur complement will always be positive semi-definite. Let us define h as the maximum diameter (`diam`) of all elements K in the discretization, whereas $\texttt{diam}(K) = \max\{|\boldsymbol{x} - \boldsymbol{y}|, \boldsymbol{x}, \boldsymbol{y} \in K\}$. Then, matrix $Q^{-1}BA^{-1}B^T$ has a condition number that can be bounded independently of the size h of the discretization. This property does not hold for matrix A, whose condition number that grows indefinitely as h goes to 0. This fact plays a crucial role in the numerical solution of the Stokes and Navier–Stokes equations.

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

4. Block Preconditioners for the Navier-Stokes Discretized Systems

The following results are in the line of those e.g., of [17–19], in which symmetric saddle point matrices are considered.

4.1. Block Triangular Preconditioner

We will now analyze the Block Triangular Preconditioner (BTP) applied to the discretized Navier–Stokes equations. Let us define H the $n + m \times n + m$ Navier–Stokes coefficient matrix and M the preconditioner:

$$H = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix}, \qquad \text{and} \quad M = \begin{bmatrix} M_F & B^T \\ 0 & -M_S \end{bmatrix},$$

with M_F and M_S scalable preconditioners for F and for the negative Schur complement $S = BF^{-1}B^T$, respectively. From now on, scalability will denote the property of a given preconditioner to produce roughly the same number of iterations as the mesh is refined. In other words, the eigenvalues of the preconditioned matrices will be uniformly bounded in h.

To analyze the eigenvalues of the preconditioned matrix $HM^{-1}\boldsymbol{w} = \lambda\boldsymbol{w}$, we assume that there exist LU factorizations of $M_F = L_F U_F$ and $M_S = L_S U_S$ and define

$$\Pi_L = \text{diag}(L_F, L_S), \qquad \text{and} \quad \Pi_U = \text{diag}(U_F, U_S).$$

Subsequently, the eigenvalue problem is equivalent to

$$\Pi_L^{-1} H \Pi_U^{-1} \boldsymbol{z} = \lambda \Pi_L^{-1} M \Pi_U^{-1} \boldsymbol{z}, \qquad \boldsymbol{z} = \Pi_U \boldsymbol{w}.$$

Defining $N = L_F^{-1} B^T U_S^{-1}$, $R = L_S^{-1} B U_F^{-1}$ and exploiting the blocks, we obtain

$$\begin{bmatrix} F_P & N \\ R & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix} = \lambda \begin{bmatrix} I_n & N \\ 0 & -I_m \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix} \tag{11}$$

where $F_P = L_F^{-1} F U_F^{-1}$ is similar to the preconditioned $(1,1)$ block $M_F^{-1}F$.

Matrices R and N satisfies:

$$RN = L_S^{-1} B U_F^{-1} L_F^{-1} B^T U_S^{-1} = L_S^{-1} B F_P^{-1} B^T U_S^{-1} = L_S^{-1} \hat{S} U_S^{-1} \equiv S_P,$$

where $\hat{S} = B^T M_F^{-1} B$ is the **exact** (minus) Schur complement related to the matrix

$$\begin{bmatrix} M_F & B^T \\ B & 0 \end{bmatrix},$$

and hence S_P represents this Schur complement, preconditioned by the spectrally equivalent matrix M_S. Observing that the matrix on the right in (11) is involutory we rewrite (11) as

$$\begin{bmatrix} F_P & N \\ R & 0 \end{bmatrix} \begin{bmatrix} I_n & N \\ 0 & -I_m \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} = \begin{bmatrix} I_n & N \\ 0 & -I_m \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix}, \tag{12}$$

which finally reads

$$H_{BTP}\boldsymbol{u} = \lambda \boldsymbol{u} \qquad \text{or} \qquad \begin{bmatrix} F_P & (F_P - I_n)N \\ R & RN \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix}. \tag{13}$$

Writing now $E_F = F_P - I_n$ and $E_S = S_P - I_m$, (13) rewrites as

$$\left(\begin{bmatrix} I_n & 0 \\ R & I_m \end{bmatrix} + \begin{bmatrix} E_F & 0 \\ 0 & E_S \end{bmatrix} \begin{bmatrix} I_n & N \\ 0 & I_m \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix}. \tag{14}$$

We have shown that the BTP-preconditioned matrix is similar to a matrix that can be written as one with all eigenvalues at one plus another one whose norm is bounded by the norms of E_F and E_S.

4.2. Inexact Constraint Preconditioner

Using the previous definition, we can define the Inexact Constraint Preconditioner ICP as

$$M_{ICP} = \begin{bmatrix} M_F & B^T \\ B & BM_F^{-1}B^T - M_S \end{bmatrix},$$

The adjective inexact arises when the exact Schur complement $\hat{S} = BM_F^{-1}B^T$ is replaced by an approximation, M_S, as it is in this case. This implies that the $(2,2)$ block of M_{ICP} is no longer zero, as it is in the Constraint Preconditioner. As before, pre- and post- multiplying by Π_L and Π_U yields an eigenvalue problem equivalent to $H\boldsymbol{w} = \lambda M_{ICP}\boldsymbol{w}$:

$$\begin{bmatrix} F_P & N \\ R & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix} = \lambda \begin{bmatrix} I_n & N \\ R & S_P - I_m \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix} \tag{15}$$

Matrix on the right in (15) is easily decomposed, recalling that $S_P = RN$, as

$$\begin{bmatrix} I_n & N \\ R & RN - I_m \end{bmatrix} = \begin{bmatrix} I_n & 0 \\ R & I_m \end{bmatrix} \begin{bmatrix} I_n & N \\ 0 & -I_m \end{bmatrix}$$

It turns out that $M_{ICP}^{-1}H$ is similar to the matrix

$$\begin{bmatrix} I_n & 0 \\ R & I_m \end{bmatrix}^{-1} \overbrace{\begin{bmatrix} F_P & N \\ R & 0 \end{bmatrix} \begin{bmatrix} I_n & N \\ 0 & -I_m \end{bmatrix}}^{H_{BTP}} = \begin{bmatrix} I_n & 0 \\ R & I_m \end{bmatrix}^{-1} \left(\begin{bmatrix} I_n & 0 \\ R & I_m \end{bmatrix} + \begin{bmatrix} E_F & E_F N \\ 0 & E_S \end{bmatrix} \right) \tag{16}$$

$$= \begin{bmatrix} I_n & 0 \\ 0 & I_m \end{bmatrix} + \begin{bmatrix} I_n & 0 \\ -R & I_m \end{bmatrix} \begin{bmatrix} E_F & 0 \\ 0 & E_S \end{bmatrix} \begin{bmatrix} I_n & N \\ 0 & I_m \end{bmatrix}, \tag{17}$$

in view of (14). We have shown that the ICP-preconditioned matrix is similar to a matrix that can be written as the identity matrix of order $n + m$ plus a matrix whose norm is bounded by the norms of E_F and E_S.

The results developed for ICP and BTP preconditioned matrices show that unit eigenvalues are perturbed by the presence of error matrices E_F and E_S whose norms, in the case of scalable preconditioners for the blocks, swill be small and independent of the mesh discretization parameter h. This is a pleasant property, which, in many cases, is responsible of fast convergence of the GMRES method. However, it is well known [20] that a clustering of the eigenvalues around the unit value is not sufficient to guarantee fast convergence of the GMRES method due to the possible ill-conditioning of the matrix of the eigenvectors. Analyses based on the field of values of the preconditioned matrix have been carried out for simplified preconditioners in order to overcome this theoretical problem. See, e.g., [21,22].

4.3. Relaxation

In case the spectral regions of the preconditioned $(1,1)$ block and the preconditioned Schur complement are not perfectly overlapped, the influence of a relaxation parameter ω has been investigated in [7], where, however, the two block matrices are both SPD. In short, the relaxation parameter acts on the $(2,2)$ block of the Inexact Constraint Preconditioner, as

$$M_{ICP}(\omega) = \begin{bmatrix} M_F & B^T \\ B & BM_F^{-1}B^T - \omega M_S \end{bmatrix},$$

with ω to be chosen so that the smallest of the two previously mentioned spectral regions is included in the other one.

5. Preconditioners for the Blocks

Any block preconditioner for a saddle-point linear system, like the one that we considered, can only yield scalable results if the preconditioners used to approximate the $(1,1)$ block and the Schur complement are scalable. Therefore, the main task is to find such suitable preconditioners for the two blocks involved.

5.1. $(1,1)-$Block Preconditioner

It is well known that Multigrid methods allow for obtaining scalable results for convection-diffusion problems. Our $(1,1)-$block indeed represents a discretized convection-diffusion operator and, therefore, we expect to obtain scalable results while using a suitable Multigrid scheme. An accurate analysis of the scalability provided by Multigrid can be found, for instance, in [23,24].

The application of a Multigrid preconditioner is particularly easy in the case of diffusion-dominated problems, since even the very simple damped Jacobi smoother is able to produce optimal results. However, in the case of convection-dominated problems, the choice of the smoother is fundamental for obtaining satisfactory results. In these situations, it is necessary to use a more sophisticated smoother, like the Gauss–Seidel method, coupled with an adequate choice of the pattern used to apply it; this pattern should try to follow the convective flow as much as possible. In the case of complicated recirculating flows, it is not possible to generate a pattern that is able to do this and so different strategies must be employed. In [2], the suggestion is to use multiple iterations of the Gauss-Seidel smoother, each one performed following a simple pattern. In particular, if the flow is recirculating and therefore has components both negative and positive in directions x and y, then there should be four patterns and they should sweep through the domain in both directions, going back and forth. Hence, one pattern will evolve in the x direction, from the left to the right, another one from right to left. The same holds for the y direction, yielding a total of four different smoothing patterns. An alternative using only one pattern in every direction is possible: this approach will be cheaper, but

a single iteration will be less accurate. This option has been used, for instance, in [13] when dealing with convection-diffusion problems.

One thing to remember is that, when multiple patterns are used for every smoothing iteration, they should be applied in opposite order in the pre- and post-smoothing phases. If in the pre-smoothing step, we apply first pattern A and then pattern B, then in the post smoothing phase we should apply first pattern B and then A.

The drawback of using multiple patterns is that some parts of the flow, which only have component of the velocity in one direction, do not need smoothing in the other direction. Thus, a lot of time is lost smoothing in directions that are not necessary. This is not a problem for convergence, since this excessive smoothing does not worsen the solution, but it is a problem for the computational time. Moreover, this approach does not exploit the separability of the $(1,1)-$block: indeed, the convection-diffusion operator consists of a block-diagonal matrix, with two identical blocks, one for direction x and one for y. This structure of the matrix can be exploited to obtain a more efficient implementation: indeed, when applying the preconditioner to a vector r, we can split this vector into r_x and r_y and apply to each of these components just the preconditioner for one of the diagonal blocks of matrix F. In this way, the dimensions of the matrices used are halved and, more importantly, we can choose to perform the smoothing only in the direction that we are considering. This means that when we apply the preconditioner to r_x, we use Gauss–Seidel with a pattern that only evolves in the x direction. We do not expect more accurate solutions with this method, since applying extra smoothing does not create such a problem, but we expect to reduce the computational time required.

Two approaches are possible when applying one of the patterns in the smoothing phase. We can divide the pattern in groups and apply the smoother to all the nodes of every group simultaneously; for instance, if the pattern to be applied evolves in the x direction, then each group would be one of the vertical lines of nodes. In this way, the smoother updates all of the values for these nodes at once, which is the best approach possible. However, in this way, the linear system to be solved inside Gauss–Seidel is block triangular and not triangular. Alternatively, we can apply the smoother to every node individually; in this way, the system to be solved is triangular, but along every line, some nodes are updated before and some others later. We used the second approach, as the results were very similar, but the computational time was lower.

In the numerical experiments, we tested four different approaches, which are summarized in the following:

- *Jacobi*, which identifies the Multigrid scheme using the simple damped Jacobi smoother.
- *simple-GS*, which indicates the use of Gauss-Seidel with lexicographic ordering.
- *2dir-GS*, which identifies the Gauss-Seidel smoother applied with two patterns, one for every direction.
- *split-GS*, which indicates the new approach that exploits the structure of the $(1,1)-$block.

The patterns that are used in the last two approaches are shown in Figure 2 for a small grid.

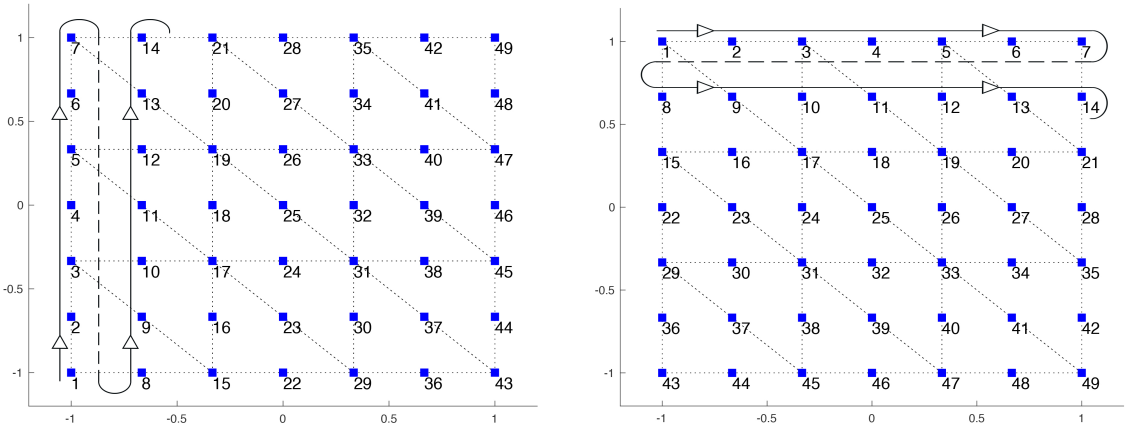


Figure 2. Ordering of the nodes for the smoothing.

Besides the smoother, there are other aspects of the Multigrid preconditioner that must be treated with care when dealing with convection-dominated problems. Stabilization needs to be taken into account: when passing from one grid to the next, the matrix to be used in the following grid must be the exact matrix, with the accurate stabilization; the Galerkin coarse grid operator should be avoided, because the coarser grids need stabilization, while the fine ones may not need it. In our experiments, we used streamline diffusion stabilization.

The restriction and prolongation operators can be taken to be the natural ones, i.e., the ones obtained exploiting the values of the basis functions over the nodes of the discretizations. Finally, we need to choose the cycle to be used: a V-cycle would be cheaper, but it may need more iterations to obtain the same accuracy. We chose to use a W-cycle instead, which requires more computational effort, but more closely clusters the eigenvalues around one. However, as we shall see in the next section, the Multigrid method is also used by the Schur complement preconditioner. In that case, we made a different choice of cycle.

5.2. Schur Complement Preconditioner

Let us now focus on finding a scalable preconditioner M_S for the Schur complement. In the case of diffusion-dominated problems, then the mass matrix or even a diagonal approximation of it can produce scalable results. Indeed, for the Stokes problem, which represents the limit case where convection is not present, the pressure mass matrix is spectrally equivalent to the Schur complement.

The difficult challenge is to find a scalable preconditioner in the case of convection-dominated problems. The first approaches used the mass matrix that was scaled by the viscosity. This choice, although very simple, does not yield scalable results in general. Therefore, more complicated preconditioners are necessary. We now summarize some of these alternatives.

5.2.1. PCD Preconditioner

The Pressure convection-diffusion (PCD) approach can be intuitively justified when considering the operator that the Schur complement represents: matrix $S = BF^{-1}B^T$ is the discrete counterpart of the continuous operator $\mathrm{div}(-\nu\Delta + \mathbf{w}\cdot\nabla)^{-1}\nabla$, where $\mathbf{w}$ represents the local wind (given by the solution of the previous iteration in the case of the Oseen problem). We can think of a preconditioner for the Schur complement as an approximation of the discrete version of the inverse of this operator: then, calling F_p the convection-diffusion operator built in the pressure space and L_p the corresponding Laplacian, we can assume that matrix $F_pL_p^{-1}$ is an approximation for S^{-1}. However, in the case of diffusion-dominated problems, F_p and L_p tend to coincide, yielding the identity matrix as

preconditioner. Premultiplying by Q_p, the pressure mass matrix, we can assure that in this extreme case the preconditioner goes back to the known optimal alternative.

This preconditioner was proposed and derived formally in [4] and studied thoroughly in [25]. It provides a significant improvement with respect to the scaled mass matrix, keeping the computational cost low: matrices F_p and L_p are indeed smaller than F and L, making this preconditioner particularly efficient. However, the rate of convergence seems to suffer when ν gets close to zero, even for simple problems. Moreover, matrix F_p is not used in the original problem and, thus, needs to be built at every Picard iteration; the proper boundary conditions to apply to it are not straightforward to choose and they do not necessarily coincide with the conditions applied to the original problem. An analysis of this issue can be found in [14].

5.2.2. BFBt Preconditioner

A different preconditioner can be derived in a more algebraic way: in [9], it is proved that, under the assumption that range$(B^T) \subset$ range$(F^{-1}B^T)$, the exact inverse of the Schur complement is given by matrix

$$M_S = (BB^T)^{-1}BFB^T(BB^T)^{-1}.$$

Hence, we can take M_S as a preconditioner for the Schur complement, even if the assumption does not hold in general. Further analysis showed that an improvement can be obtained when considering the scaled version

$$M_S = (BQ_v^{-1}B^T)^{-1}BQ_v^{-1}FQ_v^{-1}B^T(BQ_v^{-1}B^T)^{-1},$$

which, in the case of continuous pressure elements, simplifies to

$$M_S = L_p^{-1}BQ_v^{-1}FQ_v^{-1}B^TL_p^{-1}, \tag{18}$$

where Q_v is the velocity mass matrix, or an approximation of it. This preconditioner is clearly more expensive than the PCD, since it involves two applications of L_p^{-1} and a product with matrix F instead of F_p. However, most of the matrices used are already available and the additional boundary conditions to apply to L_p represent a less critical problem. Dependence on the mesh size and on the viscosity is observed, especially for complicated flows.

5.2.3. Commuted BFBt Preconditioner

An improvement to the BFBt preconditioner was proposed in [14], where they analyzed the continuous counterpart of matrix (18) and suggested to commute some of the operators involved, yielding the preconditioner

$$M_S = Q_p^{-1}BL^{-1}FL^{-1}B^TQ_p^{-1} \tag{19}$$

where L is the diffusive part of matrix F. This alternative is even more expensive than the previous one, since matrix L is larger than L_p. On the positive side, all of the matrices are already available and there are no new boundary conditions to apply. Moreover, the inversion of matrix L can be performed with the same Multigrid technique used to precondition matrix F.

This last alternative is the one that has showed the best results in term of scalability, even if some dependence on the mesh size can still be observed for very low viscosities and complicated flows. It is also worth pointing out that, in the case of diffusion-dominated problems, this approach behaves decisively worse than PCD.

5.2.4. Augmented Lagrangian Approach

We briefly sketch a completely different approach from the ones previously presented: suppose modifying the $(1,1)-$block of the original saddle-point linear system, so that the matrix becomes

$$\begin{bmatrix} F + \gamma B^T W^{-1} B & B^T \\ B & 0 \end{bmatrix},$$

where $\gamma > 0$ is a parameter and W is positive definite. The solution is not changed, but it is now possible to simplify the Schur complement and find a very efficient preconditioner. As shown in [1], if W is taken to be the pressure mass matrix, an optimal choice for M_S is simply $(\nu + \gamma)Q_p^{-1}$. However, this great efficiency in the Schur complement preconditioner is balanced by the need for a much more complicated Multigrid scheme for the new $(1,1)-$block. In [3], this approach was generalized to the three dimensional case and it was also pointed out that the highly specialized Multigrid scheme used depends on the choice of the discretization and it may not be possible to use this method with every such choice.

In our numerical experiments, we chose to use preconditioner (19), since it is the one with the best chance of providing scalable results for a recirculating flow and since it does not involve matrices to be computed separately. Moreover, it exploits the same Multigrid that we used for the $(1,1)$ block, which means that any improvement that we make there also has an impact on the Schur complement.

Some comments must be made regarding the Multigrid used to invert matrix L. We used the same strategy used for matrix F, with the following differences: since no convection is present, we found out that using the V cycle is sufficient to provide good results; we performed a larger number of cycles, since, in this case, we would like to obtain the exact inverse of L and not just a preconditioner. We maintained the approach of splitting matrix L, since this allows for saving computational time.

6. Low-Rank Update of the Block Preconditioner

6.1. Left Preconditioning

Consider the generic linear system $A\boldsymbol{x} = \boldsymbol{b}$. Following [26], given an initial preconditioner $P^{(0)} \approx A^{-1}$ and a tall, full-rank, $n \times p$ matrix V, the Broyden-tuning preconditioner is defined as

$$P = P^{(0)} - (P^{(0)}AV - V)\left(V^T P^{(0)} AV\right)^{-1} V^T P^{(0)} \tag{20}$$

with the property that

$$PAV = V. \tag{21}$$

Hence, the preconditioned matrix has the eigenvalue 1 with multiplicity (at least) p. The best choice of the columns of V is represented by the eigenvectors of $P^{(0)}A$ corresponding to the outlier eigenvalues. Let us assume then that the columns of V are approximate eigenvectors of the initially preconditioned matrix:

$$P^{(0)}AV = V\Lambda + E,$$

with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ and $\|E\|$ is small enough. The eigenvalues are decreasingly ordered with respect to the function $f(\lambda) = |\lambda - 1|$.

Neglecting matrix E we can simplify the low-rank preconditioner as

$$\begin{aligned} P &= P^{(0)} - (P^{(0)}AV - V)\left(V^T P^{(0)} AV\right)^{-1} V^T P^{(0)} \\ &= P^{(0)} - (V\Lambda - V)\left(V^T V\Lambda\right)^{-1} V^T P^{(0)} \\ &= P^{(0)} - V(\Lambda - I_p)\Lambda^{-1}(V^T V)^{-1} V^T P^{(0)} \\ &= P^{(0)} - V(I_p - \Lambda^{-1})(V^T V)^{-1} V^T P^{(0)} \\ &= P^{(0)} - VHV^T P^{(0)}, \end{aligned} \tag{22}$$

with small matrix $H = (I_p - \Lambda^{-1})(V^T V)^{-1}$ to be built once and for all before starting of the iteration process.

6.2. Right Preconditioning

The Broyden preconditioner inherits the tuning condition (21) as a reformulation of the secant equation, which only holds for left preconditioning. However, it can be easily reformulated to handle the case of right preconditioning, as follows

$$P_R = P^{(0)} - (P^{(0)}V - A^{-1}V)\left(V^T P^{(0)} V\right)^{-1} V^T P^{(0)}$$

satisfying now

$$AP_R V = V.$$

The unpleasant presence of A^{-1} in this expression can be eliminated reasoning as before and assuming that now $AP^{(0)}V = V\Lambda + E_R$ and, therefore, $A^{-1}V \approx P^{(0)}V\Lambda^{-1}$, so that we can define an approximate, but more effective right preconditioner as

$$\begin{aligned} P_R &= P^{(0)} - (P^{(0)}V - P^{(0)}V\Lambda^{-1})\left(V^T P^{(0)} V\right)^{-1} V^T P^{(0)} \\ &= P^{(0)} - WHV^T P^{(0)}, \qquad \text{with} \qquad W = P^{(0)}V, \quad H = (I_p - \Lambda^{-1})(V^T W)^{-1} \end{aligned}$$

Differently from the left preconditioning the preprocessing stage now requires p applications of the preconditioner $P^{(0)}$ to form matrix W.

6.3. Avoiding Complex Eigenpairs

One possible obstacle to this approach is the fact that both V and Λ are likely to be complex. This is solved by constructing a real invariant subspace of the same size, as described in Algorithm 1.

Algorithm 1 Constructing a real invariant subspace for the preconditioned matrix.

$\Gamma = \Lambda$
$j = 1$
while $j < p$ **do**
  if $|\mathrm{Im}(\lambda_j)| > 10^{-3}$ **then**
    $a = \mathrm{Re}(\lambda_j); b = \mathrm{Im}(\lambda_j)$
    $\Gamma(j:j+1, j:j+1) = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$
    $Z(:, j) = \mathrm{Re}(V(:, j));$
    $Z(:, j+1) = \mathrm{Im}(V(:, j));$
    $j = j + 2;$
  else
    $Z(:, j) = V(:, j);$
    $j = j + 1;$
  end if
end while
if $j == p$ **then**
  $Z(:, j) = V(:, j);$
end if

By construction matrices Z and Γ (which is no longer diagonal) approximately satisfy $P^{(0)}AZ = Z\Gamma$ and the low-rank left preconditioner is defined consequently as

$$P = P^{(0)} - Z(I_p - \Gamma^{-1})(Z^TZ)^{-1}Z^TP^{(0)} = P^{(0)} - ZHZ^TP^{(0)}, \qquad H = (I_p - \Gamma^{-1})(Z^TZ)^{-1}.$$

Analogously the right preconditioner is modified with Z, Γ in place of V, Λ.

6.4. Application of the Low-Rank Approach to the Blocks

Starting from our block preconditioners we can apply the low-rank correction either to the (1,1) block only or also to the Schur complement preconditioner. We consider then two cases, both in the framework of the left preconditioning:

1. Let $P_F^{(0)}$ be the (inverse of the) multigrid preconditioner for the $(1,1)$ block and consider the left preconditioner (22). To compute the relevant eigenpairs we run the nonrestarted Arnoldi method with $m = 50$ as the subspace size for the preconditioned matrix $P_F^{(0)}F$. Then we selected all the p eigenvalues satisfying $f(\lambda) \equiv |\lambda - 1| > 1$ and the corresponding eigenvectors as columns of V_F to form the updated preconditioner for the $(1,1)$ block as

$$P_F = P_F^{(0)} - V_F H_F V_F^T P_F^{(0)}.$$

2. In addition to the previous, define $P_S^{(0)}$ the (inverse of the) Schur complement preconditioner to be used in (22) and run the nonrestarted Arnoldi method for the preconditioned matrix $P_S S_{LR}$, where S_{LR} is the Schur complement of the block preconditioner defined as

$$S_{LR} = B\left(P_F^{(0)} - VHV^TP_F^{(0)}\right)B^T.$$

   Again, we consider all of the eigenvalues satisfying $f(\lambda) > 1$ and form the updated Schur complement preconditioner as

$$P_S = P_S^{(0)} - V_S H_S V_S^T P_S^{(0)}$$

7. Numerical Results

In this Section, 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 show convergence profiles of GMRES.

7.1. Model Problem

The model problem that is used in this work is the well-known lid-driven cavity problem; a thorough description of this problem can be found in [27]. 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 two-dimensional square $[-1,1] \times [-1,1]$. The equations, together with the boundary conditions, are

$$\begin{cases} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{0} \\ \nabla\cdot\mathbf{u} = 0 \\ \mathbf{u} = 0, \quad \mathbf{x} \in \{-1,1\} \times [-1,1] \cup [-1,1] \times \{-1\} \\ u_y = 0, \quad \mathbf{x} \in [-1,1] \times \{1\} \\ u_x = 1, \quad \mathbf{x} \in [-1,1] \times \{1\} \end{cases} . \tag{23}$$

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 3 and 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.

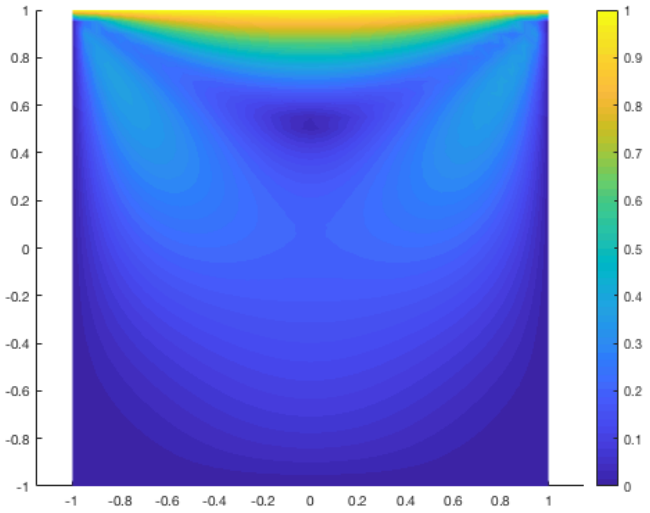


Figure 3. Velocity magnitude for the Stokes problem.

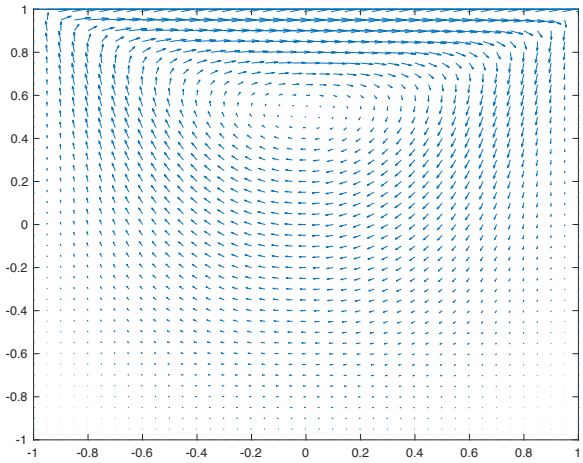


Figure 4. Velocity direction for the Stokes problem.

7.2. Problem Description

The matrices used for the numerical tests are generated using values of n from 10 to 320. In particular, Table 1 illustrates all of the matrices used, showing their dimension and number of nonzero entries, while Figure 5 shows the sparsity pattern the matrix M10.

Table 1. Properties of the matrices.

Matrix	Dimension	NNZ
M10	1003	14,280
M20	3803	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

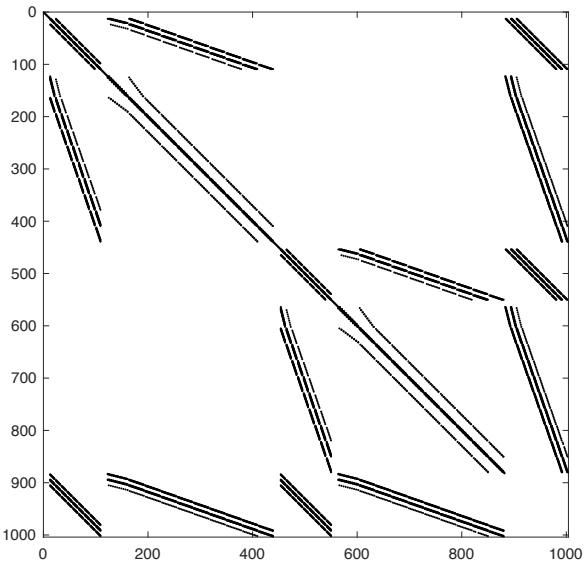


Figure 5. Sparsity pattern of matrix M10.

The number of nonzero entries per row vary between 14 and 18 and the density increases as n grows. The smaller matrices M10 and M20 will only be used 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 will regard the Navier–Stokes equations, for which we will be analyzing both the spectral properties and number of iterations required for the GMRES to converge. It is worth pointing out that the scalability of the results would not change using another Krylov subspace solver, e.g., BICGSTAB, since the preconditioner is derived based on the properties of the underlying problem, and not considering the specific solver used. The notation will be the following: α_F and β_F 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 always present).

7.3. Some Details on Implementation

All of the results have been obtained using codes written in Matlab on a laptop with equipped with an Intel processor i5-8350U quad core with 16GB RAM. 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^{-8} and without restart. Using the right-preconditioned version instead would not change the obtained scalability results. See the brief discussion in Section 7.8. All of the times reported are in seconds. The † symbol in the Tables will denote no convergence within 400 iterations.

7.4. Multigrid and BFBt

We now report the results that were obtained using the various Multigrid schemes and the BFBt preconditioner. These results are not scalable, due to the poor smoothers and the non-scalable implementation of the BFBt preconditioner. Hence, we will only show the results in terms of the iteration count. We will also show the eigenvalue distribution, which helps to understand the quality of the Multigrid smoother. The Multigrid parameters will be shown as $mV(pre, post)$, where m is the number of cycles used, V or W represent the type of cycle used, and pre and $post$ indicate the number of pre and post smoothing steps. The coarsest level for Multigrid in these tests is the matrix M10.

7.4.1. Jacobi/BFBt

The first combination involves a damped Jacobi smoother; the Multigrid used is $5V(1,1)$, while the BFBt preconditioner is implemented using an incomplete factorization.

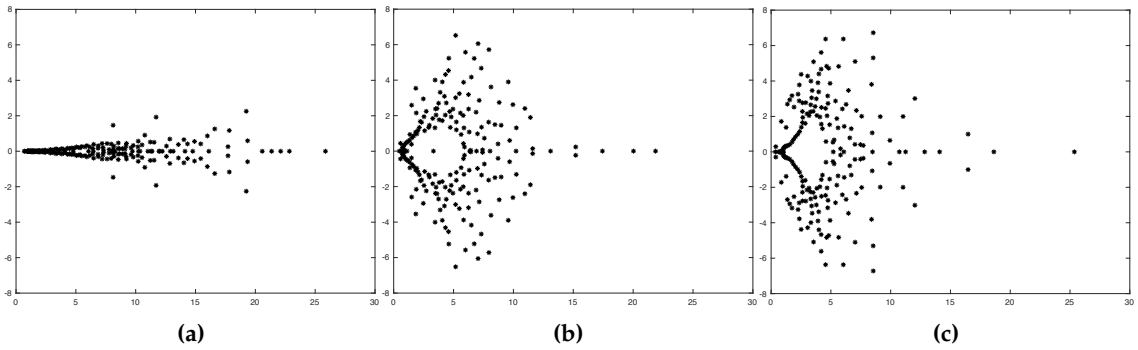
Table 2 reports the spectral properties of the preconditioned matrices.

Table 2. Spectral properties of the preconditioned matrices using Jacobi and BFBt.

Matrix	Viscosity	α_F	β_F	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_F is decreasing, β_F is increasing), even for the highest viscosity, which represents a problem that is similar to the Stokes one. The smallest eigenvalue α_F does not seem to be affected by the value of the viscosity, while the biggest one β_F 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 increasingly 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.



(a) **(b)** **(c)**

Figure 6. Spectra of the BFBt-preconditioned Schur complement for different values of viscosity. **(a)** $\nu = 0.1$; **(b)** $\nu = 0.01$; **(c)** $\nu = 0.005$.

Figure 6 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 as even using a different Multigrid. It is clear that, as the viscosity gets lower, the system becomes more asymmetric and, consequently, the imaginary part of the eigenvalues increases.

Table 3 shows the results obtained with Jacobi and BFB: even with very high viscosity, the preconditioner is not scalable; for low viscosity instead, it fails completely.

Table 3. Results using Jacobi and BFBt.

Matrix	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.005$
M40	67	110	†
M80	97	195	†
M160	141	336	†

7.4.2. Simple GS/BFBt

Using the simple Gauss–Seidel method as smoother is expected to improve the results, but it still should not be scalable, since the smoothing is performed following the original numbering of the nodes and not following the convective flow. The Multigrid is again $5V(1,1)$. Table 4 shows the spectral properties in this case.

Table 4. Spectral properties of the preconditioned matrices while using sGS and BFBt.

Matrix	Viscosity	α_F	β_F	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

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, even leading 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.

Table 5 shows the results in this case: they are still not scalable and again for low viscosities convergence is not achieved.

Table 5. Results using sGS and BFBt.

Matrix	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.005$
M40	64	107	†
M80	87	183	†
M160	135	334	†

7.4.3. Two-Direction GS/BFBt

The next combination uses a Gauss–Seidel smoother applied using two patterns that try to follow the convective flow. We also employed a W cycle, which means that the Multigrid used is now $2W(1,1)$. Table 6 shows the spectral properties.

Table 6. Spectral properties of the preconditioned matrices using 2dir GS and BFBt.

Matrix	Viscosity	α_F	β_F	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_F is very close to 1 and it does not suffer from refinement of the mesh or reduction in the viscosity. The spectral radius β_F 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 that is based on these phenomena).

Figure 7 shows the spectrum of the preconditioned $(1,1)$ block while using the three previous smoothers. It is clearly visible how only the last choice is able to cluster the eigenvalues around one.

Table 7 displays the results with this combination: they are again not scalable, but the behavior for the lowest viscosity is improved, thanks to the more efficient smoother.

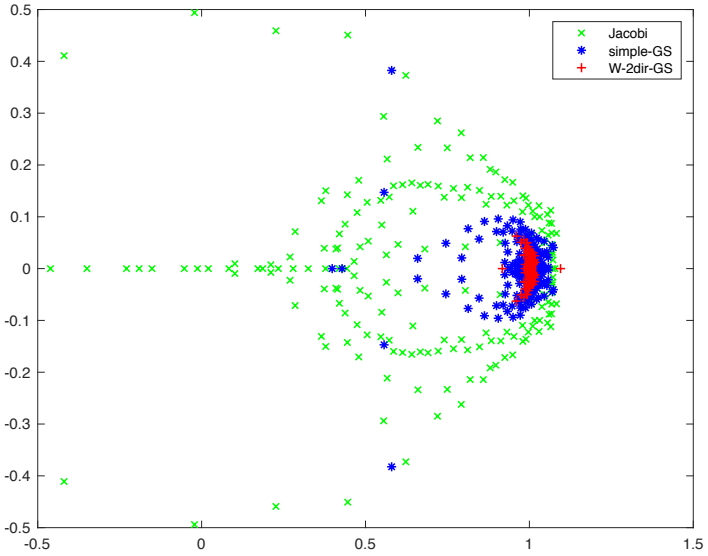


Figure 7. Spectra of various Multigrid schemes for the matrix M40 and $\nu = 0.005$.

Table 7. NS-W2dGS-BFBt results for various values of the relaxation parameter.

ω	M40			M80			M160		
	0.1	0.01	0.005	0.1	0.01	0.005	0.1	0.01	0.005
1	64	107	118	88	181	253	136	330	†
0.5	62	104	115	86	177	249	135	328	†
0.25	61	102	111	85	174	246	133	325	†
0.1	59	100	112	82	171	242	129	321	†
0.05	59	113	134	81	174	265	128	319	†

In Table 7, we also show the effect of the relaxation parameter ω, as proposed in [7], and briefly described at the end of Section 4, with the aim of improving the condition number of the preconditioned matrices. Setting $\omega < 1$ provides a slight decrease in the number of iterations, and this is due to the spectra of the preconditioned blocks, which partially overlap. Because of these (not completely satisfactory) preliminary results, the relaxation parameter will not be employed in the next results.

7.5. Multigrid and BFBt-c

To obtain scalable results, we used the BFBt-c preconditioner, which was also implemented using a Multigrid scheme. In the following tables, we also show the true residual `tres` of the solution computed by GMRES ($\|M\mathbf{x} - \mathbf{f}\|/\|\mathbf{f}\|$). This datum is useful to understand the quality of the preconditioner: indeed, using left preconditioning, the exit test of GMRES is performed with the residual of the preconditioned system, which may differ substantially from the residual of the actual system. Moreover, we also show the computational time, which includes the time to build the preconditioner, apply it,s and solve the linear system with GMRES.

The coarsest level for Multigrid in these tests is the matrix M20.

7.5.1. Jacobi/BFBt-c

The first test involves the Jacobi smoother; the Multigrid in this case is $2V(2,2)$ for the $(1,1)$ block and $5V(2,2)$ for the BFBt-c preconditioner.

The results are shown in Table 8: we can see that, for high viscosities, the results are scalable, but the preconditioner fails when using low viscosities, providing erratic iteration numbers.

Table 8. Results using Jacobi and BFBt-c.

Matrix	$\nu = 0.1$			$\nu = 0.01$			$\nu = 0.005$		
	iter	time	tres	iter	time	tres	iter	time	tres
M20	45	1.76	2.40E-3	59	2.29	3.33E-3	99	3.83	3.56E-2
M40	49	3.49	3.30E-3	65	4.53	3.90E-3	215	15.77	4.06E-2
M80	53	8.76	2.20E-3	69	11.46	3.50E-3	232	39.45	1.22E-2
M160	55	38.46	1.11E-3	71	50.04	2.10E-3	113	82.38	4.60E-3
M320	55	162.95	4.69E-4	73	215.95	1.10E-3	121	420.06	1.50E-3

7.5.2. Simple GS/BFBt-c

We then used the simple Gauss-Seidel smoother with the BFBt-c preconditioner, with the same parameters as before. The results, shown in Table 9, show an improved behavior, with good results also in the case of $\nu = 0.005$. However, for the lowest $\nu = 10^{-3}$ viscosity, the non-scalability remains.

Table 9. Results using sGS and BFBt-c.

Matrix	$\nu = 0.1$			$\nu = 0.01$			$\nu = 0.005$		
	iter	time	tres	iter	time	tres	iter	time	tres
M20	43	1.70	2.00E-4	56	2.21	8.85E-4	55	2.18	1.78E-2
M40	47	3.49	1.32E-4	60	4.47	6.75E-4	78	5.75	1.25E-2
M80	50	8.87	3.43E-5	63	11.20	1.66E-4	91	16.23	1.60E-3
M160	52	39.91	3.49E-5	62	47.67	1.32E-5	103	81.14	5.58E-4
M320	53	171.20	9.57E-5	65	211.05	3.71E-5	111	370.35	2.74E-4

7.6. Handling the Low Viscosity Case. Two-Direction GS/BFBt-c

The next test is performed using the 2dir GS smoother and the BFBt-c preconditioner. The Multigrid for the $(1,1)$ block is now $2W(2,2)$.

In Figure 8 (left), we compare the spectrum of BFBt and BFBt-c, while, in Figure 8 (right), we show the spectrum of the $(1,1)$ block and the one of the BFBt-c preconditioned Schur complement. The plot reveals the clear superiority of the BFBt-c preconditioner, which provides a better clustering of the eigenvalues.

To better follow the circulant flow with so high Reynolds number, we choose to select as the coarsest mesh M20 instead of M10. As a first consequence, the cost per iteration is expected to increase, mainly for the smallest problems.

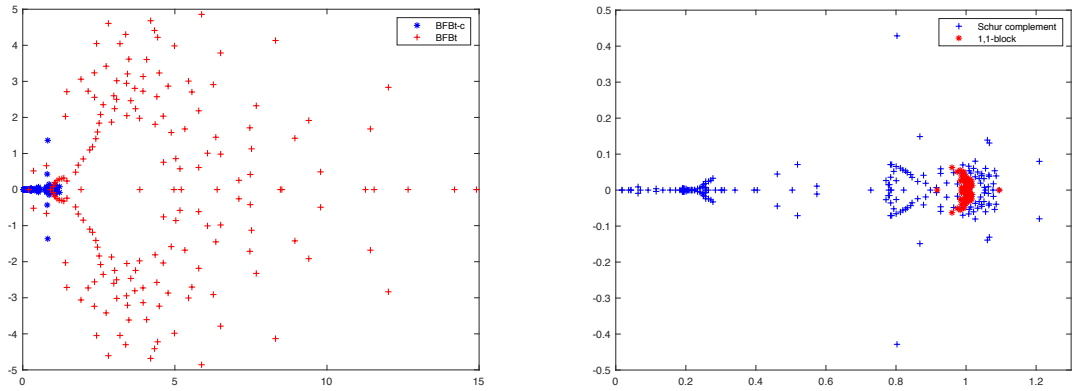


Figure 8. On the left, the spectra of the preconditioned Schur complement for BFBt and BFBt-c preconditioners. On the right the spectra of the preconditioned $(1,1)$ block and the Schur complement using BFBT-c. Plots referring to matrix M40 and $\nu = 0.005$.

Table 10 shows the results while using this combination. The preconditioners are scalable for all viscosities, confirming the promising impression that we got from the spectral information. However, the true residual of the final solution is a couple of orders of magnitude larger than the original tolerance of GMRES.

Table 10. Results using 2dir-GS and BFBt-c.

Matrix	$\nu = 0.1$			$\nu = 0.01$			$\nu = 0.005$			$\nu = 0.001$		
	iter	time	tres	iter	time	tres	iter	time	tres	iter	time	tres
M40	46	14.08	1.8e-5	52	21.14	4.6e-5	60	18.53	3.0e-4	80	24.49	1.5e-2
M80	49	33.99	1.5e-5	55	49.64	2.3e-5	65	42.35	7.6e-5	85	56.16	6.9e-3
M160	52	125.01	8.9e-6	53	127.17	1.4e-5	66	158.49	2.1e-5	93	224.67	9.3e-4
M320	52	589.25	1.1e-6	52	614.96	1.1e-5	66	651.42	1.3e-5	99	1128.95	1.4e-4

Table 11. Results using ICP with split-GS and M20 as the coarsest mesh.

Matrix	$\nu = 0.1$			$\nu = 0.01$			$\nu = 0.005$			$\nu = 0.001$		
	iter	time	tres	iter	time	tres	iter	time	tres	iter	time	tres
M40	47	13.13	1.6e-4	53	14.76	1.5e-4	62	16.97	4.7e-4	82	22.31	1.5e-2
M80	50	27.43	1.7e-4	52	28.10	8.2e-5	64	35.47	5.2e-4	89	49.34	5.5e-3
M160	52	75.46	3.2e-4	56	81.91	5.7e-6	67	98.77	1.4e-4	94	140.11	6.4e-3
M320	55	312.79	3.3e-4	58	332.21	1.5e-5	67	384.49	1.2e-4	99	697.55	1.9e-3

To try saving some of the added computational time, we employed strategy *split-GS*, which is supposed to be cheaper than *2dir-GS*. We show the results in Table 11: when compared with Table 10, these results indeed show a lower computational time, revealing, as expected, that *split-GS* is a cheap variant of *2dir-GS*. However, it also introduces some instabilities that prevent the reduction of the true residual in some cases.

If we compare Tables 8–11, we can see that, for high viscosities, the results are scalable using any method. In this situation, the most efficient approach in terms of computational time is the one that involves the Jacobi smoother, since the convection is not sufficiently strong to justify the use of a complicated flow-following technique. However, when it comes to low viscosities, the Gauss–Seidel smoother with proper ordering is the only strategy that manages to produce scalable results.

We also report the results while using a Block Triangular Preconditioner, as shown in Table 12. The higher number of iterations needed is balanced by the reduced cost per iteration, producing a final computational time that is better than in the previous case. However, the true residual gets even higher in some cases.

Table 12. Results using BTP with split-GS and M20 as the coarsest mesh.

Matrix	$\nu = 0.1$			$\nu = 0.01$			$\nu = 0.005$			$\nu = 0.001$		
	iter	time	tres	iter	time	tres	iter	time	tres	iter	time	tres
M40	52	11.66	1.20E-3	55	11.93	1.62E-4	71	15.37	4.43E-4	94	20.29	2.10E-3
M80	55	24.46	5.20E-3	57	22.98	2.31E-4	72	28.83	6.64E-5	102	41.56	1.50E-3
M160	58	63.11	1.55E-3	59	63.81	4.29E-4	72	78.87	1.57E-4	106	118.64	1.60E-3
M320	62	281.67	5.29E-2	59	273.62	1.20E-3	67	301.17	3.90E-4	108	493.86	1.66E-4

The low-rank update, whose effects will be described next, will mitigate this effect.

7.7. Handling the Low Viscosity Case. Preconditioner Update by Low-Rank Matrices

We have obtained a scalable preconditioner for all viscosities considered using, as the coarsest level, the matrix M20 instead of M10, which implies a high computational cost at each application of the Multigrid. This choice is necessary, since using matrix M10 produces the spectral distribution that is shown in Table 13, where the Multigrid is completely unable to properly precondition the $(1,1)$ block.

Table 13. Spectral properties of the preconditioned matrices with viscosity 0.001 using the coarsest matrix M10.

Matrix	# outliers in the (1,1) block	$\max \lvert\lambda(M_F^{-1}F) - 1\rvert$	# outliers in the Schur complement	$\max \lvert\lambda(M_S^{-1}S) - 1\rvert$
M40	11	22.8	4	2.9
M80	10	511.9	7	3.7
M160	10	1.2×10^5	10	3.4

The situation that is depicted by Table 13 is a promising situation that suggests the use of the low-rank update: the number of outlier for the preconditioned F block is relatively small and does not roughly change with the meshsize. On the contrary, updating the preconditioner for the Schur complement block does not seem to be advisable, as the eigenvalues of this block are not much far away from 1.

The results where the split-GS based preconditioner is used in combination with the low-rank update for the $(1,1)$ block, as described in Section 6, are reported in Table 14. We employed 50 non-restarted Arnoldi iterations to assess the corresponding eigenvectors (as they are well separated, Arnoldi convergence reveals very fast). The preprocessing time that is related to this task is reported in the Table as `prep`. The results using the Constraint Preconditioner and the BTP are summarized in Table 14.

The results show that this approach yields similar results as in the previous Tables, with two exceptions: first, the fact that we selected M10 as the coarsest mesh provided a notable reduction of the CPU time per iterations. Second, the true norm of the residual is much lower, showing a closer relation between the residual provided by the GMRES method and the true residual, and suggesting the better conditioning of the preconditioner. Moreover, the preprocessing time to assess dominant eigenvectors is around 10% of the overall CPU time. Scalability with respect to h is almost perfect, the number of iterations, on the average, being slightly increased with respect to starting from M20 mesh with no update.

Table 14. Results using split-GS and low-rank update for $\nu = 0.001$ with ICP (left) and BTP (right). The coarsest mesh is M10.

Matrix	Inexact Constraint Preconditioner					Block Triangular Preconditioner				
	Iter	Time			tres	Iter	Time			tres
		prep	GMRES	total			prep	GMRES	total	
M20	115	0.51	4.85	5.36	6.60E-3	93	0.55	5.23	5.78	9.80E-3
M40	86	1.38	9.29	10.67	2.40E-3	106	1.27	8.20	9.47	3.85E-4
M80	92	4.15	31.20	35.35	3.32E-4	112	3.57	25.58	29.15	4.27E-4
M160	98	11.87	127.11	138.98	4.75E-4	113	11.92	96.51	108.43	2.00E-4
M320	118	59.26	658.55	707.81	5.90E-5	117	48.97	488.38	537.35	5.89E-5

7.8. Comparisons with Right-Preconditioned GMRES

As anticipated, the scalability results obtained so far with the left-preconditioned GMRES would also be confirmed by the right-preconditioned GMRES implementation. It is well-known that, in this last case, the exit test is on the true residual (instead of the preconditioned residual). However this advantage is in some instances purely theoretical as the exact residual may be much different from the computed residual due to ill-conditioning of the systems matrices. We report in Table 15 the results for $\nu = 0.001$ and both BTP and ICP approaches while using the right-preconditioned GMRES, to be compared with the last columns in Tables 11 and 12 and with Table 14. The results confirm the optimal scalability of the proposed preconditioner.

Table 15. Results using right-preconditioned GMRES with split-GS for $\nu = 0.001$ with ICP (left) and BTP (right). The coarsest mesh is M20.

Matrix	ICP			BTP		
	Iter	Time	tres	Iter	Time	tres
M40	71	22.55	3.24E-4	84	21.20	2.00E-4
M80	72	48.26	6.64E-5	86	43.73	7.56E-4
M160	75	123.80	5.94E-5	83	108.82	1.84E-4
M320	78	485.84	1.32E-5	82	422.72	2.77E-5

8. Conclusions

The algebraic solution of the discretized and linearized Navier Stokes equations, in case of high Reynolds number, is still a challenging numerical linear algebra problem. Our contribution is to give an overview of the main multigrid approaches to precondition the advection-diffusion block and to approximate the Schur complement matrix. We have also proposed some variants that can improve efficiency and scalability of the block preconditioner in the case of low viscosity value. The use of such sophisticated multigrid variants leads to solving the finest Navier–Stokes discretized system with almost one-million unknowns and 17 million nonzeros in a few minutes, even for the smallest ($\nu = 10^{-3}$) value of the viscosity. We have also shown that the use of low-rank updates may constitute a viable improvement of a given block preconditioner when the number of the eigenvalues of the preconditioned $(1,1)$ block, which are far away from 1 (outliers) is small and roughly independent of h.

Author Contributions: writing—original draft preparation, F.Z. and L.B.; writing—review and editing, F.Z. and L.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the project granted by the Fondazione Cassa di Risparmio di Padova e Rovigo (CARIPARO): *Matrix-Free Preconditioners for Large-Scale Convex Constrained Optimization Problems (PRECOOP)*.

Acknowledgments: We are indebted to M. Nozza for providing the initial version of the multigrid code and for fruitful discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Benzi, M.; Olshanskii, M. An augmented Lagrangian-based approach to the Oseen problem. *SIAM J. Sci. Comput.* **2006**, *28*, 2095–2113.
2. Elman, H.; Sylvester, D.; Wathen, A. *Finite Elements and Fast Iterative Solvers*, 2nd ed.; Oxford University Press: Oxford, UK, 2014.
3. Farrell, P.; Mitchell, L.; Wechsung, F. An augmented Lagrangian preconditioner for the 3D stationary incompressible Navier-Stokes equations at high Reynolds number. *SIAM J. Sci. Comput.* **2019**, *41*, A3073–A3096.
4. Kay, D.; Loghin, D.; Wathen, A. A Preconditioner for the Steady-State Navier-Stokes Equations. *SIAM J. Sci. Comput.* **2002**, *24*, 237–256.
5. Silvester, D.; Elman, H.; Kay, D.; Wathen, A. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *J. Comput. Appl. Math.* **2001**, *128*, 261–279.
6. Bergamaschi, L.; Ferronato, M.; Gambolati, G. Mixed Constraint Preconditioners for the solution to FE coupled consolidation equations. *J. Comput. Phys.* **2008**, *227*, 9885–9897.
7. Bergamaschi, L.; Martínez, A. RMCP: Relaxed Mixed Constraint Preconditioners for Saddle Point Linear Systems Arising in Geomechanics. *Comp. Methods App. Mech. Engrg.* **2012**, *221–222*, 54–62.
8. Keller, C.; Gould, N.; Wathen, A. Constraint Preconditioning for Indefinite Linear Systems. *SIAM J. Matrix Anal. Appl.* **2000**, *21*, 1300–1317.
9. Elman, H. Preconditioning For The Steady-State Navier-Stokes Equations With Low Viscosity. *SIAM J. Sci. Comput.* **2001**, *20*, 1299–1316.
10. Elman, H. Preconditioners for saddle point problems arising in computational fluid dynamics. *Appl. Numer. Math.* **2002**, *43*, 75–89.
11. Benzi, M.; Golub, G.H.; Liesen, J. Numerical solution of saddle point problems. *Acta Numer.* **2005**, *14*, 1–137. doi:10.1017/S0962492904000212.
12. Olshanskii, M.; Reusken, A. Convergence analysis of a multigrid method for a convection-dominated model problem. *SIAM J. Numer. Anal.* **2004**, *42*, 1261–1291.
13. Ramage, A. A multigrid preconditioner for stabilised discretisations of advection-diffusion problems. *J. Comput. Appl. Math.* **1999**, *110*, 187–203.
14. Olshanskii, M.; Vassilevski, Y. Pressure Schur Complement Preconditioners for the Discrete Oseen Problem. *SIAM J. Sci. Comput.* **2007**, *29*, 2686–2704.
15. Quarteroni, A. *Numerical Models for Differential Problems*, 2 ed.; Springer: Milan, Italy, 2012.
16. Elman, H.C.; Silvester, D.J.; Wathen, A.J. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*; Numerical Mathematics and Scientific Computation, Oxford University Press: New York, USA, 2014; pp. xiv+400.
17. Benzi, M.; Simoncini, V. On the eigenvalues of a class of saddle point matrices. *Numer. Math.* **2006**, *103*, 173–196.
18. Simoncini, V. Block triangular preconditioners for symmetric saddle-point problems. *Appl. Numer. Math.* **2004**, *49*, 63–80.
19. Bergamaschi, L. On Eigenvalue distribution of constraint-preconditioned symmetric saddle point matrices. *Numer. Linear Algebra Appl.* **2012**, *19*, 754–772. doi:10.1002/nla.806.
20. Greenbaum, A.; Pták, V.; Strakoš, Z. Any Nonincreasing Convergence Curve is Possible for GMRES. *SIAM J. Matrix Anal. Appl.* **1996**, *17*, 465–469.
21. Klawonn, A.; Starke, G. Block triangular preconditioners for nonsymmetric saddle point problems: field-of-values analysis. *Numer. Math.* **1999**, *81*, 577–594.
22. Benzi, M.; Olshanskii, M.A. Field-of-Values Convergence Analysis of Augmented Lagrangian Preconditioners for the Linearized Navier–Stokes Problem. *SIAM J. Numer. Anal.* **2011**, *49*, 770–788. doi:10.1137/100806485.
23. Trottenberg, U.; Oosterlee, C.; Schuller, A. *Multigrid*; Academic Press: London, UK, 2001.
24. Wesseling, P. *An Introduction to Multigrid Methods*; John Wiley and Sons: Chichester, UK, 1992.
25. Loghin, D.; Wathen, A. J. Schur complement preconditioners for the Navier-Stokes equations. *Int. J. Numer. Methods Fluids* **2002**, *40*, 403–412. doi:10.1002/fld.296.

26. Bergamaschi, L. A Survey of Low-rank Updates of Preconditioners for Sequences of Symmetric Linear Systems. *Algorithms* **2020**, *13*, 100.
27. Kuhlmann, H.; Romanò, F. The Lid-Driven Cavity. *Comput. Methods Appl. Sci.* **2019**, *50*, 233–309.