Comparison of some preconditioners for the coupled Navier-Stokes equations

by

Kristian Hesselberg Brox

THESIS
for the degree of
MASTER OF SCIENCE

(Master i Anvendt matematikk og mekanikk)

Faculty of Mathematics and Natural Sciences
University of Oslo

December 2015

Det matematisk- naturvitenskapelige fakultet
Universitetet i Oslo

Acknowledgements

I want to thank my supervisor, Kent-Andre Mardal. He has always been willing to take the time to answer my questions, while still making me work independently. I also want to thank Magne Nordaas, for providing invaluable help and feedback on both theory and implementation.

The social environment provided by Realistforeningen during my time at the University of Oslo has been an important part of my life for many years. I would especially like to express my gratitude to Kine, Rafael and Luca, for giving me a reason to get up in the morning.

Lastly, I'd like to thank my parents, for always being there for me.

Contents

1 Introduction

Partial differential equations (PDEs) are equations involving the rate of change of continuous variables in two or more dimensions. Some examples of things modeled by PDEs are the transfer of heat through solids, the flow of air around an airplane, and the deformation of blood vessels due to changes in blood pressure and flow velocity. The PDEs we are interested in are the Navier-Stokes equations. They are used to model and simulate the flow of incompressible Newtonian fluids, that is, fluids where the density is constant and the viscosity is independent of the shear rate. Efficient simulation of blood flow could lead to faster and more reliable diagnostication of cerebral aneurysms, giving earlier warning of danger of intracranial bleeding and preventing unnecessary invasive procedures.Although blood is not a Newtonian fluid, in most arteries it is close enough to Newtonian that the Navier-Stokes equations are a reasonable model for arterial blood flow ([1], p.328).

Most PDEs, including the Navier-Stokes equations, do not have closed-form analytical solutions, or only have closed-form analytical solutions for a limited set of boundary conditions. For this reason we usually instead seek an approximate numerical solution. Constructing good numerical methods for solving the Navier-Stokes equations has been an important field of research for a long time, and many different numerical frameworks for simulating Navier-Stokes flow have been developed. We will follow Deparis et al. [2] in using an implicit finite difference scheme to linearize and discretize the equations in time, and the finite element method (FEM) with mixed finite elements to discretize the coupled equations in the spatial dimensions. The discretization process leads to an indefinite, ill-conditioned linear system of equations, with a block structure shown in equation (1.1).

$$A \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} F & B^T \\ B & C \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ g \end{bmatrix} \tag{1.1}$$

The system (1.1) is large and sparse. While direct solution methods for such systems exist, systems with those two properties are often solved with an iterative solution method. Iterative methods are good at exploiting the sparsity of a linear system to minimize memory usage and computation time, but they have trouble with slow convergence when used for ill-conditioned systems [3]. To speed up convergence and solve the system in reasonable time, a preconditioner is required. A preconditioner for a discrete system like (1.1) is a nonsingular matrix which in some way mimics the inverse of the coefficient matrix A, and multiplying a system with a preconditioner is referred to as preconditioning.

Many different preconditioners for (1.1) have been suggested in the literature, and several general strategies for constructing such preconditioners are discussed in [3].

Benzi et al. [4] propose a preconditioning strategy based on a skew-symmetric splitting of the coefficient matrix, while more recently Heister et al. [5] propose an approach leveraging Grad-Div stabilization. Deparis et al. [2] compare several different preconditioners, including one based on SIMPLE iteration and one based on the Yosida method, in the context of highly parallellized simulations of blood flow.

The time dependent Stokes equations can be seen as a simplificiaton of the Navier-Stokes equations, and preconditioners developed for use on the discretized time dependent Stokes equations can be modified to be applicable to (1.1), at least for low Reynolds number flow where convection is not the dominant factor. Mardal et al. [6] motivate a block diagonal preconditioner for the time dependent Stokes equations by operator preconditioning of the continuous equations, while Cai [7] proposes several projection method based preconditioners.

In this thesis, we will compare the Yosida method based preconditioner from [2], the block triangular preconditioner from [6], and one projection method based preconditioner from [7]. The performance of each preconditioner will be examined on both the time dependent Stokes equations and the Navier-Stokes equations. We will also consider a variant of the Yosida method based preconditioner, which can be seen as a combination of that and the one based on the SIMPLE iteration. The results will include eigenvalues and condition numbers of the preconditioned differential operators, the performance of the preconditioners on two 2D model problems, and the performance of the preconditioners in a 3D blood flow simulation.

This thesis is organized in the following way: Chapter 2 introduces the mathematical model, including the Navier-Stokes equations and two closely related problems, the time dependent Stokes equations and the Oseen equations. In chapter 2 we also introduce the finite element method. Some iterative solution methods for discrete systems of linear equations, including the BiCGStab algorithm which was used in the simulations done for this thesis, are the topic of chapter 3. In chapter 4 we define the preconditioners that will be the focus of this thesis. Chapter 4 also includes a brief analysis of the eigenvalues of the preconditioned systems. A detailed description of the numerical work done is the topic of chapter 5, and the results are presented in chapter 6. Chapter 7 contains a summary and discussion of the results in chapter 6.

The framework we have used for the numerical experiments is FEniCS [8], using the Python interface dolfin. To easily build operators like those discussed in chapter 4, we have taken advantage of the cbc.block module [9]. The source code for the numerical work is available on the web page http://www.bitbucket.com/krisbrox/thesis.

1.1 Notation

The following is a non-exhaustive list of symbols used throughout this thesis. We will adopt the convention that a bold face character denotes a vector.

	General Symbols
Ω	An open subset of $\mathbb{R}^n$
$\partial\Omega$	boundary of Ω
$\boldsymbol{u}$	Velocity field
p	Pressure
$\boldsymbol{n}$	Unit vector normal to $\partial\Omega$
$\nabla \cdot \boldsymbol{u}$	Divergence of $\boldsymbol{u}$
$\nabla \boldsymbol{u}$	Gradient of $\boldsymbol{u}$
$\Delta \boldsymbol{u}$	Laplacian of $\boldsymbol{u}$, defined by $\Delta := \nabla \cdot \nabla$
$\langle \cdot, \cdot \rangle$	L^2-inner product on Ω
$\lVert \cdot \rVert_k$	Norm defined by the H^k-inner product on Ω if $k = 0, 1$
$\lVert \cdot \rVert_2$	Euclidian norm
	Parameters
μ	Viscosity
ρ	Density
ν	Kinematic viscosity, defined as $\nu := \mu/\rho$
Δt	Timestep
Re	Reynolds number
h	Discretization parameter, proportional to the length of the longest edge of an element in a partition of Ω
	Properties of matrices
$\sigma(A)$	Set of singular values of A
$\lambda(A)$	Set of eigenvalues of A
$\kappa_1(A)$	2-norm condition number of A, defined by $\kappa_1(A) := \max \sigma A / \min \sigma A$
$\kappa_2(A)$	Largest eigenvalue of A divided by the smallest (by moduli)
	Function Spaces
$L^2(\Omega)$	Space of square-integrable functions on Ω
$H^k(\Omega)$	Space of L^2-functions on Ω with derivatives up to order k which are also in $L^2(\Omega)$
$H_0^k(\Omega)$	Space of functions in $H^k(\Omega)$ which equal zero on $\partial\Omega$
H^{-1}	Dual space of H^1
	Matrices
K	Stiffness matrix
M	Mass matrix
G	Discrete gradient
D	Discrete divergence

2 Mathematical Model

2.1 The Navier-Stokes equations

The incompressible Navier-Stokes equations describe flow in incompressible, viscous fluids. They are derived from Newtons second law, stress-strain relations and conservation of mass. For the derivation see e.g. White [10]. The equations are

$$\rho(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) - \mu \, \Delta \boldsymbol{u} + \nabla \hat{p} = \hat{\boldsymbol{f}} \quad \text{in } \Omega, t > 0, \tag{2.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } \Omega, t > 0, \tag{2.2}$$

with suitable boundary conditions. Here Ω is the fluid domain, $\boldsymbol{u}$ is the fluid velocity, and $\hat{p}$ is the pressure. $\hat{\boldsymbol{f}}$ are the external forces, and μ is the viscosity of the fluid. In this thesis, we will usually scale the momentum equation (2.1) by $\frac{1}{\rho}$. Doing so, and including boundary and initial conditions, leads to the following system of equations:

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} - \nu \, \Delta \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \boldsymbol{f} &\quad \text{in } \Omega, t > 0, \\ \nabla \cdot \boldsymbol{u} = 0 &\quad \text{in } \Omega, t > 0, \\ \boldsymbol{u} = \boldsymbol{g}_D &\quad \text{on } \partial\Omega_D, \\ \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n} = \boldsymbol{g}_N &\quad \text{on } \partial\Omega_N, \\ \boldsymbol{u} = \boldsymbol{u}_0 &\quad \text{in } \Omega, t = 0. \end{aligned} \tag{2.3}$$

Here, $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ is the boundary of Ω, divided into a Neumann part and a Dirichlet part. The parameter $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity, $p = \frac{1}{\rho}\hat{p}$ is the scaled pressure and $\boldsymbol{f} = \frac{1}{\rho}\hat{\boldsymbol{f}}$ the scaled external forces. These equations are of fundamental importance in studying flow of fluids at low to moderate speeds, e.g. water flowing through a pipe, wind hitting a windmill or arterial bloodflow.

The equations (2.3) are non-linear, and analytic solutions only exist for a very limited number of boundary conditions. The velocity field may change direction rapidly in thin boundary layers close to solid walls, or as a consequence of complex flow domain geometry [11]. This calls for fine meshes and small time steps, making numerical approximations expensive to obtain. In addition, the saddle-point nature of the problem makes efficient and robust solution methods difficult to find. For these reasons, and due to their importance in industrial research and applications, methods to numerically solve the Navier-Stokes equations efficiently has been, and still is, an important area of research [11].

Many characteristics of any given incompressible viscous fluid flow are predicted by the Reynolds number of the flow. The definition of the Reynolds number is

$$\mathrm{Re} = \frac{\text{inertial forces}}{\text{viscous forces}} = \frac{\rho L v}{\mu} = \frac{Lv}{\nu}, \tag{2.4}$$

where v is the velocity, L is a characteristic length, μ is the viscosity of the fluid, and ρ is the density of the fluid. The velocity v and length L have different meanings in different contexts, depending on the flow domain and boundary. When considering flow through e.g. a circular pipe, the convention is to define

$$\mathrm{Re} = \frac{QD}{\nu A}, \tag{2.5}$$

where D is the diameter, A the cross-sectional area, and Q the volumetric flow rate, or mass flux. For flow in an unbounded domain past a thin plate, the Reynolds number might be defined as in (2.4), with L the length of the plate, and v the velocity of the fluid relative to the plate at the leading edge.

When Re is small the flow is laminar, i.e. the flow will be without eddies and swirls, and without cross-currents perpedicular to the main direction of the flow. If Re is large (above ~ 2000), the flow will usually be fully turbulent. For flow at moderate Reynolds numbers (100 to 1000), like most arterial blood flow, the effects of turbulence are small.

In this thesis we will be considering two sets of equations which are closely related to the Navier-Stokes equations: the time dependent Stokes equations and the modified Oseen equations.

2.1.1 The modified Oseen equations

The modified Oseen equations

$$\begin{aligned} \frac{1}{\Delta t}\boldsymbol{u} - \nu\,\Delta\boldsymbol{u} + \boldsymbol{u}_1\cdot\nabla\boldsymbol{u} + \nabla p &= \boldsymbol{f}, && \text{in } \Omega,\\ \nabla\cdot\boldsymbol{u} &= 0, && \text{in } \Omega,\\ \boldsymbol{u} &= \boldsymbol{g}_D && \text{on } \partial\Omega_D,\\ \nu\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{n}} - p\boldsymbol{n} &= \boldsymbol{g}_N && \text{on } \partial\Omega_N, \end{aligned} \tag{2.6}$$

arise from applying an implicit timestepping procedure on the Navier-Stokes equations and linearizing the convective term $\boldsymbol{u}^n\cdot\nabla\boldsymbol{u}^n$ as $\boldsymbol{u}^{n-1}\cdot\nabla\boldsymbol{u}^n$. Here $\boldsymbol{u} = \boldsymbol{u}^n$ is the unknown velocity at some time $t_n = t_0 + n\Delta t$, $\boldsymbol{u}_1 = \boldsymbol{u}^{n-1}$ is the known velocity from the previous iteration, and $\boldsymbol{f}$ includes external forces and the velocity from the previous iteration. Solving the problem (2.6) is equivalent to solving the implicitly time-discretized Navier-Stokes equations by applying one fixed-point iteration, so we will treat a series of solutions of (2.6) as a solution of (2.3).

2.1.2 The time dependent Stokes equations

The time dependent Stokes equations read

$$
\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} - \nu\,\Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} && \text{in } \Omega, t > 0,\\
\nabla \cdot \boldsymbol{u} &= 0 && \text{in } \Omega, t > 0.\\
\boldsymbol{u} &= \boldsymbol{g}_D && \text{on } \partial\Omega_D,\\
\nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n} &= \boldsymbol{g}_N && \text{on } \partial\Omega_N,\\
\boldsymbol{u} &= \boldsymbol{u}_0 && \text{in } \Omega, t = 0.
\end{aligned}
\tag{2.7}
$$

Discretizing in time with an implicit finite difference leads to

$$
\begin{aligned}
\frac{1}{\Delta t}\boldsymbol{u} - \nu\,\Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f}, && \text{in } \Omega,\\
\nabla \cdot \boldsymbol{u} &= 0, && \text{in } \Omega,\\
\boldsymbol{u} &= \boldsymbol{g}_D && \text{on } \partial\Omega_D,\\
\nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n} &= \boldsymbol{g}_N && \text{on } \partial\Omega_N,
\end{aligned}
\tag{2.8}
$$

which is just the modified Oseen equations without the convective term. The system (2.8) can be written as the stationary singular perturbation problem

$$
\begin{aligned}
(I - \epsilon^2\,\Delta)\boldsymbol{u} + \nabla p &= \boldsymbol{f},\\
\nabla \cdot \boldsymbol{u} &= 0,
\end{aligned}
\tag{2.9}
$$

plus boundary conditions, where $\epsilon = \sqrt{\nu \Delta t}$ and the pressure has been scaled by Δt.

The equations (2.7) arise from discretizing (2.3) in time with a semi-implicit time-stepping procedure, linearizing the convective term $\boldsymbol{u}^{n+1} \cdot \nabla \boldsymbol{u}^{n+1}$ as $\boldsymbol{u}^n \cdot \nabla \boldsymbol{u}^n$ and absorbing it into $\boldsymbol{f}$. The equations (2.7) can be seen as a simplification of the modified Oseen equations, and as such (2.7) can be useful for testing solution methods for (2.3), as the performance of some particular solver or preconditioner on the time dependent Stokes equations might give an indication of an upper bound for the performance of the same solver or preconditioner on the modified oseen problem [6].

2.2 The finite element method

As our model problem for introducing the finite element method we will use the Poisson problem with homogeneous Dirichlet boundary conditions:
Find a function $u(\boldsymbol{x})$ in some function space $V(\Omega)$ such that

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}) \qquad && \forall \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= 0 && \forall \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{2.10}$$

where $\Omega \in \mathbb{R}^d$ is the domain and $\partial\Omega$ is the domain boundary. In the following we omit the domain variables, i.e. we write $u := u(\boldsymbol{x})$.

2.2.1 The weak formulation

The above set of equations is referred to as the *strong formulation* of the problem. Solving the strong formulation directly is in many cases impractical. Enforcing boundary conditions on a discretized strongly formulated problem poses difficulties, especially for complex domains. Further, the double derivative in (2.10) can be problematic [3]. Instead of working directly with the strong formulation, we will find a *weak formulation* of the problem.

We multiply the first equation with some function v from a function space $\hat{V}$ and integrate over Ω to get

$$\int_\Omega -\Delta u v \,\mathrm{d}\boldsymbol{x} = \int_\Omega f v \,\mathrm{d}\boldsymbol{x}. \tag{2.11}$$

Provided the involved functions are sufficiently smooth, integrating the left-hand side by parts gives

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x} = \int_\Omega f v \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \frac{\partial u}{\partial \boldsymbol{n}} v \,\mathrm{d}S, \tag{2.12}$$

Where the dot indicates a scalar product. Let

$$L^2(\Omega) = \{u : \Omega \to \mathbb{R} \Big| \int_\Omega u^2 < \infty\}$$

and

$$H_0^1(\Omega) = \{u \in L^2(\Omega) \Big| \nabla u \in L^2(\Omega), u|_{\partial\Omega} = 0\},$$

i.e. $H_0^1(\Omega)$ is the space of square integrable functions on Ω that equal 0 on $\partial\Omega$, with square integrable gradient. The gradient ∇u being square integrable means that each first-order partial derivative $\frac{\partial v}{\partial x_i}$ for $i = 1, \ldots, d$, is square integrable. If we equip this space with the inner product

$$(u, v)_{1,\Omega} = \int_\Omega uv \,\mathrm{d}\boldsymbol{x} + \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x},$$

and the associated norm

$$\|u\|_{1,\Omega} = (\int_\Omega |u|^2 + |\nabla u|^2)^{\frac{1}{2}} \,\mathrm{d}\boldsymbol{x},$$

then $H_0^1(\Omega)$ space is the Sobolev space $W^{1,2}(\Omega)$, which is also a Hilbert space. In the rest of this section we omit the Ω-subscript. Now we set $V(\Omega) = V = H_0^1(\Omega)$ and $\hat{V}(\Omega) = \hat{V} = H_0^1(\Omega)$. The weak formulation of (2.10) is

Find $u \in V$ such that, for all $v \in \hat{V}$,

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x} = \int_\Omega fv \,\mathrm{d}\boldsymbol{x}. \tag{2.13}$$

Note that the second term in the right hand side of (2.10) is zero due to v vanishing on $\partial\Omega$.

We would like the problem (2.13) to be well posed, in the sense that there should exist a unique solution to it. A sufficient condition for this is (2.13) satistying the **Lax-Milgram theorem** (see e.g. Evans [12]). The version of the theorem that applies in the current situation states that, if $V(\Omega)$ is a real Hilbert space, and $a(\cdot, \cdot)$ is a symmetric bilinear form which is

(i) Bounded: $|a(u, v)| < C\|u\|_1\|v\|_1$, $\forall u, v \in V$, and

(ii) Coercive: $a(u, u) \geq D\|u\|_1^2$, $\forall u \in V$

for some constants C and D, then for any bounded linear functional $l : V \to \mathbb{R}$ and all $v \in V$, the equation

$$a(u, v) = l(v)$$

has a unique solution. A proof of this theorem can be found in [12].

In our current example, $a(u, v) := \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x}$ and $l(v) := \int_\Omega fv \,\mathrm{d}\boldsymbol{x}$. To show (i):

$$\begin{aligned}
(\|u\|_1\|v\|_1)^2 &= \Big(\int_\Omega u^2 \,\mathrm{d}\boldsymbol{x} + \int_\Omega (\nabla u)^2 \,\mathrm{d}\boldsymbol{x}\Big)\Big(\int_\Omega v^2 \,\mathrm{d}\boldsymbol{x} + \int_\Omega (\nabla v)^2 \,\mathrm{d}\boldsymbol{x}\Big) \\
&\geq \int_\Omega (\nabla u)^2 \,\mathrm{d}\boldsymbol{x} \int_\Omega (\nabla v)^2 \,\mathrm{d}\boldsymbol{x} \\
&\geq |\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x}|^2 \\
&= |a(u, v)|^2 \\
\Rightarrow |a(u, v)| &\leq C\|u\|_1\|v\|_1,
\end{aligned}$$

where we used the Cauchy-Schwarz inequality between line two and line three. To show that $a(\cdot,\cdot)$ is coercive, we need the Poincaré inequality: Given $p \in [1,\infty)$ and Ω a bounded subset of $\mathbb{R}^n$, there exists a constant C_0 such that, for all $f \in H_0^1(\Omega)$,

$$\|f\|_{L^p(\Omega)} \leq C_0 \|\nabla f\|_{L^p(\Omega)}. \tag{2.14}$$

Now, observe that

$$\|u\|_1^2 = \int_\Omega u^2 + (\nabla u)^2 \,\mathrm{d}\boldsymbol{x} \leq (1 + C_0) \int_\Omega (\nabla u)^2 \,\mathrm{d}\boldsymbol{x},$$

implying

$$a(u,u) = \int_\Omega (\nabla u)^2 \,\mathrm{d}\boldsymbol{x} \geq \frac{1}{1+C_0} \|u\|_1^2,$$

which shows the coercivity of a with $D = \frac{1}{1+C_0}$. Since $a(\cdot,\cdot)$ is both coercive and bounded, the Lax-Milgram theorem guarantees the existence of a unique solution to the problem 2.13 for any given $l \in V'$.

If the strong formulation (2.10) has a solution, that solution coincides with the solution of the corresponding weak formulation (2.13) ([3], p. 17). A solution to the strong formulation needs to be twice differentiable in Ω, which is a much more stringent requirement than square integrability of the first derivatives (Ibid.).

2.2.2 The finite element

We will follow Ciarlet's definition of a finite element [13].

Definition 2.1. *Let*

(i) $K \subset \mathbb{R}^d$ *be a bounded closed set with nonempty interior and piecewise smooth boundary (**The element domain**).*

(ii) $\mathcal{V} = \mathcal{V}(K)$ *be a finite-dimensional space of functions on* K.

(iii) $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ *be a basis for* $\mathcal{V}'$ *(the **nodal variables**)*

Then $(K, \mathcal{V}, \mathcal{L})$ *is called a **finite element**.*

Definition 2.2. *Let* $(K, \mathcal{V}, \mathcal{L})$ *be a finite element. The basis* $\{\phi_1, \phi_2, \ldots, \phi_n\}$ *of* $\mathcal{V}$ *dual to* $\mathcal{L}$ *(i.e.* $l_i(\phi_j) = \delta_{ij}$*) is called the **nodal basis** of* $\mathcal{V}$.

There are many different kinds of elements, but among the simplest and most commonly used are the *Lagrange* (or *continuous Galerkin*) elements. We will denote the

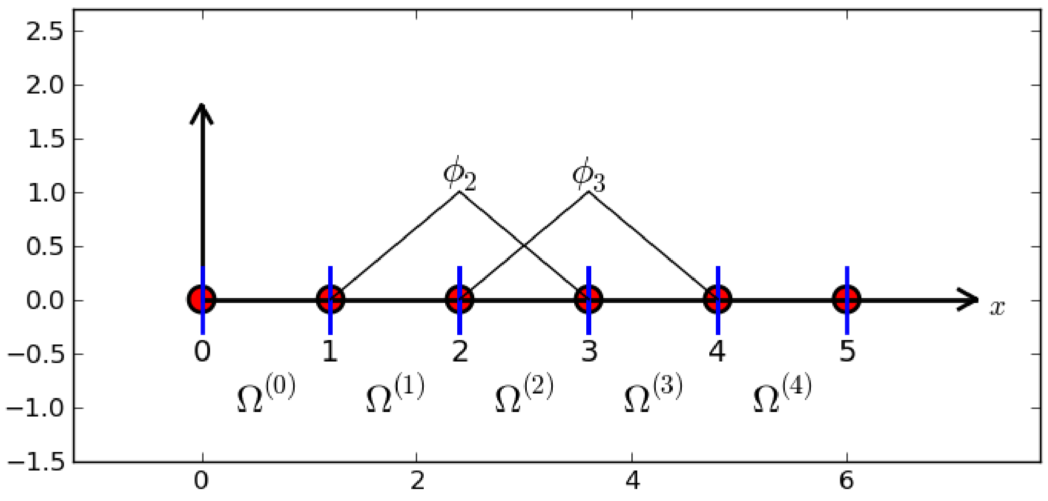


Figure 1: P1 elements in 1D

Lagrange element of polynomial order q by P_q. The domain K we will divide into k non-overlapping intervals in 1D, triangles in 2D and tetrahedras in 3D, so that $\cup_{j=1}^{n} K_j = \Omega$.

Example 2.1. The basis P_1 for $\mathcal{V}(K)$ is composed of the the piecewise linear functions $\{\phi_k\}_{j=1}^{n}$ satisfying $\phi_j(l_i) = \delta_{ij}$. On the 1D reference element $[-1, 1]$ with the nodes/degrees of freedom $\{-1, 1\}$ the basis functions are $\{0.5 - 0.5x, 0.5x - 0.5\}$ (see figure 1).

2.2.3 Discretization

To better illustrate the discretization process, we will use a more general form of the Poisson problem:
Find u such that

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= g_D(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \partial\Omega_D, \\ \partial_{\boldsymbol{n}} u(\boldsymbol{x}) &= g_N(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \partial\Omega_N, \end{aligned} \tag{2.15}$$

where $\partial_{\boldsymbol{n}} u = \nabla u \cdot \boldsymbol{n}$ is the normal derivative of u at the boundary, $\partial\Omega_D$ is the Dirichlet part of the boundary and $\partial\Omega_N$ is the Neumann part of the boundary. We assume $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. The weak formulation of (2.15) is:
Find $u \in V$ such that, for all $v \in \hat{V}$,

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} = \int_\Omega f v \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_N} g_N v \, \mathrm{d}S. \tag{2.16}$$

Discretizing this as outlined in the previous section leads to:
Find $u_h \in V_{h,g_D,D}$ *such that*

$$\int_\Omega \nabla u_h \cdot \nabla v_h \,\mathrm{d}\boldsymbol{x} = \int_\Omega f v_h \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_N} g_N v_h \,\mathrm{d}S, \quad \forall v_h \in V_{h,g_D,D}, \tag{2.17}$$

where $V_{h,g_D,D}$ is the space spanned by our basis, chosen to equal g_D on $\partial\Omega_D$. We write u_h as a linear combination of the basis functions, $u_h = \sum_{j=1}^n c_j\phi_j$, and without loss of generality set $v = \phi_i$. Then

$$\int_\Omega (\nabla \sum_{j=1}^n c_j\phi_j) \cdot \nabla\phi_i \,\mathrm{d}\boldsymbol{x} = \int_\Omega f\phi_i \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_N} g_N\phi_i \,\mathrm{d}S \quad i = 1, 2, \ldots, n, \tag{2.18}$$

$$\Rightarrow \sum_{j=1}^n c_j \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}\boldsymbol{x} = \int_\Omega f\phi_i \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_N} g_N\phi_i \,\mathrm{d}S \quad i = 1, 2, \ldots, n. \tag{2.19}$$

This is equivalent to the linear system

$$A\boldsymbol{u} = \boldsymbol{b}, \tag{2.20}$$

where

$$A_{i,j} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}\boldsymbol{x}, \tag{2.21}$$

$$\boldsymbol{b}_i = \int_\Omega f\phi_i \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_N} g_N\phi_i \,\mathrm{d}S, \tag{2.22}$$

and $\boldsymbol{u}$ is a vector containing the unknown c_j for $j = 1, 2, \ldots, n$.

The system (2.20) can be solved either directly with Gaussian elimination, or with an iterative method like the conjugate gradient or Gauss-Seidel methods. The systems we get with the finite element method are typically sparse (meaning the number of nonzero entries in A is $O(n)$). The reason for this is that the basis functions used have support on a finite, small subset of the domain, so the integral $\int_\Omega \nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}\boldsymbol{x}$ is zero for most $i \neq j$. In fact, the number of such integrals which are nonzero, for any given i, can be independent of the dimension of A. This is evident when considering e.g., in 2D, a partition of the unit square into m non-overlapping triangles, or in 1D, figure 1. Sparse systems, like (2.20), are often more easily solved by an iterative solver than a direct one. This will be be discussed further in chapter 3.

An alternative, more abstract version of the above formulation of the finite element method (exemplified by the Poisson problem) is as follows: the weak form of the Poisson problem can be written

Find $u \in V$ such that

$$Au = f, \tag{2.23}$$

where $A : V \to V'$ is a differential operator, V is a Hilbert space with dual V' and inner product $\langle \cdot, \cdot \rangle$, and $f \in V'$. Define a suitable test space $\hat{V}$ and set

$$\begin{aligned} a(u, v) &= \langle Au, v \rangle, \\ l(v) &= \langle f, v \rangle, \end{aligned} \tag{2.24}$$

where $a : V \times \hat{V} \to \mathbb{R}$ is a bilinear form and $l : \hat{V} \to \mathbb{R}$ is a linear functional on $\hat{V}$. The problem (2.23) now reads

$$a(u, v) = l(v) \quad \forall v \in \hat{V}. \tag{2.25}$$

Choose a finite element and discretize the domain:

Find $u_h \in V_h$ such that

$$a(u_h, v_h) = l(v_h) \quad \forall v_h \in \hat{V}_h. \tag{2.26}$$

Write u_h as a linear combination of the basis functions of V_h and solve the resulting linear system.

2.2.4 Weak Formulations of the time dependent Stokes and Oseen problems

In this section we will find weak formulations and discretizations of the time dependent Stokes problem and the Oseen problem. To discretize equations (2.7) and (2.6) in space we will use the finite element method as outlined in chapter 2.2. For each set of equations, we multiply the first equation with a test function $\boldsymbol{v} \in \hat{V} = H^1_{0,D}(\Omega)$, and the second equation with a test function $q \in \hat{P} = L^2(\Omega)$. Integrating by parts where applicable, the weak formulation of problem (2.7) is:

Find $(\boldsymbol{u}, p) \in H^1_{g_D,D}(\Omega) \times L^2(\Omega)$ such that, for all $(\boldsymbol{v}, q) \in \hat{V} \times \hat{P}$,

$$\begin{aligned} \frac{1}{\Delta t} \int_\Omega \boldsymbol{u}\boldsymbol{v} \,\mathrm{d}\boldsymbol{x} + \nu \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} + \int_\Omega p \nabla \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} &= \int_\Omega \boldsymbol{f}\boldsymbol{v} \,\mathrm{d}\boldsymbol{x}, \\ \int_\Omega q \nabla \cdot \boldsymbol{u} \,\mathrm{d}\boldsymbol{x} &= 0. \end{aligned} \tag{2.27}$$

Here, $\boldsymbol{f}$ has been redefined to also contain the boundary terms arising from the partial integrations of the $\Delta \boldsymbol{u}$ and ∇p terms. Note that we have redefined the equations to solve for the negative pressure, meaning we have substituted $-p$ for p.

The weak formulation of the modified Oseen problem reads:

Find $(\boldsymbol{u}, p) \in H^1_{g_D,D}(\Omega) \times L^2(\Omega)$ *such that, for all* $(\boldsymbol{v}, q) \in \hat{V} \times \hat{P}$,

$$\begin{aligned} \frac{1}{\Delta t}\int_\Omega \boldsymbol{u}\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} + \nu\int_\Omega \nabla\boldsymbol{u} : \nabla\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} + \int_\Omega \boldsymbol{u}_1\cdot\nabla\boldsymbol{u}\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} + \int_\Omega p\nabla\cdot\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} &= \int_\Omega \boldsymbol{f}\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}, \\ \int_\Omega q\nabla\cdot\boldsymbol{u}\,\mathrm{d}\boldsymbol{x} &= 0, \end{aligned} \tag{2.28}$$

where as with the time dependent Stokes equations $\boldsymbol{f}$ has been redefined to also contain the boundary terms and $-p$ substituted for p.

Let $\{V_h \times P_h\}_{h\in(0,1]} \subset H^1(\Omega) \times L^2(\Omega)$ be finite element spaces, where h represents the scale of the discretization of the domain. If h_i is the length of the longest edge of element domain number i, we define h as $\max_i h_i$. Let $\langle\cdot,\cdot\rangle$ denote the L^2 inner product on Ω, and define

$$a_\epsilon(\boldsymbol{u}_h, \boldsymbol{v}) = \int_\Omega \boldsymbol{u}_h\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} + \epsilon^2\int_\Omega \nabla\boldsymbol{u}_h : \nabla\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}.$$

Then the finite element formulation of (2.9) is:

Find $(\boldsymbol{u}_h, p) \in V_h \times P_h$ *such that, for all* $(\boldsymbol{v}, q) \in \hat{V}_h \times \hat{P}_h$,

$$\begin{aligned} a_\epsilon(\boldsymbol{u}_h, v) + \langle p_h, \nabla\cdot\boldsymbol{v}\rangle &= \langle f, \boldsymbol{v}\rangle \\ \langle\nabla\cdot\boldsymbol{u}_h, q\rangle &= 0. \end{aligned} \tag{2.29}$$

For (2.6) we define

$$f_\nu(\boldsymbol{u}_h, v) = \frac{1}{\Delta t}\int_\Omega \boldsymbol{u}_h\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} + \nu\int_\Omega \nabla\boldsymbol{u}_h : \nabla\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} + \int_\Omega \boldsymbol{u}_1\cdot\nabla\boldsymbol{u}_h\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}. \tag{2.30}$$

The finite element formulation of (2.6) is:

Find $(\boldsymbol{u}_h, p) \in V_h \times P_h$ *such that, for all* $(\boldsymbol{v}, q) \in \hat{V}_h \times \hat{P}_h$,

$$\begin{aligned} f_\nu(\boldsymbol{u}_h, v) + \langle p_h, \nabla\cdot\boldsymbol{v}\rangle &= \langle f, \boldsymbol{v}\rangle \\ \langle\nabla\cdot\boldsymbol{u}_h, q\rangle &= 0. \end{aligned} \tag{2.31}$$

As outlined in section 2.2.3, for any given choice of bases $V_h \times P_h$, the equations (2.29) and (2.31) lead to linear systems on the form

$$A\begin{bmatrix}\boldsymbol{u}\\ p\end{bmatrix} = \begin{bmatrix}F & B\\ B^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{u}\\ p\end{bmatrix} = \begin{bmatrix}\boldsymbol{f}\\ 0\end{bmatrix}. \tag{2.32}$$

We may equivalently write (2.32) as

$$F\boldsymbol{u} + Bp = \boldsymbol{f}, \tag{2.33}$$

$$B^T\boldsymbol{u} = 0. \tag{2.34}$$

Before choosing our bases for V_h and P_h, we need to know the conditions under which the system (2.33)-(2.34) has a unique solution. For the following section we have leaned heavily on [14].

Assuming that F is positive definite, we can multiply the first equation by F^{-1} to obtain an expression for $\boldsymbol{u}$, and insert that into the second. This leads to the following linear system for p:

$$B^T F^{-1} B p = B^T F^{-1} \boldsymbol{f}. \tag{2.35}$$

The solution to (2.35) may then be plugged into 2.33, letting us solve

$$F\boldsymbol{u} = \boldsymbol{f} - Bp \tag{2.36}$$

for $\boldsymbol{u}$. For the above process to output uniquely determined $\boldsymbol{u}$ and p, the Schur complement $B^T F^{-1} B$ needs to be non-singular. As F is positive definite by assumption, this simplifies to requiring $B^T B$ to be non-singular. A necessary and sufficient condition for $B^T B$ to be non-singular is $\mathrm{Ker}(B) = 0$, which is equivalent to

$$\sup_{\boldsymbol{v} \in \hat{V}_h} \int_\Omega p \nabla \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} > 0, \tag{2.37}$$

for all $p \in P_h$.

In addition to needing the discrete system to be solvable, we need it to converge to the true solution as the mesh is refined. To guarantee convergence, our choice of elements must fulfill the *discrete* $\inf\sup$-*condition*, commonly referred to as the *Ladyzenskaja-Babuška-Brezzi condition* (see [15], [16]):

$$\inf_p \sup_{\boldsymbol{v}} \frac{\int_\Omega p \nabla \cdot \boldsymbol{v}}{\|\boldsymbol{v}\|_1 \|p\|_0} > \beta. \tag{2.38}$$

Here, β is a positive, nonzero constant that is independent of the mesh resolution. The condition (2.38) is of particular usefulness when determining which pairs of finite element bases $\{\phi_i\}$, $\{\psi_j\}$ for V_h and P_h lead to a solvable linear system and a stable, converging method [14].

The Taylor-Hood (P2-P1) elements are piecewise quadratic Lagrange polynomials for the velocity, and piecewise linear Lagrange polynomials for the pressure, defined on a mesh of triangles in 2D or tetrahedra in 3D. The P2-P1 elements are uniformly stable for the time-dependent Stokes equations, in the sense that they fulfill the discrete $\inf\sup$-condition (2.38). This is established in e.g. [17].

One relevant part of the finite element solution algorithm for these problems that we have not mentioned here is how to enforce essential (Dirichlet) boundary conditions in practice. One way of enforcing Dirichlet boundary conditions is following the

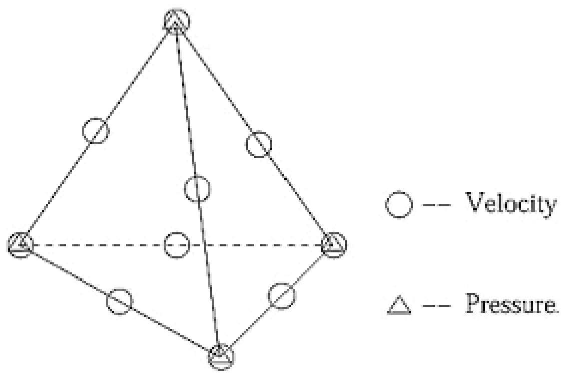


Figure 2: The Taylor-Hood element in 3D

above procedure until (2.32) is reached, and then modifying the coefficient matrix on the left-hand side as well as the vector on the right hand side to fulfill the boundary conditions. This can be done in several ways, and the approach taken in in this thesis is as follows: Assume the i'th component of $\boldsymbol{u}$, $\boldsymbol{u}_i = c_1$ is the velocity in the x-direction at a node where the velocity is known and equal to (c_1, c_2, c_3). To make sure the computed solution of (2.32) has $\boldsymbol{u}_i = c_1$, we set $\boldsymbol{f}_i = c_1$ and $A_{i,j} = A_{j,i} = \delta_{i,j}$ for each $j = 1, \ldots, n$.

3 Iterative solution methods

Discretizing a PDE with the finite element method results in a system of equations. If those equations are linear, the system is on the form

$$A\boldsymbol{x} = \boldsymbol{b}, \quad A \in \mathbb{R}^{n\times n}, \quad \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n. \tag{3.1}$$

A differential operator discretized by the finite element method is typically sparse, meaning it contains only $O(n)$ non-zero entries. If A is nonsingular, the system (3.1) has the solution $\boldsymbol{x} = A^{-1}\boldsymbol{b}$. Unfortunately, A^{-1} is typically a full matrix with $O(n^2)$ non-zero entries. For this reason, solving the equation by computing the inverse directly, by way of Gaussian elimination or LU factorization, requires $O(n^3)$ operations in the general case, as well as $O(n^2)$ bytes of storage. This is very inefficient.

Iterative methods allow us to capitalize on the sparsity of A, requiring only $O(n)$ operations and bytes of storage in the best case. It is also easier in practice to approach this optimal performance with iterative methods than with direct methods. The idea is to make an initial guess $\boldsymbol{x}_0$, followed by generating a sequence $\{\boldsymbol{x}_k\}$ of approximations (hopefully) converging to the solution $\boldsymbol{x}$. Iterative methods thus allow using any $\boldsymbol{x}_k$ as an approximation to $\boldsymbol{x}$, while any intermediate result in a direct method will be of little use.

The Krylov subspace family of iterative methods is the one most used in solving the kinds of problems we are interested in, especially nonsymmetric and highly ill-conditioned problems like the discrete modified Oseen equations. We largely base our exposition of Krylov subspace methods and the biconjugate gradient stabilized algorithm on the book by Yousef Saad [18].

Definition 3.1. The r-order Krylov subspace generated by the $n \times n$ matrix A and the vector $\boldsymbol{y} \in \mathbb{R}^n$ is given by

$$\mathcal{K}_r(A, \boldsymbol{y}) = span\{\boldsymbol{y}, A\boldsymbol{y}, A^2\boldsymbol{y}, \dots, A^{r-1}\boldsymbol{y}\}. \tag{3.2}$$

The idea behind Krylov subspace methods is to generate a sequence of approximations $\boldsymbol{x}_k$ such that $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$, i.e. $\boldsymbol{x}_k$ is in the affine space spanned by x_0 and $\{A^l\boldsymbol{r}_0\}_{l=0}^{k-1}$, where $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$. If the residual $\boldsymbol{r}_k \in \mathcal{K}_{k+1}(A, \boldsymbol{r}_0)$ at some point in this process reaches a value sufficiently close to zero in an appropriate norm, we have found an approximate solution. Many methods will ensure that the residuals are linearly independent, guaranteeing convergence in at most n iterations (in the absence of round-off errors). This is done by imposing the condition that

$$\boldsymbol{b} - A\boldsymbol{x}_k \;\perp\; \mathcal{L}_k,$$

where $\mathcal{L}_k$ is some other Krylov subspace. Different krylov subspace methods are based on different choices of $\mathcal{L}_k$, and can be divided broadly into two categories: methods that choose $\mathcal{L}_k = \mathcal{K}_k(A, \boldsymbol{r}_0)$, or the variation $\mathcal{L}_k = A\mathcal{K}_k$, and methods that choose $\mathcal{L}_k$ to be the Krylov subspace associated with A^T, i.e. $\mathcal{L}_k = \mathcal{K}_k(A^T, \boldsymbol{r}_0)$.

3.1 The BiCGStab algorithm

A biorthogonal system in $\mathcal{R}^m$ is a pair of indexed families of vectors

$$\{\boldsymbol{u}_i\}_{i=0}^n, \boldsymbol{u}_i \in E \subset \mathcal{R}^m \qquad \text{and} \qquad \{\boldsymbol{v}_i\}_{i=0}^n, \boldsymbol{v}_i \in F \subset \mathcal{R}^m,$$

such that

$$\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle = \delta_{i,j},$$

where $\langle \cdot, \cdot \rangle$ is a bilinear mapping and $\delta_{i,j}$ is the Kronecker delta.

The Biconjugate Gradient Stabilized (BiCGStab) algorithm is a member of the second family of Krylov subspace methods. It is the iterative solution method we have used for all of the simulations in this thesis, and so we will give a brief overview of it and its origins in this section. The MinRes algorithm is better than BiCGStab in terms of both speed and memory requirements, but requires both a symmetric A and a symmetric preconditioner, and three out of the four preconditioners we were interested in are nonsymmetric, as is the discrete Oseen operator. Several variants of the GMRES algorithm could have been used instead of BiCGStab in the simulations we have done. The problem with GMRES is the need to keep the intermediate vectors $\boldsymbol{x}_k$, leading to larger memory requirements and restarts. We concluded after some preliminary testing that BiCGStab seemed to outperform GMRES in our test cases, and made results more transparently comparable.

BiCGStab was developed by Vorst in [19]. The algorithm is derived from the Biconjugate Gradient (BCG) and Conjugate gradient Squared (CGS) algorithms, both based on the Lanczos biorthogonalization algorithm. These methods build a pair of biorthogonal bases for the two subspaces

$$\mathcal{K}_k(A, \boldsymbol{v}_0) = span(\boldsymbol{v}_0, A\boldsymbol{v}_0, A^2\boldsymbol{v}_0, \ldots, A^{k-1}\boldsymbol{v}_0)$$

and

$$\mathcal{K}_k(A^T, \boldsymbol{v}_0) = span(\boldsymbol{v}_0, A^T\boldsymbol{v}_0, (A^T)^2\boldsymbol{v}_0, \ldots, (A^T)^{k-1}\boldsymbol{v}_0).$$

The approximate solution $\boldsymbol{x}_k$ to the system $A\boldsymbol{x} = \boldsymbol{b}$ is then the $\boldsymbol{x}_k$ that makes $\boldsymbol{r}_k$ less than a predefined value ϵ in some chosen norm.

The BCG algorithm 3.1 requires A^T. This is often not available, e.g. when A is defined partly as a procedure and not as an explicit matrix. Moreover, the vectors $\boldsymbol{p}_j^\star$

Algorithm 3.1. *Biconjugate Gradient*

```
1  Compute $\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_o$.
2  Choose $\boldsymbol{r}_0^\star$ such that $(\boldsymbol{r}_0, \boldsymbol{r}_0^\star) \neq 0$.
3  Set $\boldsymbol{p}_0 := \boldsymbol{r}_0$, $\boldsymbol{p}_0^\star := \boldsymbol{r}_0^\star$.
4  for $j = 0, 1, \ldots$ until convergence,
5      do $\alpha_j := (\boldsymbol{r}_j, \boldsymbol{r}_0^\star)/(A\boldsymbol{p}_j, \boldsymbol{p}_j^\star)$
6          $\boldsymbol{x}_{j+1} := \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$
7          $\boldsymbol{r}_{j+1} := \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$
8          $\boldsymbol{r}_{j+1}^\star := \boldsymbol{r}_j^\star - \alpha_j A^T \boldsymbol{p}_j^\star$
9          $\beta_{j+1} := (\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1}^\star)/(\boldsymbol{r}_j, \boldsymbol{r}_j^\star)$
10         $\boldsymbol{p}_{j+1} := \boldsymbol{r}_{j+1} + \beta_j \boldsymbol{p}_j$
11         $\boldsymbol{p}_{j+1}^\star := \boldsymbol{r}_{j+1}^\star + \beta_j \boldsymbol{p}_{j+1}^\star$
```

Algorithm 3.2. *Conjugate Gradient Squared*

```
1  Compute $\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_o$.
2  Choose $\boldsymbol{r}_0^\star$ such that $(\boldsymbol{r}_0, \boldsymbol{r}_0^\star) \neq 0$.
3  Set $\boldsymbol{p}_0 := \boldsymbol{u}_0 := \boldsymbol{r}_0$.
4  for $j = 0, 1, \ldots$ until convergence,
5      do $\alpha_j := (\boldsymbol{r}_j, \boldsymbol{r}_0^\star)/(A\boldsymbol{p}_j, \boldsymbol{r}_0^\star)$
6          $\boldsymbol{q}_j := \boldsymbol{u}_j - \alpha_j A\boldsymbol{p}_j$
7          $\boldsymbol{x}_{j+1} := \boldsymbol{x}_j + \alpha_j(\boldsymbol{q}_j + \boldsymbol{u}_j)$
8          $\boldsymbol{r}_{j+1} := \boldsymbol{r}_j - \alpha_j A(\boldsymbol{q}_j - \boldsymbol{u}_j)$
9          $\beta_{j+1} := (\boldsymbol{r}_{j+1}, \boldsymbol{r}_0^\star)/(\boldsymbol{r}_j, r_0^\star)$
10         $\boldsymbol{u}_{j+1} := \boldsymbol{r}_j + \beta_j \boldsymbol{q}_j$
11         $\boldsymbol{p}_{j+1} := \boldsymbol{u}_{j+1} + \beta_j(\boldsymbol{q}_j + \beta_j \boldsymbol{p}_j)$
```

generated with it do not contribute directly to the solution. The CGS algorithm 3.2 was developed as an improvement on BCG, with the aim of not requiring A^T.

The residuals $\boldsymbol{r}_j$ in BCG can be expressed as

$$\boldsymbol{r}_j = \phi_j(A)\boldsymbol{r}_0,$$

where ϕ_j is a polynomial satisfying $\phi_j(0) = 1$. Similarly, the conjugate-direction polynomial π_j is given by

$$\boldsymbol{p}_j = \pi_j(A)\boldsymbol{r}_0.$$

In the same way, $\boldsymbol{r}_j^\star = \phi_j(A^T)\boldsymbol{r}_0^\star$ and $\boldsymbol{p}_j^\star = \pi(A^T)\boldsymbol{r}_0^\star$. Then the scalar α_j in BCG can be written

$$\alpha_j = \frac{(\phi_j^2(A)\boldsymbol{r}_0, \boldsymbol{r}_0^\star)}{(A\pi_j^2(A)\boldsymbol{r}_0, \boldsymbol{r}_0^\star)}.$$

This indicates that if we can find a recursion for $\phi_j^2(A)\boldsymbol{r}_0$ and $\pi_j^2(A)\boldsymbol{r}_0$, then computing the scalars α_j and β_j can be done without having A^T. That is what the CGS algorithm 3.2 does, by computing $\boldsymbol{r}_j$ s.t. it satisfies

$$\boldsymbol{r}_j = \phi_j^2(A)\boldsymbol{r}_0.$$

The problem with CGS is that squaring the residual polynomial can lead to a build up of round-off errors, especially when convergence is irregular. The BiCGStab algorithm was developed to deal with this problem. In BiCGStab, the residual vectors are in the form

$$\boldsymbol{r}_j = \psi_j(A)\phi_j(A)\boldsymbol{r}_0.$$

Here, ϕ_j is the same as in BCG, while ψ_j is a new polynomial defined to stabilize the convergence of the procedure. In each step it is defined as

$$\psi_{j+1}(A) = (1 - \omega_j A)\psi_j(A),$$

Algorithm 3.3. *Biconjugate Gradient Stabilized*

```
1   Compute $\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_o$.
2   Choose $\boldsymbol{r}_0^\star$ such that $(\boldsymbol{r}_0, \boldsymbol{r}_0^\star) \neq 0$.
3   Set $\boldsymbol{p}_0 := \boldsymbol{r}_0$.
4   for $j = 0, 1, \ldots$ until convergence,
5           do $\boldsymbol{v}_j := A\boldsymbol{p}_j$
6              $\rho_j := (\boldsymbol{r}_j, \boldsymbol{r}_0^\star)$
7              $\alpha_j := \rho_j / (\boldsymbol{v}_j, \boldsymbol{r}_0^\star)$
8              $\boldsymbol{s}_j := \boldsymbol{r}_j - \alpha_j \boldsymbol{v}_j$
9              $\boldsymbol{t}_j := A\boldsymbol{s}_j$
10             $\omega_j := (\boldsymbol{t}_j, \boldsymbol{s}_j)/(\boldsymbol{t}_j, \boldsymbol{t}_j)$
11             $\boldsymbol{x}_{j+1} := \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j + \omega_j \boldsymbol{s}_j$
12             $\boldsymbol{r}_{j+1} := \boldsymbol{s}_j - \omega_j \boldsymbol{t}_j$
13             $\beta_j := \frac{(\boldsymbol{r}_{j+1}, \boldsymbol{r}_0^\star)}{(\boldsymbol{r}_j, \boldsymbol{r}_0^\star)} \times \frac{\alpha_j}{\omega_j}$
14             $\boldsymbol{p}_{j+1} := \boldsymbol{r}_{j+1} + \beta_j(\boldsymbol{p}_j - \omega_j \boldsymbol{v}_j)$
```

where ω_j is a scalar. The direction vectors are defined analogously by $\boldsymbol{p}_j = \psi_j(A)\pi_j(A)\boldsymbol{r}_0$. Finding recurrence relations to update $\boldsymbol{r}_{j+1}$ and $\boldsymbol{p}_{j+1}$, and determining ω_j, leads to algorithm 3.3.

The orthogonality property

$$(P_i(A)\boldsymbol{r}_0, Q_j(A^T)\boldsymbol{r}_0^\star) = 0, j < i, \tag{3.3}$$

of the BiCGStab method guarantees, in the absence of breakdown, convergence in at most m iterations. Breakdown is possible even in exact arithmetic. Several of the scalars in algorithm 3.3 may in some iteration be zero. This can lead to e.g. division by zero, or a loss of the orthogonality property (3.3). With finite-precision arithmetic, near-zero coefficients can lead to an increase in numerical errors. The reason for these problems is that, for general matrices, the bilinear form

$$(\boldsymbol{x}, \boldsymbol{y}) = (P(A)\boldsymbol{x}, P(A^T)\boldsymbol{y})$$

does not define an inner product, meaning $(\boldsymbol{r}_j, \boldsymbol{r}_0^\star)$ or $(\boldsymbol{r}_0^\star, A\boldsymbol{p}_j)$ may be zero, or sufficiently close to zero, without the process having converged. A study of the different ways the BiCGStab algorithm can break down is done in [20].

For most problems, we need to use a preconditioner with BiCGStab. Let K be our preconditioner, and $K = K_1 K_2$. Then we instead of solving $A\boldsymbol{x} = \boldsymbol{b}$, solve the equivalent system

$$\hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}},$$

Algorithm 3.4. *Preconditioned Biconjugate Gradient Stabilized*

```
1  Compute $\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_o$.
2  Set $\boldsymbol{p}_0 := \boldsymbol{r}_0^\star := \boldsymbol{r}_0$.
3  Compute $\rho_0 = (\boldsymbol{r}_0, \boldsymbol{r}_0^\star)$
4  for $j = 0, 1, \ldots$ until $r < \epsilon$,
5      do
6          $\boldsymbol{q}_j := B\boldsymbol{p}_j$ $\boldsymbol{v}_j := A\boldsymbol{q}_j$
7          $\alpha_j = \rho_j / (\boldsymbol{r}_0^\star, \boldsymbol{v}_j)$
8          $\boldsymbol{s}_j := \boldsymbol{r}_j - \alpha_j \boldsymbol{v}_j$
9          $\boldsymbol{u}_j := B\boldsymbol{s}_j$
10         $\boldsymbol{t}_j := A\boldsymbol{u}_j$
11         $\gamma := (\boldsymbol{t}_j, \boldsymbol{t}_j)$
12         $\delta := (\boldsymbol{t}_j, \boldsymbol{s}_j)$
13         if $\gamma = 0$ or $\delta = 0$ return "Breakdown, zero inner product"
14         $\omega_j = \delta / \gamma$
15         $\boldsymbol{x}_{j+1} := \boldsymbol{x}_j + \alpha_j \boldsymbol{q}_j + \omega_j \boldsymbol{u}_j$
16         $\boldsymbol{r}_{j+1} := \boldsymbol{s}_j - \omega_j \boldsymbol{t}_j$
17         $\rho_{j+1} := (\boldsymbol{r}_{j+1}, \boldsymbol{r}_0^\star)$
18         $r = \sqrt{(\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1})}$
19         if $r = 0$ return "Breakdown, zero residual"
20         $\beta_j := \frac{\rho_{j+1}}{\rho_j} \times \frac{\alpha}{\omega}$
21         if $\beta = 0$ return "Breakdown, zero beta"
22         $\boldsymbol{p}_{j+1} := \boldsymbol{r}_{j+1} + \beta(\boldsymbol{p}_j - \omega \boldsymbol{v}_j)$
```

with $\hat{A} = K_1^{-1} A K_2^{-1}$, $\hat{\boldsymbol{x}} = K_2 \boldsymbol{x}$ and $\hat{\boldsymbol{b}} = K_1^{-1}\boldsymbol{b}$. In the implementation in cbc.block, $K_2 = I$, i.e. we use left-preconditioning with $K_1^{-1} = B$.

Algorithm 3.4 has several opportunities to break down in each iteration, and in practice BiCGStab breakdowns are a common occurence. Some factors contributing to the likelihood of breakdown is

- The difference between the initial guess and the expected solution $\|\boldsymbol{x} - \boldsymbol{x}_0\|$ is too large. This may be the case when, e.g., simulating a time dependent problem with a too-large time step.
- The preconditioned problem $BA\boldsymbol{x} = B\boldsymbol{b}$ is too ill-conditioned, i.e. the preconditioner is not good enough.

However, when BiCGStab avoids breaking down it is fast and efficient in memory usage, both for symmetric and nonsymmetric systems.

4 Preconditioning

The rate at which most iterative methods, and in particular Krylov subspace methods, converge is related to the spectrum, meaning the set $\lambda(A)$ of eigenvalues of A, or the *condition number* $\kappa(A)$ of A ([21], [22]). The condition number is defined as

$$\kappa(A) = \|A\|\|A^{-1}\|,$$

where $\|\cdot\|$ is some consistent matrix norm. As all matrix norms are equivalent for some given dimension $n \times m$, the condition number of a matrix with respect to two different norms, $\|\cdot\|_a$ and $\|\cdot\|_b$, can only differ by a constant factor. The most commonly used definition of $\kappa(A)$ is the condition number with respect to the spectral norm

$$\begin{aligned} \|A\| = \|A\|_2 &= \max \sigma(A), \\ \Rightarrow \kappa(A) &= \frac{\max \sigma(A)}{\min \sigma(A)}, \end{aligned}$$

where $\sigma(A)$ is the set of singular values of A i.e. the square roots of the eigenvalues of $A^\star A$, where $A^\star$ is the conjugate transpose of A.

If $\kappa(A)$ is large, the iterative method for solving $A\boldsymbol{x} = \boldsymbol{b}$ might diverge, or converge too slowly. The condition number of a discretized PDE typically, and also in the case of the Stokes and Navier-Stokes equations, grows as $1/h^2$. $h > 0$ is proportional to the mesh resolution, i.e. a finer mesh means a smaller h. For these reasons it is necessary to precondition the system to solve it in reasonable time. To precondition the system means to multiply equation (3.1) with a *preconditioner* P.

A commonly used criterium for a good preconditioner P for a matrix A is that P should be *order optimal* with respect to the discretization parameter h. A preconditioner is considered order optimal if:

(i) The condition number of PA is bounded independently of h, i.e. $\kappa(PA) < c_0$ where the constant c_0 is independent of the parameter h.

(ii) Evaluating P on a vector, $P\boldsymbol{x}$, requires $O(n)$ arithmetic operations.

(iii) Storing P requires $O(n)$ bytes.

The above definition of order-optimality is taken from [23].

Another way to express *(i)*, if A and P are both symmetric and positive, is to require P to be spectrally equivalent to A^{-1}. That P is spectrally equivalent to A^{-1} means that there exists constants c_1 and c_2 such that

$$c_1(A^{-1}v, v) \leq (Pv, v) \leq c_2(A^{-1}v, v) \quad \forall v \in \mathbb{R}^n, \tag{4.1}$$

where $(\cdot,\cdot)$ denotes some inner product. If A^{-1} and P are spectrally equivalent then the condition number of the preconditioned system is bounded by c_2/c_1, and if c_1 and c_2 both are independent of h, then κ is independent of the discretization parameter, and a uniform refinement of the mesh will not lead to a more slowly converging iterative solution process. $P = A^{-1}$ satisfies the first condition, but it is typically dense and so fails the second and third. What we seek, in the symmetric and positive case, is some kind of approximation $P \approx A^{-1}$ to the inverse of A that is order optimal in the sense described above.

In the case where A is nonsymmetric and indefinite the above paragraph does not apply. Iterative solution methods with well-known convergence properties, like the mimimum residual method, cannot be applied to such a system. This makes finding a good preconditioner difficult. The spectral norm may also not be the most useful norm to define the condition number by. Instead, we might choose a norm that depends in part on the parameters of the differential operator A is a discretization of, or the dimension n of the discrete system, or both. Which properties of A the preconditioner P should be designed to improve, i.e., what should the preconditioned system PA look like in terms of eigenvalues, singular values, or something else, probably depends crucially on the chosen iterative solution method. Clear answers to this question exists for algorithms like the minimum residual and conjugent gradient methods applied to symmetric, positive systems [24]. However, for general matrices the convergence properties of the most commonly used Krylov subspace methods are not as well understood.

4.1 Preconditioning the time dependent Stokes equations

4.1.1 Block diagonal preconditioner

While we are primarily interested in the (finite element-) discretized version of (2.27), it can be useful to first consider the preconditioning of the continuous problem. Here we will only consider the case of $\partial\Omega_D = \partial\Omega$, with $\boldsymbol{u}\big|_{\partial\Omega_D} = g_D = 0$. We rewrite (2.27) as

$$\begin{aligned} a_\epsilon(\boldsymbol{u},\boldsymbol{v}) - b(\boldsymbol{v},p) &= \langle f,\boldsymbol{v}\rangle \qquad && \boldsymbol{v}\in V, \\ b(\boldsymbol{u},q) &= 0 && q\in Q, \end{aligned} \tag{4.2}$$

where $a_\epsilon(\boldsymbol{u},\boldsymbol{v}) = \langle \boldsymbol{u},\boldsymbol{v}\rangle + \epsilon^2\langle\nabla\boldsymbol{u},\nabla\boldsymbol{v}\rangle$ and $b(\boldsymbol{u},q) = \langle\nabla\cdot\boldsymbol{u},q\rangle$, and we assume $f\in H^{-1}$. We then introduce the operator

$$\mathcal{A}_\epsilon = \begin{pmatrix} I-\epsilon^2\Delta & \nabla \\ \nabla\cdot & 0 \end{pmatrix}, \tag{4.3}$$

where $\mathcal{A}_\epsilon : H_0^1(\Omega) \times L^2(\Omega) \to H^{-1}(\Omega) \times L^2(\Omega)$ is defined by

$$\langle \mathcal{A}_\epsilon(\boldsymbol{u}, p), (\boldsymbol{v}, q)\rangle = \langle \boldsymbol{u}, \boldsymbol{v}\rangle + \epsilon^2 \langle \nabla \boldsymbol{u}, \nabla \boldsymbol{v}\rangle - \langle \nabla\cdot \boldsymbol{v}, p\rangle + \langle \nabla \cdot \boldsymbol{u}, q\rangle, \forall (\boldsymbol{v}, q) \in H_0^1(\Omega) \times L^2(\Omega).$$

Here, $\langle \cdot, \cdot \rangle$ has been used both to denote duality pairings and L^2 inner products.

As ϵ tends to zero, $I - \epsilon^2 \Delta$ will be dominated by the identity operator. This is a problem, as we would like to have bounds on the norms of $\mathcal{A}_\epsilon$ and $\mathcal{A}_\epsilon^{-1}$ which are independent of ϵ. To achieve this, we will define $\mathcal{A}_\epsilon$ on the space

$$\mathcal{X}_\epsilon = (L^2(\Omega) \cap \epsilon \cdot H_0^1(\Omega))^n \times ((H^1(\Omega) \cap L_0^2(\Omega)) + \epsilon^{-1} L_0^2(\Omega)), \tag{4.4}$$

where $L_0^2(\Omega)$ is the space of L^2 functions on Ω with mean value zero. The norm of the space $(L^2(\Omega) \cap \epsilon \cdot H_0^1(\Omega))^n$ is

$$\|\boldsymbol{v}\|_V = (\|\boldsymbol{v}\|_{L^2}^2 + \epsilon^2 \|\nabla \boldsymbol{v}\|_{L^2}^2)^{\frac{1}{2}}$$

Note that for $\epsilon > 0$, the space $\mathcal{X}_\epsilon$ is equal to $H_0^1(\Omega) \times L^2(\Omega)$ as a set, and therefore $\mathcal{A}_\epsilon$ is well defined on $\mathcal{X}_\epsilon$. If X and Y are Hilbert spaces, both continuously contained in some larger Hilbert spaces, then $X + Y$ and $X \cap Y$ are also Hilbert spaces. Additionaly, if $X \cap Y$ is dense in both X and Y then its dual is

$$(X \cap Y)^* = X^* + Y^*. \tag{4.5}$$

These results concerning Hilbert spaces can be found in [25]. Due to (4.5), the dual of $\mathcal{X}_\epsilon$ is

$$\mathcal{X}_\epsilon^* = (L^2(\Omega) + \epsilon^{-1} \cdot H^{-1}(\Omega))^n \times ((H^1(\Omega) \cap L_0^2(\Omega))^* \cap \epsilon \cdot L_0^2(\Omega)). \tag{4.6}$$

Having established this, we will now consider how to precondition (4.2). A suitable preconditioner $\mathcal{B}_\epsilon : \mathcal{X}_\epsilon^* \to \mathcal{X}_\epsilon$ is one that maps $\mathcal{X}_\epsilon^*$ to $\mathcal{X}_\epsilon$, so that the composition

$$\mathcal{B}_\epsilon \mathcal{A}_\epsilon : \mathcal{X}_\epsilon \to \mathcal{X}_\epsilon^* \to \mathcal{X}_\epsilon \tag{4.7}$$

maps $\mathcal{X}_\epsilon$ to itself. The operators

$$(I - \epsilon^2 \Delta)^{-1} : (L^2 + \epsilon^{-1} H_0^{-1})^n \to (L^2 \cap \epsilon \cdot H_0^1)^n$$

and

$$\epsilon^2 I + (-\Delta)^{-1} : ((H^1 \cap L_0^2)^* \cap \epsilon \cdot L_0^2) \to (H^1 \cap L_0^2 + \epsilon^{-1} L_0^2)$$

have the properties we need, so we conclude that

$$\mathcal{B}_\epsilon = \begin{pmatrix} (I - \epsilon^2 \Delta)^{-1} & 0 \\ 0 & \epsilon^2 I + (-\Delta)^{-1} \end{pmatrix} \tag{4.8}$$

fulfills the requirement stated in 4.7.

The mapping properties of the operator $\mathcal{A}_\epsilon$, and the consequenses of those properties for the preconditioning of (2.27), are thoroughly examined in [26] and [6], and we refer to those papers for a comprehensive study of the problem and more detailed derivation of (4.8).

The discrete analogue of (4.3) is a block matrix on the form

$$A_\epsilon = \begin{bmatrix} F & G \\ D & 0 \end{bmatrix}. \tag{4.9}$$

The discretization of (4.8) is a block diagonal matrix on the form

$$B_\epsilon = \begin{bmatrix} F^{-1} & 0 \\ 0 & \hat{S}^{-1} \end{bmatrix}. \tag{4.10}$$

Here $F = M + \epsilon^2 K$, where M is a mass matrix, the finite element identity operator, and K is a stiffness matrix, the finite element Laplace operator. D is the discrete divergence, G is the discrete gradient, and $\hat{S}^{-1} = K^{-1} + \epsilon^2 M^{-1}$. Note the subsitution of p with $-p$, making (4.9) symmetric ($G^T = D$).

4.1.2 Projection method based preconditioner

Another idea for a preconditioner for the discretized time-dependent Stokes equations is explored in [7]. Take a discrete saddle-point system on the form

$$\begin{bmatrix} F & G \\ -D & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix}. \tag{4.11}$$

Here $F = \frac{\rho}{\Delta t} I - \mu L$ is the discrete operator for $\frac{\rho}{\Delta t} I - \mu \Delta$, D for $\nabla\cdot$, and G for ∇. Equation (4.11) is clearly equivalent to (2.32) if $D^T = G$.

The following solution algorithm is presented in [7]:

(i) Solve for an intermediate velocity $\boldsymbol{u}^\star$

$$F\boldsymbol{u}^\star = \boldsymbol{f}. \tag{4.12}$$

(ii) Project $\boldsymbol{u}^\star$ to the divergence free space by solving

$$\begin{aligned} \rho \frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^\star}{\Delta t} &= -G\phi \\ -D\boldsymbol{u}^{k+1} &= 0, \end{aligned} \tag{4.13}$$

which leads to the following Poisson equation for the pressure:

$$-L\phi = -\frac{\rho}{\Delta t} D\boldsymbol{u}^\star \tag{4.14}$$

(iii) Update the velocity and correct the pressure term by

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^\star - \frac{\Delta t}{\rho} G\phi, \tag{4.15}$$

$$p^{k+1} = \phi - \frac{\Delta t}{\rho}\mu L\phi. \tag{4.16}$$

This algorithm can be written in matrix form as

$$\tilde{P}^{-1} = \begin{bmatrix} I & -G \\ 0 & \frac{\rho}{\Delta t} I - \mu L \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -(L)^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -D & -I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}, \tag{4.17}$$

or, combining the two leftmost matrices, as the following:

$$P^{-1} = \begin{bmatrix} I & GL^{-1} \\ 0 & \hat{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -D & -I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}, \tag{4.18}$$

where $\hat{S}^{-1} = \frac{\rho}{\Delta t}(-L)^{-1} + \mu I$. The idea is then to use this matrix as a preconditioner for the discretized time dependent Stokes equations.

The formulation of (4.18) that we will use is

$$P_\epsilon = \begin{bmatrix} I & GK^{-1} \\ 0 & \hat{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ D & -I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}. \tag{4.19}$$

This formulation is a translation of (4.18) to the finite element formulation from section 2.2.4, with the naming conventions we used for (4.9) and (4.10). That is, K is a stiffness matrix, $\hat{S}^{-1} = \nu M^{-1} + \frac{\rho}{\Delta t} K^{-1}$ is the same operator present in (4.10), G is the (finite element) negative discrete gradient and D the (finite element) discrete divergence. The difference in the parameters in $\hat{S}^{-1}$ here is due to the momentum equation not being scaled by Δt as it was in section 4.1.1 and $\frac{1}{\rho}$ as it was in 2.1.

Note that the definition of A that we use in this thesis is, in contrast to the coefficient matrix in equation (4.11),

$$A = \begin{bmatrix} F & G \\ D & 0 \end{bmatrix}. \tag{4.20}$$

This difference in the sign of the continuity equation leads to both occurrences of D in (4.19) also having opposite sign of what they have in (4.18), with the result that the operator $P_\epsilon A$ here looks different from the same in [7].

4.1.3 Yosida method based preconditioner

The third preconditioner we consider is one based on the Yosida method introduced in [27]. Take a saddle point problem on the form

$$A\boldsymbol{x} = \begin{bmatrix} F & G \\ D & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1^{n+1} \\ \boldsymbol{b}_2^{n+1} \end{bmatrix}$$

and observe that a block LU factorization of A is

$$A = \begin{bmatrix} F & 0 \\ D & -DF^{-1}G \end{bmatrix} \begin{bmatrix} I & F^{-1}G \\ 0 & I \end{bmatrix}. \tag{4.21}$$

Approximating F^{-1} by a matrix H_1 in the first block, and another matrix H_2 in the second, an inexact factorization of A is

$$\tilde{A} = \begin{bmatrix} F & 0 \\ D & -DH_1G \end{bmatrix} \begin{bmatrix} I & H_2G \\ 0 & I \end{bmatrix} = \begin{bmatrix} F & FH_2G \\ D & D(H_2 - H_1)G \end{bmatrix}. \tag{4.22}$$

Different choices of H_1 and H_2 will lead to different schemes. In the Yosida method,

$$H_1 = \Delta t M^{-1}, \quad H_2 = F^{-1}.$$

This leads to the following pressure correction scheme:

$$\begin{aligned} F\tilde{\boldsymbol{u}}^{n+1} &= \boldsymbol{b}_1^{n+1} && \textit{Tentative velocity,} \\ -\Delta t DM^{-1}Gp^{n+1} &= \boldsymbol{b}_2^{n+1} - D\tilde{\boldsymbol{u}}^{n+1} && \textit{Pressure computation,} \\ F\boldsymbol{u}^{n+1} &= F\tilde{\boldsymbol{u}}^{n+1} - Gp^{n+1} && \textit{Velocity correction.} \end{aligned}$$

The scheme 4.23 expressed in matrix form is

$$A_Y = \begin{bmatrix} F & 0 \\ D & -\Delta t DM^{-1}G \end{bmatrix} \begin{bmatrix} I & F^{-1}G \\ 0 & I \end{bmatrix} = \begin{bmatrix} F & G \\ D & D(F^{-1} - \Delta t M^{-1})G \end{bmatrix}. \tag{4.23}$$

The matrix 4.23 was first used as a preconditioner by Deparis et al. in [2]. In implementation, we use a factorization of the inverse of 4.23, i.e.

$$\begin{aligned} A_Y^{-1} = Y_1 &= \Big(\begin{bmatrix} F & 0 \\ D & -\Delta t DM^{-1}G \end{bmatrix} \begin{bmatrix} I & F^{-1}G \\ 0 & I \end{bmatrix} \Big)^{-1} \\ &= \begin{bmatrix} I & F^{-1}G \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} F & 0 \\ D & -\Delta t DM^{-1}G \end{bmatrix}^{-1} \\ &= \begin{bmatrix} I & -F^{-1}G \\ 0 & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ (\Delta t DM^{-1}G)^{-1}DF^{-1} & (-\Delta t DM^{-1}G)^{-1} \end{bmatrix} \\ &= \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & -G \\ 0 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -\tilde{S}_1^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -D & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}, \end{aligned} \tag{4.24}$$

if

$$\tilde{S}_1^{-1} = (\Delta t D M^{-1} G)^{-1}. \tag{4.25}$$

In practice, using the explicit inverse of M is not feasible. Instead, we will follow [2] in approximating M with $\tilde{M}$, where $\tilde{M}_{i,i} = \sum_j |M_{i,j}|$, i.e. a diagonal matrix with the row sums of M on the diagonal.

In addition to the lumped mass matrix approximation, we will look at a variant where we use the inverse diagonal of F in the Schur complement, i.e. we set

$$\tilde{S}_2^{-1} = (D\tilde{F}^{-1}G)^{-1}, \tag{4.26}$$

where $\tilde{F}$ is a diagonal matrix with $\tilde{F}_{i,i} = F_{i,i}$. This variation we will refer to as Y_2. The reason why we include the variant is that it is an easy variation to implement, and the performance of Y_2 might scale better with regard to ν as that parameter contributes to $\tilde{F}$, but not to $\Delta t\tilde{M}$.

4.1.4 Analysis of the preconditioners

Our analysis of the B_ϵ, P_ϵ and Yosida preconditioners will be limited to examining the eigenvalues of the preconditioned systems $B_\epsilon A$, $P_\epsilon A$, $Y_1 A$ and $Y_2 A$. As Y_1 and Y_2 only differ in the choice of Schur complement approximation, we study both as one and refer to that as Y_ϵ. Similarly, we refer to both $\tilde{S}_1^{-1}$ and $\tilde{S}_2^{-1}$ by $\tilde{S}^{-1}$ whenever what we write applies to both.

To study the eigenvalues of

$$B_\epsilon A = \begin{bmatrix} F^{-1} & 0 \\ 0 & \hat{S}^{-1} \end{bmatrix} \begin{bmatrix} F & G \\ D & 0 \end{bmatrix}, \tag{4.27}$$

we consider the generalized eigenvalue problem

$$\begin{bmatrix} F & G \\ D & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & 0 \\ 0 & \hat{S} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix}. \tag{4.28}$$

We assume here that $\hat{S}^{-1}$ is invertible. This problem is examined in Elman, Silvester & Wathen in [3], where it is shown that if $\hat{S}$ is the exact schur complement, then all the eigenvalues λ in (4.28) lie in

$$\{\frac{1}{2} - \frac{\sqrt{5}}{2}\} \cup \{1\} \cup \{\frac{1}{2} + \frac{\sqrt{5}}{2}\}.$$

That is, the eigenvalues are $\lambda = 1$, with multiplicity $n_u - n_p$, and $\lambda = \frac{1}{2} \pm \frac{\sqrt{5}}{2}$, each with multiplicity n_p. For the Schur complement approximation $\hat{S}^{-1} = K^{-1} + \epsilon^2 M$, consider the following argument (adapted from p. 293 in [3]): Eliminate $\boldsymbol{u}$ in the second equation of (4.28) using the first, to obtain

$$DF^{-1}Gp = \lambda(\lambda - 1)\hat{S}p = \gamma\hat{S}p.$$

Now, for each eigenvalue γ of $\hat{S}^{-1}S$, there is a pair of eigenvalues of (4.28) given by

$$\lambda = \frac{1}{2} - \frac{1}{2}\sqrt{1+4\gamma} \quad \text{and} \quad \lambda = \frac{1}{2} + \frac{1}{2}\sqrt{1+4\gamma}.$$

Now, in [7] it is shown that, for a certain finite difference discretization, the discrete operator $\hat{S}^{-1}S$ has eigenvalues

$$\beta^2 < \lambda(\hat{S}^{-1}S) < 1,$$

where β is a constant that does not depend on the mesh resolution. This result is not necessarily applicable to the finite element context of this thesis, but in chapter 6 we will examine the eigenvalues of $\hat{S}^{-1}S$ numerically.

The matrix $P_\epsilon A$ is

$$P_\epsilon A = \begin{bmatrix} I & GK^{-1} \\ 0 & \hat{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ D & -I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} F & G \\ D & 0 \end{bmatrix}. \tag{4.29}$$

If we do the multiplications, this can be written

$$P_\epsilon A = \begin{bmatrix} I & (I + GK^{-1}D)F^{-1}G \\ 0 & \hat{S}^{-1}S \end{bmatrix}. \tag{4.30}$$

Similarly, the system $Y_\epsilon A$ is

$$Y_\epsilon A = \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & -G \\ 0 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -\tilde{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -D & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} F & G \\ D & 0 \end{bmatrix}, \tag{4.31}$$

which, carrying out the multiplications, we can write as

$$Y_\epsilon A = \begin{bmatrix} I & F^{-1}G(I - \tilde{S}^{-1}S) \\ 0 & \tilde{S}^{-1}S \end{bmatrix}. \tag{4.32}$$

As the determinant of a block triangular matrix is the product of the determinants of the diagonal blocks, each eigenvalue of (4.32) and (4.30) is either 1, or an eigenvalue of $\tilde{S}^{-1}S$ or $\hat{S}^{-1}S$, respectively. The formal analysis of the eigenvalues $\lambda(\tilde{S}^{-1}S)$ of $\tilde{S}^{-1}S$ lies outside the scope of this thesis. They will instead be examined numerically in chapter 6.

The spectral norm of the matrix (4.30), defined as

$$\|P_\epsilon A\|_2 = \sup_{x\in\mathbb{R}^n} \frac{\|P_\epsilon Ax\|_2}{\|x\|_2} = \max \sigma(P_\epsilon A),$$

will depend on the operator in the (1,2)-block, $(I + GK^{-1}D)F^{-1}G$, in addition to $\hat{S}^{-1}S$. We note that this operator is different from the (1,2)-block operator in (4.18) from [7].

The spectral norm of the matrix (4.32) depends only on how close $\tilde{S}^{-1}S$ is to identity, or in other words how well $\tilde{S}^{-1}$ approximates S^{-1}. Whether the value of this norm is important for the convergence of iterative methods applied to the preconditioned system will be investigated through numerical experiments in chapter 6, but we note that in a paper by Nachtigal et al. [21] it is argued that for the CGS algorithm (3.2) convergence does not depend on singular values, but on eigenvalues or pseudoeigenvalues. Further, the BiCGStab algorithm (3.4) is based on CGS.

The δ-pseudospectrum of a matrix A is the set of point $z \in \mathbb{C}$ which are eigenvalues of some matrix $A + E$ with $\|E\| <= \delta$, for some $\delta > 0$ [21]. We take a cursory look at some pseudoeigenvalues of the preconditioned operators in chapter 5.

4.2 Preconditioning the modified Oseen problem

The linearized and finite element-discretized modified Oseen problem (2.31) is a linear saddle point problem on the form

$$A\boldsymbol{x} = \begin{bmatrix} F & G \\ D & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix}, \tag{4.33}$$

where G is the discrete gradient, D is the discrete divergence and $F = \frac{1}{\Delta t}M + \nu K + C(\boldsymbol{u}_1)$ is the discretization of (2.30). Here M is a finite element mass matrix, K is a finite element stiffness matrix and $C(\boldsymbol{u}_1)$ the discretized convective term. As in (4.9) we have subsituted $-p$ for p.

It is easily verifiable that the inverse of A in (4.33) is given by

$$A^{-1} = \begin{bmatrix} F^{-1} + F^{-1}G(-DF^{-1}G)DF^{-1} & -F^{-1}G(-DF^{-1}G)^{-1} \\ (DF^{-1}G)^{-1}DF^{-1} & -(DF^{-1}G)^{-1} \end{bmatrix} \tag{4.34}$$

$$= \begin{bmatrix} F^{-1} - F^{-1}GS^{-1}DF^{-1} & F^{-1}GS^{-1} \\ S^{-1}DF^{-1} & -S^{-1} \end{bmatrix}. \tag{4.35}$$

For our problems 2.27 and 2.28, the action of the inverse F^{-1} can be approximated well with e.g. an algebraic multigrid method, while the inverse Schur complement

S^{-1} poses some difficulty. Several approximations of S have been suggested, e.g. in [28], [29].

Let $F = \frac{1}{\Delta t}M + \nu K + C(\boldsymbol{u}_1)$. In the B_ϵ and P_ϵ preconditioners, the heuristic we used to approximate $S^{-1} = (DF^{-1}G)^{-1}$ for the modified Oseen problem is, assuming commutativity of the continuous operators ∇ and $\nabla\cdot$, and ignoring the fact that they operate on different fields, to write

$$\begin{aligned}
-S^{-1} &= (-DF^{-1}G)^{-1} \\
&= (-D(\frac{1}{\Delta t}M + \nu K + C(\boldsymbol{u}_1))^{-1}G)^{-1} \\
&\approx (-\nabla \cdot (\frac{1}{\Delta t}I + \nu(-\Delta) + \boldsymbol{u}_1 \cdot \nabla)^{-1}\nabla)^{-1} \\
&\approx (-\nabla \cdot \nabla(\frac{1}{\Delta t}I + \nu(-\Delta) + \boldsymbol{u}_1 \cdot \nabla)^{-1})^{-1} \\
&\approx (-\Delta)^{-1}(\frac{1}{\Delta t}I + \nu(-\Delta) + \boldsymbol{u}_1 \cdot \nabla) \\
&\approx \frac{1}{\Delta t}(-\Delta)^{-1} + \nu(-\Delta)^{-1}(-\Delta) + (-\Delta)^{-1}\boldsymbol{u}_1 \cdot \nabla \\
&\approx \frac{1}{\Delta t}(-\Delta)^{-1} + \nu I + (-\Delta)^{-1}\boldsymbol{u}_1 \cdot \nabla \\
&\approx \frac{1}{\Delta t}K^{-1} + \nu M^{-1} + K^{-1}C(\boldsymbol{u}_1) = \hat{S}^{-1}.
\end{aligned}$$

The argument above is motivated partly by a similar approach being used in [28] and the operator preconditioning arguments from [6], reproduced in section 4.1.1.

We have compared the same preconditioners we used for the time dependent Stokes equations for 4.33, and we restate them here:

- The block triangular preconditioner described in 4.1,

$$B_O = \begin{bmatrix} F^{-1} & 0 \\ 0 & \hat{S}^{-1} \end{bmatrix}. \tag{4.36}$$

- The projection method based preconditioner described in 4.1.2,

$$P_O = \begin{bmatrix} I & G(K)^{-1} \\ 0 & \hat{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ D & -I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}. \tag{4.37}$$

- The Yosida preconditioner (4.24) described in section 4.1.3,

$$Y_1 = \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & -G \\ 0 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -\tilde{S}_1^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -D & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}. \tag{4.38}$$

- The variant of the Yosida preconditioner above that was discussed in section 4.1.3,

$$Y_2 = \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & -G \\ 0 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -\tilde{S}_2^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -C & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix}. \tag{4.39}$$

To summarize, the pressure Schur complement approximations in (4.36)-(4.39) are

$$\hat{S}^{-1} = \frac{1}{\Delta t} K_p^{-1} + \nu M_p^{-1} + K_p^{-1} C(\boldsymbol{u}_1), \tag{4.40}$$

$$\tilde{S}_1^{-1} = (\Delta t D \tilde{M}^{-1} G)^{-1}, \tag{4.41}$$

$$\tilde{S}_2^{-1} = (D \tilde{F}^{-1} G)^{-1}, \tag{4.42}$$

where $\tilde{M}$ is a lumped mass matrix and $\tilde{F}$ is a diagonal matrix with $\tilde{F}_{i,i} = F_{i,i}$.

5 Description of numerical experiments

A note on notation used in this chapter: A is a partial differential operator discretized with the finite element method, either the time dependent Stokes equations or the modified Oseen problem, and P^{-1} is a preconditioner. K and M_p means a stiffness matrix on u and a mass matrix on p, respectively. As previously, D is the divergence matrix and G is the gradient matrix. We will by $\mathbb{S}^{-1}$ denote an approximation to the inverse pressure Schur complement.

When we talk about the *performance* of a preconditioner P^{-1}, we will be referring to the average number of BiCGStab iterations to solve a time step of a flow problem when the discretized problem was preconditioned by P^{-1}. In this context, better performance means lower iteration count. The reason we do not measure performance by time expenditure is that the simulations were done on several different shared machines. By preconditioner *robustness* we mean insensitivity to both the viscosity parameter ν and the maximum velocity magnitude $v_{\max}$.

In all the numerical work presented in this chapter and the next we have used continuous Galerkin elements of degree 2 for the velocity $\boldsymbol{u}$, and continuous Galerkin elements of degree 1 for the pressure p. We have not done any testing of parallell implementations; each simulation was done on a single processor. Also note that we have not done any monitoring of memory usage.

For reference we state the discretized differential operators. They are

$$A_S = \begin{bmatrix} \frac{1}{\Delta t}M + \nu K & G \\ D & 0 \end{bmatrix}, \tag{5.1}$$

$$A_O = \begin{bmatrix} \frac{1}{\Delta t}M + \nu K + C(\boldsymbol{u}_1) & G \\ D & 0 \end{bmatrix}, \tag{5.2}$$

for the time dependent Stokes equations and Oseen equations, respectively. We will sometimes drop the subscript distinguishing the two operators when the context ensures there will be no ambiguity. $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity.

The preconditioners (4.10), (4.19), (4.38) and (4.39) were compared in several different ways:

- Calculation of the condition number

  $$\kappa_1 = \kappa_1(P^{-1}A) = \|P^{-1}A\|\|(P^{-1}A)^{-1}\|$$

  of $P^{-1}A$, for various mesh sizes and parameter values. This number will in general differ for different choices of norm. We chose to find the 2-norm condition number, given by $\frac{\sigma_{\max}}{\sigma_{\min}}$, i.e. the ratio of the largest singular value of $P^{-1}A$ and the smallest.

- Calculation and plotting of the non-unitary eigenvalues of $P^{-1}A$.
- Tabulation of the ratio of the largest eigenvalue and the smallest eigenvalue (by moduli). We will denote this ratio by $\kappa_2 = \kappa_2(P^{-1}A)$.
- Comparison of performance and robustness for various mesh sizes and parameter values, on two model problems on the unit square.
- Performance in a less idealized setting, with physical parameters.

The analysis and results in [6] for the B_ϵ preconditioner, and preliminary testing, suggested that the performance of the preconditioners may be more dependent on κ_2 than κ_1. For this reason we are interested in both.

5.1 Condition number computations

In all the test cases, the discrete system and the preconditioners were built with dolfin [8], using cbc.block [9], and subsequently the matrices were exported to GNU Octave [30] where we calculated the eigenvalues $\lambda(P^{-1}A)$,

$$\kappa_1(P^{-1}A) = \frac{\max \sigma(P^{-1}A)}{\min \sigma(P^{-1}A)}, \tag{5.3}$$

$$\kappa_2(P^{-1}A) = \frac{\max_{\lambda \in \lambda(P^{-1}A)} |\lambda|}{\min_{\lambda \in \lambda(P^{-1}A), \lambda \neq 0} |\lambda|}. \tag{5.4}$$

The calculations of the eigenvalues $\lambda(P^{-1}A)$ were done by solving the eigenvalue problem $P^{-1}Ax = \lambda Ix$, using the built-in QZ decomposition algorithm from Octave. Finding the eigenvalues through instead solving the generalized eigenvalue problem $Ax = \lambda Px$ was considered, but ultimately rejected. The reason being that the approximations

$$\begin{aligned} \hat{S}_S^{-1} &= \frac{1}{\Delta t} K_p^{-1} + \nu M_p^{-1} && \text{(Stokes)} \\ \hat{S}_O^{-1} &= \frac{1}{\Delta t} K_p^{-1} + \nu M_p^{-1} + K_p^{-1} C(\boldsymbol{u}_1) && \text{(Oseen),} \end{aligned}$$

of the inverse schur pressure complement in the preconditioners P_ϵ and B_ϵ would have to be numerically inverted, something we wanted to avoid to minimize errors due to finite-precision arithmetic. Although this would not be an issue when calculating the eigenvalues of the Yosida-preconditioned system, in the interest of consistency we opted for the same approach in that case. We will omit the subscripts distinguishing the two variants of $\hat{S}^{-1}$ whenever the context ensures there will be no ambiguity. The singular values $\sigma(P^{-1}A)$ were calculated by way of the Octave implementation of the singular value decomposition.

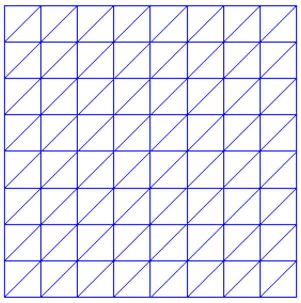


Figure 4: 8-by-8 unit square mesh

All numerically exact eigenvalue computations were done on an N-by-N triangle mesh on the unit square, with two different boundary conditions. We will in some text and figures refer to e.g. the mesh in figure 4 as having a mesh resolution of $h = 2^{-\log_2 N}$. The boundary conditions used were

(i) Lid-driven cavity: $\boldsymbol{u}(x, 1) = (s_1(t), 0)$, no-slip for the rest of the boundary. With these boundary conditions, p is only determined up to a constant, so the matrix A is rank deficient by 1. For this reason we disregard the zero eigenvalue of P^{-1}, and instead use the second smallest (by modulo) to compute the condition number. For the same reason the stiffness matrix K_p is not invertible, so we added a mass matrix, and used $(K_p + M_p)^{-1}$ in place of K_p^{-1} in the two preconditioners where K_p^{-1} is used.

(ii) Velocity driven pipe flow: $\boldsymbol{u}(0, y) = (s_2(t)4y(1 - y), 0)$, $\boldsymbol{u}(x, 0) = \boldsymbol{u}(x, 1) = (0, 0)$ and $p(1, y) = 0$.

The Schur complement approximations $\tilde{S}_1^{-1} = (\Delta t D\tilde{M}^{-1}G)^{-1}$ and $\tilde{S}_2^{-1} = (D\tilde{F}^{-1}G)^{-1}$, used in, respectively, the Y_1 and Y_2 preconditioners, were the source of some difficulties in implementation. Both required stabilization by a mass matrix in the absence of boundary conditions on p, i.e. for the lid-driven cavity setup we replaced $\tilde{S}$ by $\tilde{S} + M$. In the two dimensional pipe flow setup, this was also necessary, as the boundary conditions on p led to D being rank deficient to a degree equal to the number of nodes on the outflow boundary. The changes we made to these operators to avoid the difficulties in implementation we experienced were not motivated by theory, but by experience and trial and error.

We set $s_1(t) = s_2(t) = 1$ and $\Delta t = 1$, and calculated $\lambda(P^{-1}A)$, $\kappa_1(P^{-1}A)$ and $\kappa_2(P^{-1}A)$ for each combination of equation, boundary conditions, preconditioner, $N \in \{2, 4, 8, 16, 32\}$ and $\nu \in \{1, 0.1, 0.01, 0.005, 0.001\}$.

If we calculate the Reynolds number for the lid-driven cavity case as $\text{Re} = \frac{\rho v L}{\mu} = \frac{vL}{\nu}$, where v is the maximum velocity and L the characteristic length (here the length of the lid), we get $\mathrm{Re}_{ldc} = \nu^{-1}$.

For the velocity driven pipe flow we calculate the Reynolds number as $\mathrm{Re} = \frac{\rho Q L}{\mu A} = \frac{QL}{\nu A}$. Here, $L = D$ is the diameter of the 2D pipe, Q is the flux through the inlet given by $\int_{inlet} u \cdot n$, where n is the normal vector to the inlet, and A the area of the inlet. This gives $\mathrm{Re}_{pipe} = \frac{2}{3}\nu^{-1}$.

The convective term $\boldsymbol{u}_1 \cdot \nabla \boldsymbol{u}$ in the Oseen equations contain the velocity from the previous time step. As this initial data we use the velocity from the solution of

$$\begin{aligned} \nu\langle \nabla \boldsymbol{u}, \nabla \boldsymbol{v}\rangle + \langle p, \nabla \cdot \boldsymbol{v}\rangle &= 0, \\ \langle q, \nabla \cdot \boldsymbol{u}\rangle &= 0, \end{aligned}$$

with the same boundary conditions. This choice of initial data is most likely not ideal, especially for the lid-driven cavity boundary conditions. When the viscosity ν is small the flow is not fully developed when we do the calculations.

$P^{-1}A$ is not normal, so the pseudospectra $\sigma_\delta(P^{-1}A)$, defined in section 4.1.4, might be relevant to the performance of the preconditioners(cf. [21]). To investigate the pseudospectra of the preconditioners, we calculated $\lambda(P^{-1}A_{\mathrm{Oseen}} + E)$, with E being an n-by-n matrix with each $E_{i,j}$ a random number in the interval $(0, 2\delta/n)$ for $\delta = 10^{-3}, 10^{-4}, 10^{-5}$. With this defininitition, $\|E_\delta\| \approx 10^{-3}, 10^{-4}, 10^{-5}$. The boundary conditions used for these calculations were those of lid-driven cavity flow.

To measure how much the eigenvalues of $P^{-1}A$ were perturbed by adding the random matrix E, we sorted both the perturbed and non-perturbed eigenvalues according to moduli, calculated the 2-norm of the difference of the two vectors of eigenvalues, and scaled the result by the norm of the vector of eigenvalues of the original matrix. For each preconditioner the resulting value γ was approximately constant in δ, but inversely proportional to ν. Y_2A had the largest such difference, with $\gamma = 0.00389$ when $\nu = 0.001$ and $\delta = 0.001$ and $\gamma = 0.00335$ when $\nu = 0.001$ and $\delta = 10^{-5}$. Scatterplots of the calculated pseudoeigenvalues were sufficiently indistinguishable to the scatterplots of eigenvalues that we do not include them.

A thorough investigation into the pseudospectra of $P^{-1}A$, which would include calculating the size of the subset of $\mathbb{C}$ where the δ-pseudoeigenvalues of $P^{-1}A$ are found, would require large amounts of time and computational power, and lies outside of the scope of this thesis.

5.2 Simulations on the unit square

In this section we will describe the experiments done to compare the performance of the preconditioners when we vary the resolution of the mesh and the Reynolds number. For all these simulations we solved the preconditioned linear system with the biconjugate gradient stabilized method (BiCGStab), specifically the implementation (3.4) of BiCGStab in cbc.block. The convergence criterion for BiCGStab was $\|r_n\|_2 < 10^{-14}$ in all cases.

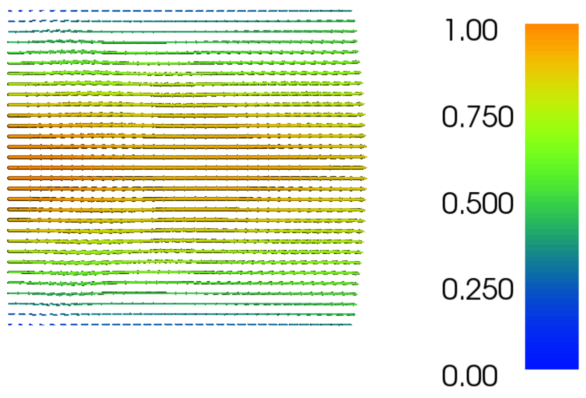


Figure 5: Developed flow at t=2, 2D pipe flow

The inverse stiffness matrices in the preconditioners were approximated with algebraic multigrid (BoomerAMG) from the Hypre library through PETSc, and the inverse mass matrices with successive over-relaxation (SOR) from PETSc, in both cases using the interface of cbc.block [9]. For details on PETSc and the implementation of these preconditioners, we refer to [31].

Two sets of boundary conditions were used for these computations:

(i) Lid-driven cavity: $\boldsymbol{u}(x,1) = (s_1(t), 0)$, no-slip for the rest of the boundary.

(ii) Velocity driven pipe stream: $\boldsymbol{u}(0,y) = (s_2(t)4y(1-y), 0)$, $\boldsymbol{u}(x,0) = \boldsymbol{u}(x,1) = (0,0)$ and $p(1,y) = 0$.

The same adjustments to certain operators, described in section 5.1, were done here to deal with rank deficiencies.

For both the preconditioned time dependent Stokes equations and the preconditioned modified Oseen problem we used a cyclically time-varying boundary condition, setting

$$s_1(t) = s_2(t) = 0.5 + 0.5\cos^2(\pi t). \tag{5.5}$$

The initial condition in all simulations was the solution of the steady state Stokes flow, with the same parameters and boundary conditions. Each simulation consisted

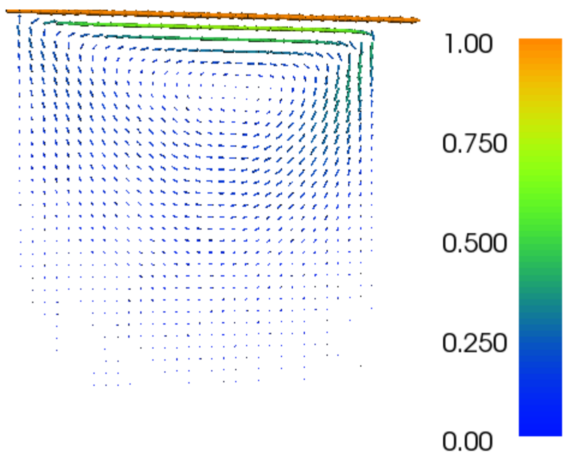


Figure 6: Developed flow at t=2, lid-driven cavity

of 200 time steps with $\Delta t = 0.01$. The numbers we report from each simulation are the average number of BiCGStab iteration before convergence, taken over the whole simulation. We experimented with running the simulation for a time before collecting data, but doing so did not significantly alter results. This was done for each combination of equation, preconditioner, $\nu = 1, 0.1, 0.01, 0.005, 0.001$ and $N = 16, 32, 64, 128, 256$.

For the preconditioners (4.10) and (4.19), on the time dependent Stokes equations, using a smaller time step is equivalent to scaling the viscosity ν. We chose to use the same parameters across all simulations to ease implementation and comparison of results.

5.3 Arterial blood flow simulations

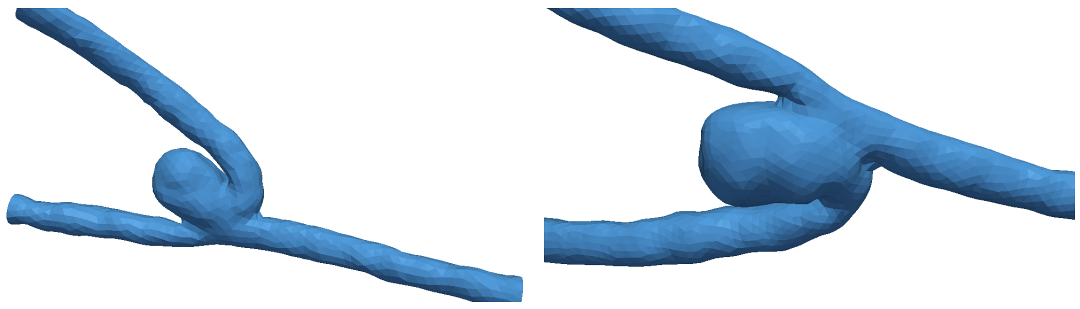


Figure 7: Mesh of cerebral aneurysm

In addition to the experiments detailed above, we also simulated blood flow through a cerebral artery with an aneurysm. The mesh used was based on an MRA image of a surgically created bifurcation aneurysm in a beagle, taken from the study [32]. Boundary conditions were no-slip for the artery walls, zero pressure at the outlets, and a time-varying parabolic velocity profile at the inlet. The inlet velocity profile was generated with code from cbcflow [33]. As initial condition we used steady Stokes flow. The reason for choosing this initial condition, and not let the fluid start at rest, was that we had problems with BiCGStab breakdown when the flow velocity magnitude varied too much over the domain.

We did two sets of simulations on the artery meshes, one set with a Reynolds number of about 200, and one with a Reynolds number of about 400. We will refer to the first set as low Reynolds number flow simulation, and the second as moderate Reynolds number flow simulation.

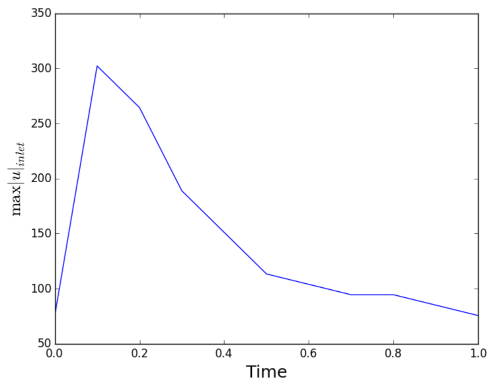
In the low Reynolds number simulations the maximum magnitude velocity at the inlet varied between $v(\boldsymbol{x}, t = 0)_{\max} = 77\,\mathrm{mm\,s^{-1}}$ and $v(\boldsymbol{x}, t = 0.1)_{max} = 309\,\mathrm{mm\,s^{-1}}$, while the flux varied between $Q_{\min} = 390\,\mathrm{mm^3\,s^{-1}}$ and $Q_{\max} = 1561\,\mathrm{mm^3\,s^{-1}}$. If we define the Reynolds number (at the inlet) as $\mathrm{Re} = \frac{\rho Q D}{\mu A}$, we get $\mathrm{Re}_{\min} = 49$ and $\mathrm{Re}_{\max} = 194$. In the moderate Reynolds number flow simulations, the above quantities were exactly twice as large. We used different time step sizes for the two different sets of simulations: 0.0005 for the low Re simulations and 0.0001 for the moderate Re simulations.

The results we report from these simulations are the average number of BiCGStab iterations per time step with empirical standard deviation, as well as the average time spent per timestep. Three different mesh resolutions were used, and we will refer to them as rf_0, rf_1, and rf_2. Using the $P2 - P1$ elements described in the beginning of this chapter, the different mesh resolutions led to the following number of unknowns:

- $rf_0 : n = 84836$
- $rf_1 : n = 200763$
- $rf_2 : n = 414297$

The values for the viscosity and density are similar to those used in [32] (in that paper the same parameters are $\mu = 0.0035\frac{\mathrm{N}}{\mathrm{mm}^2}$ and $\rho = 0.00105\frac{\mathrm{g}}{\mathrm{mm}^3}$).

In the previous section we described how we stabilized the $\tilde{S}_1^{-1}$ and $\tilde{S}_2^{-1}$ operators in the two dimensional case. When performing the experiments detailed above, we discovered that adding a mass matrix to the Schur complement approximations was not only unnecessary in the three dimensional case, but that doing so led to the Y_1-preconditioned system requiring on average more than three times as many iterations, and the Y_2-preconditioned system not converging within the set limit. For

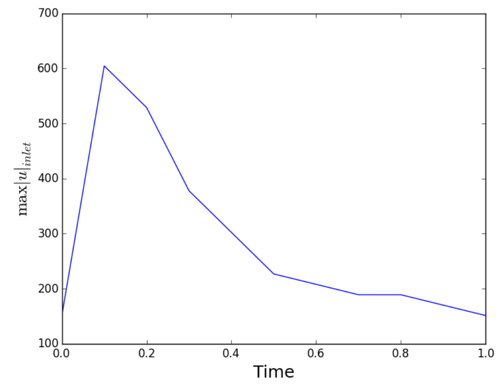


(a) Maximum velocity magnitude at the inlet vs. time, one heart cycle of the low Reynolds number simulations.

$\mu =$	$0.00345\,\mathrm{N\,mm^{-2}}$
$\rho =$	$0.00106\,\mathrm{g\,mm^{-3}}$
$\nu =$	$3.255\mathrm{m}^2/\,\mathrm{s}^2$
$\mathrm{radius}(inlet) =$	$1.8\,\mathrm{mm}$
$\mathrm{Re}_{\max}(inlet) =$	194
$\Delta t =$	0.0005

(b) Parameters for the low Reynolds number artery flow simulations.

this reason the arterial blood flow simulations were done with $\tilde{S}_1^{-1} = (\Delta t D \tilde{M}^{-1} G)^{-1}$ and $\tilde{S}_2^{-1} = (D\tilde{F}^{-1}G)^{-1}$, i.e. without the mass matrix stabilization described in the previous section. We did not find the reason for the observed difference in the two- and three dimensional experiments.



(a) Maximum velocity magnitude at the inlet vs. time, one heart cycle of the moderate Reynolds number flow simulations.

$\mu =$	$0.00345\,\mathrm{N\,mm^{-2}}$
$\rho =$	$0.00106\,\mathrm{g\,mm^{-3}}$
$\nu =$	$3.255\mathrm{m}^2/\,\mathrm{s}^2$
$\mathrm{radius}(inlet) =$	$1.8\,\mathrm{mm}$
$\mathrm{Re}_{\max}(inlet) =$	389
$\Delta t =$	0.0001

(b) Parameters for the moderate Reynolds number artery flow simulations.

6 Results

In this section we will refer to the block diagonal preconditioner defined on page 26 in section 4.1.1 by "B_ϵ", the projection method based preconditioner defined on page 27 in section 4.1.2 by "P_ϵ". The Yosida preconditioner defined on page 28 in section 4.1.3 we refer to as "Y_1," and the alternative Yosida preconditioner we will refer to by "Y_2."

The variants of the preconditioners used for the Oseen equations will be referred to by B_O, P_O, Y_1 and Y_2. They are defined on page 32 in section 4.2.

6.1 Numerically exact eigenvalues and condition numbers

6.1.1 Time-dependent stokes equations

The non-unitary eigenvalues $\lambda(P^{-1}A_S)$ of the preconditioned operators in the time-dependent Stokes case are presented below in figures 10-13. The operators were built on a uniform N-by-N grid of squares, each divided into two triangles (see figure 4 for an example), where $N = 32$.

We see in figure 11 that the non-unitary eigenvalues of $P_\epsilon A_S$ are within the interval $[0, 1)$ when the boundary conditions are those of lid-driven cavity flow. While it is not obvious from the figures, the multiplicity of the 0-eigenvalue is 1. For the velocity driven pipe flow case, some eigenvalues are greater than one. The analysis in [7] referred to in 4.1.4 does not take into account stabilization, and in the implementation used to produce these graphs the boundary conditions on the pressure function as stabilization. This may be the reason for the eigenvalues above one in the velocity driven pipe flow case. From figures 12 and 13, we see that the eigenvalues of $Y_1 A$ are all in the interval $[0, 1]$, while the largest eigenvalues of $Y_2 A$ are approximately 30 for both sets of boundary conditions. The spread of the eigenvalues of $Y_2 A$ is greatest when the kinematic viscosity ν is 0.1, while for the other preconditioners the eigenvalues either increase or decrease approximately monotonically with ν.

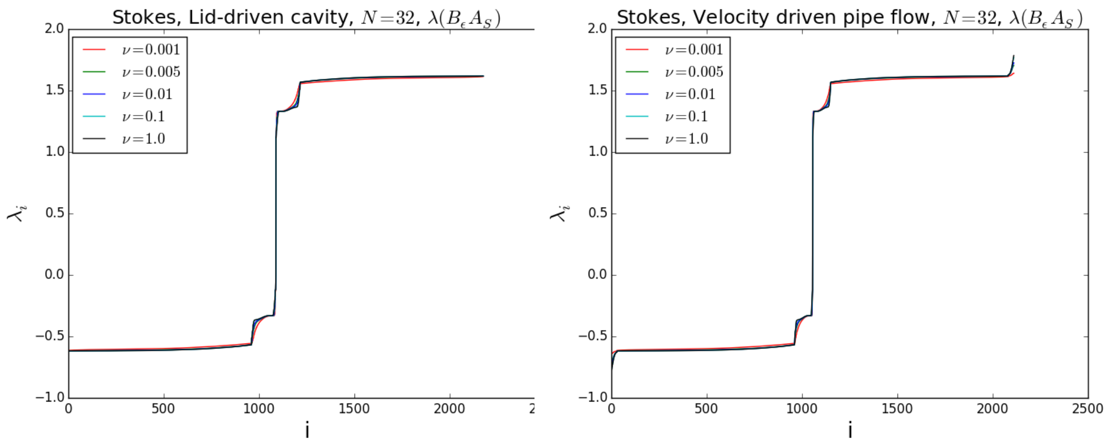


Figure 10: Non-unitary eigenvalues of $B_\epsilon A_S$ for each boundary condition sets.

We see in figure 10 that half of the non-unitary eigenvalues of $B_\epsilon A$ when $N = 32$ are distributed like those of $P_\epsilon A$, and the other half is antisymmetric to the first around the line $y = 0.5$. note that $B_\epsilon A$ has twice as many non-unitary eigenvalues as the other preconditioned operators, as predicted in section 4.1.4. The eigenvalues for each ν are so similar, it is difficult to distinguish the graphs.

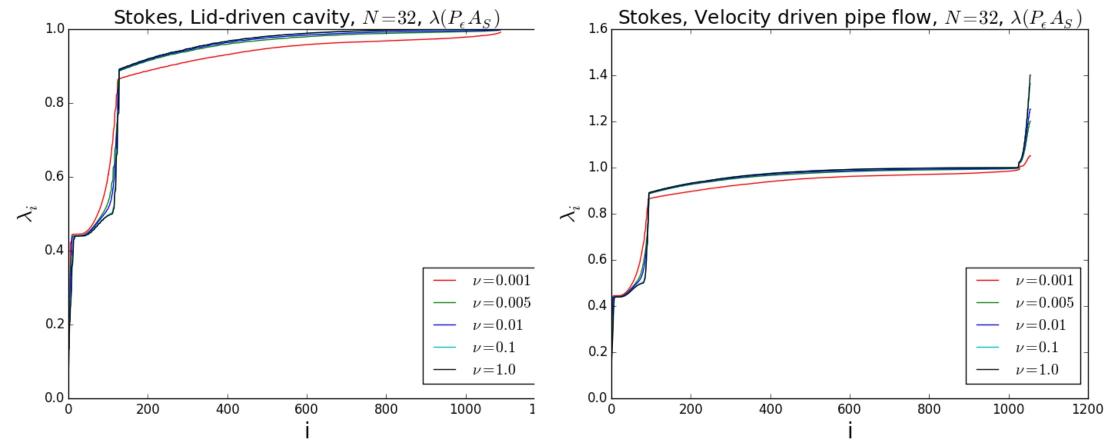


Figure 11: Non-unitary eigenvalues of $P_\epsilon A_S$ for each boundary condition set.

The non-unitary eigenvalues of $P_\epsilon A$ plotted in figure 11 have a distribution that is approximately constant in ν, with the exception of the largest eigenvalues when there are Dirichlet boundary conditions for p.

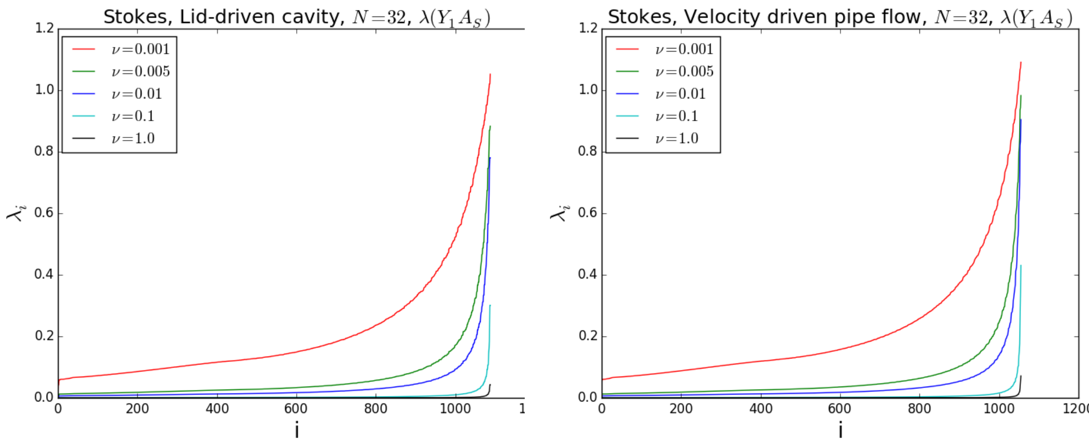


Figure 12: Non-unitary eigenvalues of $Y_1 A_S$, for each boundary condition set.

In figure 12 we see that the non-unitary eigenvalues of $Y_1 A$ are mostly close to zero when ν is large, with the distribution being more even for the smallest ν.

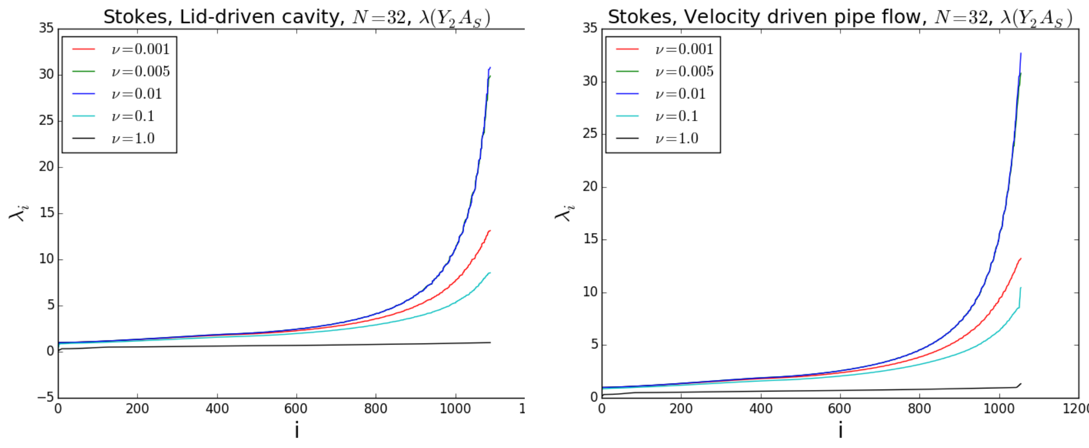


Figure 13: Non-unitary eigenvalues of $Y_2 A_S$ for each boundary condition set.

The eigenvalues of $Y_2 A$ plotted in figure 13 are largest when $\nu = 0.01$ and $\nu = 0.005$. The graphs for those two ν-values are nearly indistinguishable. The graph for $\nu = 1$ seems qualitatively different from the rest, as it has a nearly constant spectrum.

Tables 1-4 show the 2-norm condition numbers of the preconditioned operators $B_\epsilon A$, $P_\epsilon A$, $Y_1 A$, and $Y_2 A$, for both sets of boundary conditions. None of the preconditioners manage to keep the condition numbers for all values of ν in a reasonable range as N grows. When $\nu = 1$, $\kappa_1(P_\epsilon A)$ appears bounded, and the same seems to be the case for $\kappa_1(Y_2 A)$.

Lid-driven cavity, $\kappa_1(B_\epsilon A_S)$						Velocity driven pipe flow, $\kappa_1(B_\epsilon A_S)$					
$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001	$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	492.9	13.94	39.31	57.20	102.1	2^{-1}	451.8	15.73	41.19	59.54	99.36
2^{-2}	1822	26.22	59.17	96.47	278.9	2^{-2}	1791	30.79	67.66	104.0	281.2
2^{-3}	7140	79.64	82.77	154.2	501.1	2^{-3}	7130	79.55	100.1	175.1	514.7
2^{-4}	28478	293.3	96.49	185.2	794.6	2^{-4}	28478	293.3	114.7	219.1	852.7
2^{-5}	1e05	1147	108.9	201.8	935.8	2^{-5}	1e05	1147	122.1	238.4	1074

Table 1: 2-norm condition number of $B_\epsilon A_S$, for varying grid size and kinematic viscosity.

The 2-norm condition numbers of $B_\epsilon A$ listed in table 1 all increase monotonically as a function of h^{-1}, but there is no clear pattern in the dependence on ν. Neither is there a clear pattern in how large an increase in κ_2 we see between different h-values.

Lid-driven cavity, $\kappa_1(P_\epsilon A_S)$						Velocity driven pipe flow, $\kappa_1(P_\epsilon A_S)$					
$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001	$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	7.95	11.80	42.40	61.73	109.5	2^{-1}	8.73	13.49	44.14	63.95	106.4
2^{-2}	8.71	16.16	61.03	99.38	284.6	2^{-2}	9.57	18.98	69.78	107.2	287.0
2^{-3}	8.99	18.79	84.04	156.0	505.3	2^{-3}	10.53	23.00	101.3	176.8	518.7
2^{-4}	9.14	20.44	97.37	186.3	796.8	2^{-4}	11.25	24.91	115.5	220.1	854.6
2^{-5}	9.21	21.18	104.1	202.6	937.0	2^{-5}	11.69	25.77	122.3	239.2	1076

Table 2: 2-norm condition number of $P_\epsilon A_S$, for varying grid size and kinematic viscosity.

In both test cases the condition number of $P_\epsilon A$, shown in table 2, increase as a monotonic function of both ν^{-1} and h^{-1}. In the columns where $\nu = 1$ we see $\kappa_1(P_\epsilon A)$ is close to constant, while in the column where $\nu = 0.001$ the condition number grows by a factor of 10 between the coarsest and finest grids.

The condition numbers of $Y_1 A$, listed in table 3, seems to behave in a fashion similar to $\kappa_1(B_\epsilon A)$, in that it grows with N for each value of ν in both test cases. We also see the same "parabolic" behaviour if N is fixed and we view κ_1 as a function of ν.

The behaviour of $\kappa_1(Y_2 A)$, shown in table 4, as a function of h^{-1} and ν is similar to that of $\kappa_1(P_\epsilon A)$. The condition number is nearly constant in h^{-1} when $\nu = 1$, while for $\nu = 0.001$ it grows by a factor of 10^3 from $h = 0.5$ to $h = 2^{-5}$.

Tables 5-8 show the ratios of the largest by the smallest eigenvalue, by moduli, of $B_\epsilon A$, $P_\epsilon A$, $Y_1 A$, and $Y_2 A$. $\kappa_2(P^{-1}A)$ appears uniformly bounded for both $P^{-1} = B_\epsilon$

Lid-driven cavity, $\kappa_1(Y_1 A_S)$						Velocity driven pipe flow, $\kappa_1(Y_1 A_S)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	118.8	17.12	4.23	2.28	10.07	2^{-1}	110.7	17.99	3.90	2.66	8.39
2^{-2}	433.6	73.59	33.03	23.95	7.31	2^{-2}	448.5	85.33	33.41	23.52	7.11
2^{-3}	1744	333.8	136.1	143.9	72.65	2^{-3}	1819	381.9	143.2	146.6	72.04
2^{-4}	6981	1399	437.7	451.4	576.5	2^{-4}	7332	1603	490.5	485.6	578.1
2^{-5}	27943	5694	1684	1280	1900	2^{-5}	29436	6521	1974	1472	1999

Table 3: 2-norm condition number of $Y_1 A_S$, for varying grid size and kinematic viscosity.

Lid-driven cavity, $\kappa_1(Y_2 A_S)$						Velocity driven pipe flow, $\kappa_1(Y_2 A_S)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	8.51	3.06	1.79	2.89	7.80	2^{-1}	9.84	3.34	2.37	4.00	9.40
2^{-2}	8.84	8.13	20.93	9.11	16.59	2^{-2}	9.51	11.12	28.31	10.98	16.99
2^{-3}	8.92	21.39	285.1	255.3	29.89	2^{-3}	9.92	32.20	345.0	301.3	33.21
2^{-4}	8.95	34.48	1133	1579	1127	2^{-4}	11.09	53.80	1439	1870	1207
2^{-5}	8.97	40.34	2752	5292	7921	2^{-5}	11.96	66.62	3654	6592	9040

Table 4: 2-norm condition number of $Y_2 A_S$, for varying grid size and kinematic viscosity.

and $P^{-1} = P_\epsilon$. While the eigenvalue ratios of $Y_2 A$ are not large, they grow with N for all values of ν except $\nu = 1$. In contrast to the other operators, $\kappa_2(Y_1 A)$ grows with N for all values of ν.

Lid-driven cavity, $\kappa_2(B_\epsilon A_S)$						Velocity driven pipe flow, $\kappa_2(B_\epsilon A_S)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	12.97	11.92	8.67	7.77	6.65	2^{-1}	13.22	11.38	8.34	7.49	6.31
2^{-2}	13.25	12.49	10.31	9.35	7.31	2^{-2}	13.81	12.71	10.35	9.37	7.33
2^{-3}	13.44	13.07	11.88	11.22	9.16	2^{-3}	14.31	13.66	11.95	11.25	9.16
2^{-4}	13.52	13.35	12.75	12.39	11.00	2^{-4}	14.72	14.34	13.23	12.66	11.01
2^{-5}	13.56	13.48	13.19	13.01	12.26	2^{-5}	14.97	14.75	14.07	13.71	12.46

Table 5: Eigenvalue ratios $\kappa_2(B_\epsilon A_S)$, for varying grid sizes and kinematic viscosity.

We see in table 5 that the eigenvalues of B_ϵ are largely insensitve to both the grid resolution and the viscosity parameter, as $\kappa_2(B_\epsilon A)$ appears to be bounded above both in h and ν.

In table 6 we see that $\kappa_2(P_\epsilon A)$ behaves largely like $\kappa_2(B_\epsilon A)$, with the entries of the two tables being seperated by a factor of approximately 2 in favour of P_ϵ. This is

Lid-driven cavity, $\kappa_2(P_\epsilon A_S)$						Velocity driven pipe flow, $\kappa_2(P_\epsilon A_S)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	7.35	6.73	4.78	4.24	3.56	2^{-1}	7.37	6.34	4.60	4.08	3.34
2^{-2}	7.32	6.89	5.67	5.12	3.89	2^{-2}	8.11	7.14	5.67	5.12	3.89
2^{-3}	7.42	7.20	6.51	6.13	4.92	2^{-3}	9.10	8.37	6.51	6.13	4.92
2^{-4}	7.46	7.36	7.00	6.79	5.98	2^{-4}	9.95	9.43	7.94	7.26	5.98
2^{-5}	7.49	7.44	7.26	7.16	6.71	2^{-5}	10.50	10.14	9.10	8.60	7.06

Table 6: Eigenvalue ratios $\kappa_2(P_\epsilon A_S)$, for varying grid sizes and kinematic viscosity.

consistent with the theory in section 4.1.4 and consistent with the plots of eigenvalues above.

Lid-driven cavity, $\kappa_2(Y_1 A_S)$						Velocity driven pipe flow, $\kappa_2(Y_1 A_S)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	105.4	10.98	1.55	1.35	1.90	2^{-1}	96.91	10.23	1.68	1.57	2.32
2^{-2}	368.4	37.33	4.22	2.38	1.39	2^{-2}	367.0	37.20	4.21	2.38	1.73
2^{-3}	1447	145.2	15.01	7.78	2.07	2^{-3}	1447	145.2	15.02	7.78	2.08
2^{-4}	5769	577.4	58.24	29.39	6.63	2^{-4}	5769	577.4	58.24	29.39	6.63
2^{-5}	23055	2306	230.9	115.9	24.83	2^{-5}	23055	2306	231.1	115.9	24.83

Table 7: Eigenvalue ratios $\kappa_2(Y_1 A_S)$, for varying grid sizes and kinematic viscosity.

The eigenvalue ratios $\kappa_2(Y_1 A)$ shown in table 7 grow quickly in h^{-1}, but decreases nearly linearly in ν^{-1}. This reflects the reliance of the Schur complement approximation $\tilde{S}_1^{-1} = (\Delta t D \tilde{M}^{-1} G)$ on a small time step and/or small viscosity.

Lid-driven cavity, $\kappa_2(Y_2 A_S)$						Velocity driven pipe flow, $\kappa_2(Y_2 A_S)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	8.64	2.71	1.19	1.34	1.88	2^{-1}	8.99	2.81	1.46	1.67	2.30
2^{-2}	8.47	5.17	2.42	1.68	1.43	2^{-2}	8.75	5.78	2.53	1.70	1.77
2^{-3}	8.50	10.25	7.40	4.95	1.67	2^{-3}	9.34	11.18	7.53	4.98	1.66
2^{-4}	8.51	13.86	19.07	14.60	4.82	2^{-4}	10.56	15.88	19.66	14.86	4.86
2^{-5}	8.52	16.08	39.76	37.65	16.13	2^{-5}	11.44	19.67	42.17	38.77	16.23

Table 8: Eigenvalue ratios $\kappa_2(Y_2 A_S)$, for varying grid sizes and kinematic viscosity.

In table 8 we see the improvement Y_2 is to Y_1, in that the eigenvalues of Y_2 are less sensitive to the viscosity parameter. When $\nu = 1$, $\kappa_2(Y_2 A)$ is constant, or nearly constant, in h. When $\nu = 0.001$ the eigenvalue ratios look similar to $\kappa_2(Y_1 A)$ in table 7.

6.1.2 Modified Oseen problem

Figures 14-17 show the distribution of the non-unitary eigenvalues of the preconditioned operators $P^{-1}A$ in the complex plane. The operators were built on was a uniform N-by-N grid of squares, each divided into two triangles (see figure 4 for an example), where $N = 32$.

When the viscosity parameter ν is 1, the eigenvalues of each are all mostly real, while for $\nu < 1$ they spread out in the complex plane. Half the plots show the eigenvalues roughly contained in the unit disc. The eigenvalues $\lambda(Y_2A)$ are larger in magnitude than the others, while the eigenvalues of B_OA are clumped in two mirrored clusters.

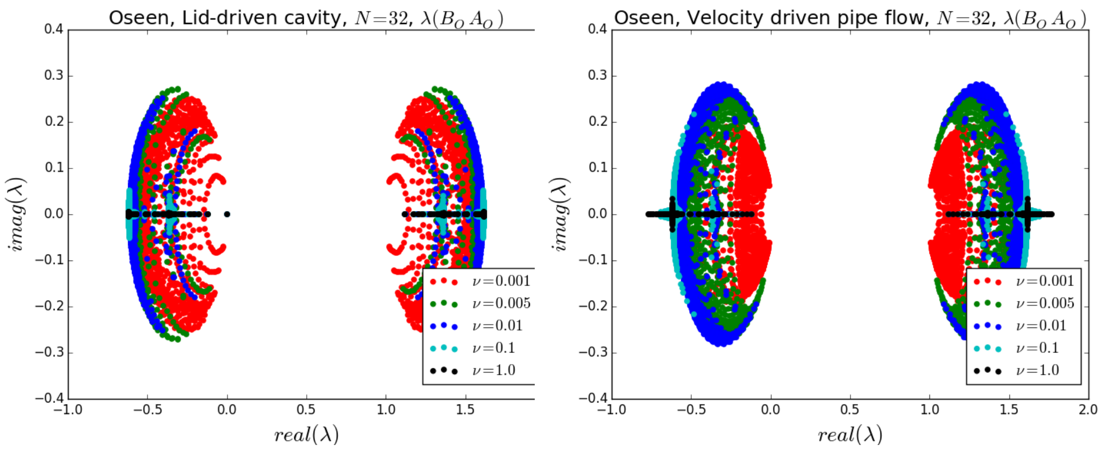


Figure 14: Non-unitary eigenvalues of B_OA_O.

The eigenvalues $\lambda(B_OA)$ plotted in figure 14 are contained in two disjunct, mirrored clusters. The distribution of the eigenvalues in the complex plane is the same as $\lambda(P_OA)$, but mirrored along the line real$(\lambda) = 0.5$

The non-unitary eigenvalues of the operator P_OA plotted in figure 15 are seen to all be contained in the unit disc in the lid driven cavity test case. In the velocity driven pipe flow test case the same is true, except for a small subset of the eigenvalues, the size of which decreases with ν. The eigenvalues seem to cluster progressively closer to the origin with a decreasing ν. We also see how close the preconditioned Oseen operator is to the time dependent Stokes operator in terms of eigenvalues when ν is large.

Figure 14 and 15 clearly demonstrate, as was argued in section 4.1.4, that the non-unitary eigenvalues of B_OA are mirrored about the line real$(\lambda) = 0.5$, with the same distribution as those of P_OA.

The eigenvalues of Y_1A_O plotted in figure 16 were mostly clustered close to the

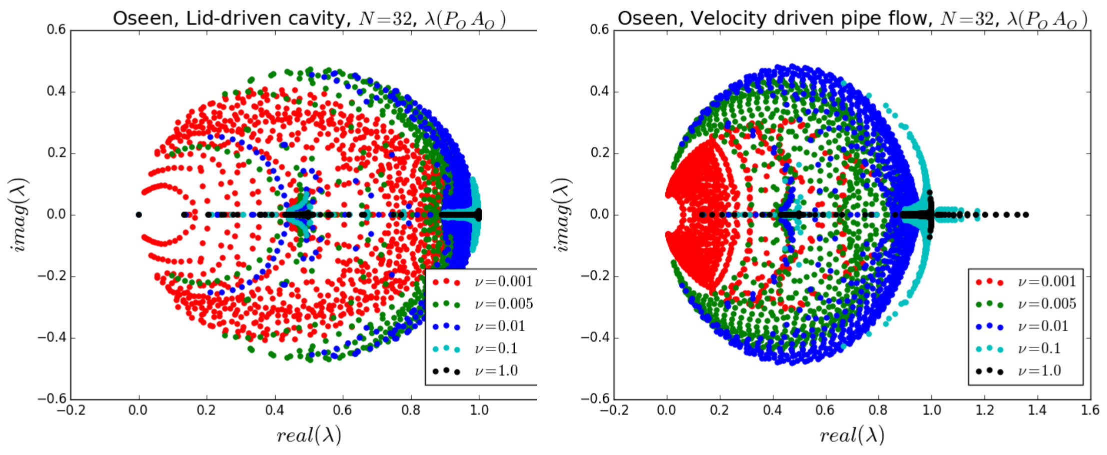


Figure 15: Non-unitary eigenvalues of $P_O A_O$.

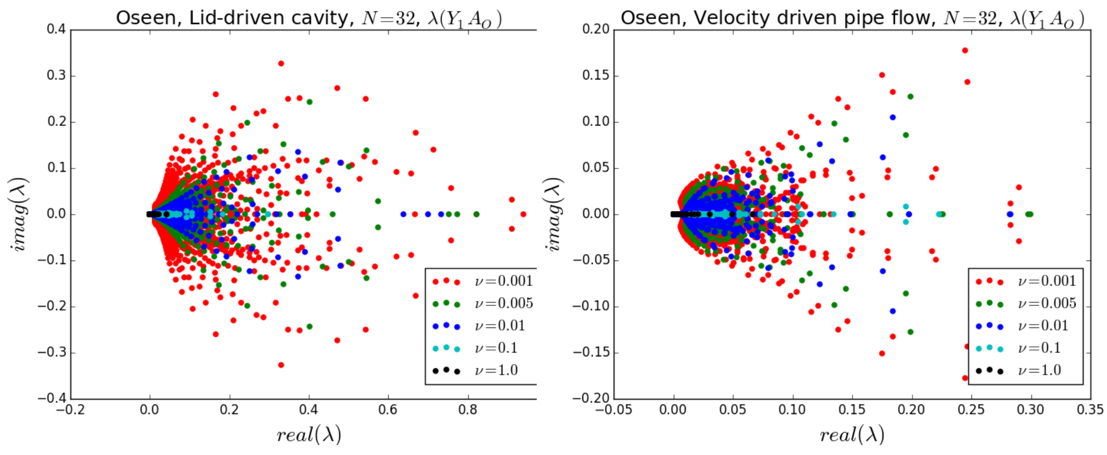


Figure 16: Non-unitary eigenvalues of $Y_1 A_O$.

origin, like they were for $Y_1 A_S$. When $\nu = 0.001$ the eigenvalues spread out in the unit disc in the complex plane.

The non-unitary eigenvalues of $Y_2 A_O$ are in figure 17 seen to be similar in magnitude to the eigenvalues of $Y_2 A_S$. They are more spread out when $\nu = 0.01$, and more tightly clustered when $\nu = 1$ and when $\nu = 0.001$, which was what we saw in the graphs in the previous section for the time dependent Stokes operator.

Tables 9-12 show the 2-norm condition numbers of the preconditioned operators. These were largely similar to the tables in the previous section, except that they are slightly larger when $\nu = 0.001$.

The 2-norm condition numbers of $B_O A$ shown in table 9 grew with h^{-1} for all values

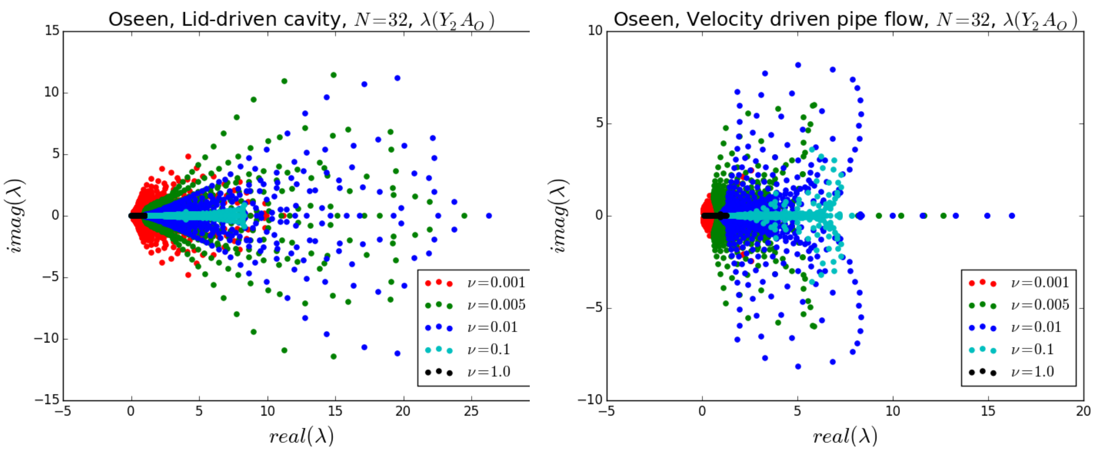


Figure 17: Non-unitary eigenvalues of Y_2A_O.

Lid-driven cavity, $\kappa_1(B_OA_O)$						Velocity driven pipe flow, $\kappa_1(B_OA_O)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	492.7	14.01	40.10	60.42	114.9	2^{-1}	450.2	14.30	50.07	76.52	127.7
2^{-2}	1821	26.27	58.96	105.0	505.7	2^{-2}	1789	25.82	90.09	182.0	563.5
2^{-3}	7139	79.60	88.47	171.8	1333	2^{-3}	7128	79.40	125.0	293.5	1854
2^{-4}	28477	293.2	105.5	220.2	1755	2^{-4}	28476	293.1	144.6	333.7	3369
2^{-5}	1e05	1147	113.8	242.5	1294	2^{-5}	1e05	1147	156.5	359.9	3173

Table 9: 2-norm condition number of B_OA_O, for varying grid size and kinematic viscosity.

of ν. $\kappa_1(B_OA)$ was larger than it was for time dependent Stokes for the smallest values of ν.

Lid-driven cavity, $\kappa_1(P_OA_O)$						Velocity driven pipe flow, $\kappa_1(P_OA_O)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	7.96	11.87	42.91	64.41	121.2	2^{-1}	8.89	12.08	51.35	78.41	130.6
2^{-2}	8.71	16.22	60.71	107.2	511.2	2^{-2}	9.49	15.58	91.14	184.0	569.4
2^{-3}	8.99	18.79	89.53	173.1	1337	2^{-3}	10.25	18.41	124.6	293.3	1860
2^{-4}	9.14	20.40	106.2	221.0	1755	2^{-4}	10.85	20.01	143.7	332.2	3368
2^{-5}	9.21	21.14	114.4	243.0	1295	2^{-5}	11.30	20.83	154.9	357.7	3165

Table 10: 2-norm condition number of P_OA_O, for varying grid size and kinematic viscosity.

In table 10 we see that $\kappa_1(P_OA)$ increased as a function of h^{-1} and ν^{-1}, except when $\nu = 1$ when the condition number was constant in h^{-1}. We see somewhat larger condition numbers when viscosity is small, compared to the results in the previous section where convection was not present.

Lid-driven cavity, $\kappa_1(Y_1 A_O)$						Velocity driven pipe flow, $\kappa_1(Y_1 A_O)$					
$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001	$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	118.7	17.12	5.91	4.94	8.05	2^{-1}	109.1	16.97	29.03	32.17	35.78
2^{-2}	433.4	73.70	43.46	54.60	111.4	2^{-2}	444.0	71.49	107.3	148.0	225.9
2^{-3}	1744	333.2	160.1	232.0	703.2	2^{-3}	1803	328.8	318.1	503.1	1339
2^{-4}	6981	1396	451.5	724.7	2904	2^{-4}	7263	1397	712.4	1308	4779
2^{-5}	27945	5683	1685	1529	6503	2^{-5}	29144	5723	1589	2705	12706

Table 11: 2-norm condition number of $Y_1 A_O$, for varying grid size and kinematic viscosity.

The 2-norm condition numbers of $Y_1 A$, shown in table 11, were larger than in the previous section (table 3) when ν is small, and nearly equal when ν is large. The 2-norm condition numbers of $Y_2 A$ listed in table 12 grew in h^{-1} and ν^{-1}, except when $\nu = 1$.

Lid-driven cavity, $\kappa_1(Y_2 A_O)$						Velocity driven pipe flow, $\kappa_1(Y_2 A_O)$					
$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001	$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	8.51	3.05	3.02	6.12	17.11	2^{-1}	10.20	4.08	21.12	34.12	58.26
2^{-2}	8.84	8.06	18.44	21.66	176.4	2^{-2}	9.71	9.29	34.93	79.91	336.9
2^{-3}	8.92	21.16	236.5	204.9	456.9	2^{-3}	9.76	22.29	116.1	218.5	917.7
2^{-4}	8.96	33.97	987.4	1322	2522	2^{-4}	10.76	33.06	494.8	644.0	3345
2^{-5}	8.97	39.73	2488	4676	7089	2^{-5}	11.60	38.54	1552	2781	10271

Table 12: 2-norm condition number of $Y_2 A_O$, for different grid sizes and kinematic viscosities.

The eigenvalue ratios $\kappa_2(P^{-1}A)$ for the Oseen problem are found in tables 13-16. These are similar to the corresponding tables in section 6.1.1, the only difference being that $\kappa_2(P^{-1}A_O)$ was larger than $\kappa_2(P^{-1}A_S)$ when $\nu = 0.001$ for each h for all the preconditioners.

The eigenvalue ratios $\kappa_2(B_O A)$ listed in table 13 seem insensitive to h, and up to a point also to ν. When $\nu = 0.001$ the values are larger, but they still do not increase in h. The behaviour of $\kappa_2(P_O A)$ was the same as that of $\kappa_2(B_O A)$. The values in the two sets of tables are seperated roughly by a constant factor of 2, like the corresponding values in the previous section.

The eigenvalue ratios $\kappa_2(Y_1 A)$ are in table 15 seen to grow in h^{-1} for each ν-value, without reaching any apparent bound. Like the others in this section, the two tables

Lid-driven cavity, $\kappa_2(B_OA_O)$						Velocity driven pipe flow, $\kappa_2(B_OA_O)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	12.98	11.96	8.79	8.03	7.52	2^{-1}	13.32	10.89	9.44	13.16	19.11
2^{-2}	13.24	12.42	10.29	9.61	12.24	2^{-2}	13.69	12.23	9.07	10.12	24.40
2^{-3}	13.44	13.04	11.79	11.22	18.26	2^{-3}	14.16	13.04	11.21	11.08	28.65
2^{-4}	13.52	13.34	12.72	12.36	22.88	2^{-4}	14.56	13.52	12.61	12.09	28.04
2^{-5}	13.56	13.47	13.18	13.00	21.86	2^{-5}	14.82	14.11	13.18	12.96	26.85

Table 13: Eigenvalue ratio $\kappa_2(B_OA_O)$, for varying grid size and kinematic viscosity.

Lid-driven cavity, $\kappa_2(P_OA_O)$						Velocity driven pipe flow, $\kappa_2(P_OA_O)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	7.36	6.76	5.00	4.57	4.20	2^{-1}	7.48	6.48	6.50	9.61	14.83
2^{-2}	7.32	6.89	5.79	5.44	7.09	2^{-2}	7.78	6.92	5.84	6.90	18.07
2^{-3}	7.42	7.20	6.53	6.24	10.96	2^{-3}	8.74	7.21	6.58	7.18	20.40
2^{-4}	7.46	7.36	7.00	6.81	13.91	2^{-4}	9.59	7.69	7.02	6.83	18.36
2^{-5}	7.49	7.44	7.26	7.16	13.28	2^{-5}	10.16	8.73	7.27	7.16	16.16

Table 14: Eigenvalue ratio $\kappa_2(P_OA_O)$, for varying grid size and kinematic viscosity.

Lid-driven cavity, $\kappa_2(Y_1A_O)$						Velocity driven pipe flow, $\kappa_2(Y_1A_O)$					
$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001	$h\backslash\nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	105.4	10.99	1.57	1.27	1.70	2^{-1}	96.92	9.99	3.63	3.58	3.53
2^{-2}	368.4	37.33	4.32	2.61	3.12	2^{-2}	367.0	37.12	6.71	6.38	7.77
2^{-3}	1449	145.1	15.23	8.12	8.62	2^{-3}	1447	145.1	15.33	15.63	18.97
2^{-4}	5882	578.0	58.41	29.90	23.72	2^{-4}	5882	578.0	58.17	33.59	43.40
2^{-5}	25000	2326	230.9	116.1	57.33	2^{-5}	25000	2326	230.9	115.7	117.0

Table 15: Eigenvalue ratio $\kappa_2(Y_1A_O)$, for varying grid size and kinematic viscosity.

are very similar to the corresponding tables in the previous section, except the values for $\nu = 0.001$ are larger here. We see that Y_1A was the only one of the operators that had a very large relative distance between the smallest and largest eigenvalues.

$\kappa_2(Y_2A)$ is in table 16 seen to be nearly constant in h^{-1} when $\nu = 1$, but increasing in h^{-1} for the other ν-values.

Lid-driven cavity, $\kappa_2(Y_2 A_O)$						Velocity driven pipe flow, $\kappa_2(Y_2 A_O)$					
$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001	$h \backslash \nu$	1.0	0.1	0.01	0.005	0.001
2^{-1}	8.64	2.71	1.19	1.29	1.74	2^{-1}	9.16	2.04	2.83	3.56	4.68
2^{-2}	8.47	5.08	2.43	1.95	3.67	2^{-2}	8.84	4.15	2.56	3.98	10.55
2^{-3}	8.50	9.97	6.25	4.55	7.70	2^{-3}	8.94	7.46	3.49	4.08	13.51
2^{-4}	8.51	13.32	16.14	12.69	18.58	2^{-4}	10.13	11.41	7.91	8.91	14.91
2^{-5}	8.52	15.72	33.99	31.01	39.16	2^{-5}	11.03	14.84	20.98	16.55	39.53

Table 16: Eigenvalue ratio $\kappa_2(Y_2 A_O)$, for varying grid size and kinematic viscosity.

6.2 Simulations on the unit square

N	16	32	64	128	256
$\kappa_2(\text{AMG}(K)K)$	1.058	1.068	1.079	1.09	1.092
$\kappa_2(\text{AMG}(M)M)$	1.055	1.051	1.05	1.051	1.051
$\kappa_2(\text{AMG}(A)A)$	1.029	1.046	1.055	1.059	1.079

Table 17: Eigenvalue ratio estimate for some operators preconditioned by AMG. In the bottom row, $A = A_{\text{Stokes}}$, $\Delta t = 0.01$ and $\nu = 1$.

In this section we present tables of the average number of iterations before convergence of the BiCGStab method, for the same problem, preconditioner and parameter combinations we listed results for earlier in this chapter. We set the maximum number of iterations to 400, so table entries "N/A" mean that combination did not converge within that limit. Numbers in parenthesis are the empirical standard deviations. We do not include the standard deviation when it is less than five percent of the average iteration number. The AMG approximation to the inverses seems to not be sensitive to the mesh resolution.

Table 17 shows the estimated eigenvalue ratios of some of the operators used in the preconditioners. The algebraic multigrid preconditioner mentioned in section 5.2 obviously does a good job of approximating the action of the inverse of the mass matrix, stiffness matrix and the combination A of those. We estimated the eigenvalues by solving e.g. $\text{AMG}(A)A\boldsymbol{u} = \text{AMG}(A)\boldsymbol{f}$ with the conjugate gradient method, where the initial guess was a vector with uniformly distributed random numbers in the interval $[-0.5, 0.5]$. For this we used the cbc.block module [9].

6.2.1 Time dependent Stokes flow simulations

All the tables in this section suggest that the performance of the preconditioners is closely tied to the spectrum of $P^{-1}A$. $\kappa_2(P^{-1}A)$ increasing with N for a given ν in section 6.1.1 was closely correlated with a corresponding increase, as a function of N, in the average number of iterations for the same P^{-1} and ν. No such correlation can be seen between $\kappa_1(P^{-1}A)$ and the number of iterations required to achieve the desired error bound.

The nonexistance of any correlation between κ_1 and the iteration number is highlighted by comparing the tables of results for P_ϵ and Y_1 in this section and section 6.1.1. $\kappa_1(P_\epsilon A)$ is approximately inversely proportional to ν, while the actual performance of P_ϵ improves as ν is decreased. On the other hand, $\kappa_1(Y_1 A)$ decreases with ν, and Y_1 performs better with a smaller ν.

Lid driven cavity flow, B_ϵ						Velocity driven pipe flow, B_ϵ					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	94(13)	61(12)	33(3)	27(2)	23(2)	16	51(7)	39(7)	33(3)	28(3)	28(3)
32	74(9)	72(14)	32(3)	29(4)	24(2)	32	55(8)	41(6)	31(2)	31(3)	28(3)
64	73(7)	73(13)	36(4)	32(4)	26(2)	64	57(6)	44(5)	32(3)	31(3)	28(3)
128	74(7)	61(8)	41(5)	32(3)	29(2)	128	58(5)	49(6)	36(3)	32(3)	27(2)
256	76(8)	57(7)	43(6)	35(3)	32(3)	256	60(5)	53(7)	39(4)	36(3)	30(3)

Table 18: Average number of BiCGStab iterations per timestep when using B_ϵ, with convergence criterion $\|r\|_2 < 10^{-14}$.

In table 18 we see that the performance of B_ϵ was somewhat variable, in that the number of iterations per timestep had an empirical standard deviation of about ten percent of the average over the simulation. The average number of iterations increased slightly with the mesh resolution, but improved uniformly as ν was decreased.

Lid driven cavity flow, P_ϵ						Velocity driven pipe flow, P_ϵ					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	19(1)	14	13	10	9	16	16	12	16	11	10
32	21	14(1)	11	10	9	32	18	14	11	11	11
64	22	16(1)	12	11	10	64	20	15	12	11	10
128	23	16(1)	13	12	10	128	21	17	13	12	10
256	23	17(1)	14	13	11	256	22	18	15	14	12

Table 19: Average number of BiCGStab iterations per timestep when using P_ϵ, with convergence criterion $\|r\|_2 < 10^{-14}$.

The preconditioner P_ϵ resulted in a very stable and low number of iterations, tabulated in table 19, independently of the mesh resolution and the value of ν. Here, like in table 18 we see a uniform improvement in performance as ν was decreased.

Lid driven cavity flow, Y_1						Velocity driven pipe flow, Y_1					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	43	13(2)	7	7	8	16	40(2)	13	8	7	9
32	88(5)	28	8(1)	5	7	32	73	25	8(1)	6	8
64	182(12)	59(5)	17	11	4	64	139(8)	48(3)	17(1)	11	5
128	N/A	117(8)	35(2)	23	10(1)	128	265(17)	89(5)	31(2)	23(2)	9
256	N/A	231(16)	71(5)	50(4)	20	256	N/A	167(12)	55(3)	40(2)	19(1)

Table 20: Average number of BiCGStab iterations per timestep when using Y_1, with convergence criterion $\|r\|_2 < 10^{-14}$.

Table 20 is evidence that the lumped mass matrix is a poor approximation of F. When the viscosity ν was small enough the number of iterations was not much higher than for P_ϵ, and better than for B_ϵ.When $\frac{1}{\Delta t} >> \nu$, $F = \frac{1}{\Delta t}M + \nu K$ is dominated by the mass matrix. Given that the lumped mass matrix is a reasonable approximation of a mass matrix, but a poor approximation of a stiffness matrix, the poor performance when $\nu = 1$ was not surprising.

Lid driven cavity flow, Y_2						Velocity driven pipe flow, Y_2					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	25	21	17(1)	17	17	16	23	21	18(1)	17	19
32	25	28(2)	19	16(1)	16	32	23	28	20(1)	17	18
64	24	34(2)	30	24	15(1)	64	24	37(2)	32	26	17(1)
128	23	35(2)	49(3)	39(3)	22	128	24	43	52	43	24
256	21	36	72	63(6)	38(3)	256	25	49	77(4)	71	43

Table 21: Average number of BiCGStab iterations per timestep when using Y_2, with convergence criterion $\|r\|_2 < 10^{-14}$.

In table 21 we see that Y_2 performed on par with P_ϵ when the viscosity parameter was equal to 1, but worse otherwise. The average number of iterations before convergence increased with N for all values of ν except 1, although at worst that increase was by a factor of 3 when N increased by a factor of 16.

6.2.2 Navier-Stokes flow simulations

In tables 22-25 we see that, for the parameter values and gridsizes tested, the preconditioners performed approximately as well when convection was turned on. Only when ν was 0.001, corresponding to a Reynolds number of about 1000, did performance deteriorate compared to what we saw in the previous section.

Lid driven cavity flow, B_O						Velocity driven pipe flow, B_O					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	91(11)	64(17)	36(4)	28(3)	28(3)	16	49(7)	39(7)	34(4)	33(4)	32(4)
32	70(6)	66(12)	34(4)	33(3)	32(3)	32	52(6)	40(5)	39(4)	39(4)	45(11)
64	71(7)	57(7)	39(4)	38(4)	54(10)	64	55(6)	43(5)	41(5)	45(7)	103(58)
128	73(6)	57(7)	43(4)	43(4)	75(18)	128	57(5)	46(5)	45(5)	49(6)	106(39)
256	78(8)	58(8)	48(5)	48(5)	85(13)	256	57(6)	49(5)	47(5)	50(6)	100(29)

Table 22: Average number of BiCGStab iterations per timestep when using B_O, with convergence criterion $\|r\|_2 < 10^{-14}$.

We see in table 22 that preconditioning the discrete Oseen equations with B_O led to some variance in the number of iterations, regardless of the grid resolution and viscosity. For the smallest viscosity this was more pronounced, especially when the boundary conditions were those of velocity driven pipe flow. A reason for the difference between the two boundary condition sets might be that in the lid-driven cavity setup the flow was less developed when results were recorded, and the gradient of the velocity in that case was steeper.

Lid driven cavity flow, P_O						Velocity driven pipe flow, P_O					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	20(2)	14	14	10	10	16	16	13	13(1)	11	11
32	21	15(1)	11	11	10	32	18	14	13(1)	13	13(1)
64	21	16(1)	13	13(1)	17(2)	64	19	15	15	14(1)	24(7)
128	22	16(1)	15	16(1)	22(2)	128	20	17	16(1)	17(1)	27(6)
256	23	17(1)	16	17(1)	25(2)	256	21	18	17(1)	18(1)	26(5)

Table 23: Average number of BiCGStab iterations per timestep when using P_O, with convergence criterion $\|r\|_2 < 10^{-14}$.

As we saw from the convectionless results in section 6.2.1, P_O is both the most robust of the preconditioners, and the best performing. Still we see, like we did for B_O in table 22, that the number of iterations required to get the residual in BiCGStab below 10^{-14} increased when the Reynolds number passed some threshold < 1000. The variance in the iteration count is clearly less with P_O than B_O

Table 24 tells the same story as table 20, namely that the Yosida preconditioner Y_1 leads to slow convergence of BiCGStab for the larger ν when $\Delta t = 0.01$. The number of iterations increase with N no matter the Reynolds number, but was manageable for even the largest grid sizes when $\mathrm{Re} \geq 100$.

The average number of iterations for the Y_2-preconditioned Oseen problem tabulated

Lid driven cavity flow, Y_1						Velocity driven pipe flow, Y_1					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	43	13(1)	8	7	9	16	39	13(1)	8	8	10
32	89(5)	28	9(1)	7	9	32	73	25	11(1)	10(1)	11(1)
64	182(14)	56(3)	18(1)	13	13(1)	64	139(8)	49	21	17(1)	24(16)
128	N/A	114(7)	37	28	23(3)	128	269(20)	90(5)	40(3)	33(3)	31(7)
256	N/A	238(19)	74(5)	58(4)	43(4)	256	N/A	168(11)	73(5)	64(6)	48(8)

Table 24: Average number of BiCGStab iterations per timestep when using Y_1, with convergence criterion $\|r\|_2 < 10^{-14}$.

Lid driven cavity flow, Y_2						Velocity driven pipe flow, Y_2					
N\\ν	1.0	0.1	0.01	0.005	0.001	N\\ν	1.0	0.1	0.01	0.005	0.001
16	25	21	18(1)	18	19	16	23	22	18(1)	18	19
32	25	30	21	20(1)	20(1)	32	23	30	25	22	24(3)
64	24	39(2)	35(2)	32(2)	26(2)	64	23	37	42	36(2)	35(16)
128	24	45(3)	62(5)	56(5)	46(5)	128	24	43	68(4)	67(5)	52(7)
256	23	50(3)	93(5)	96(9)	86(9)	256	25	48	104(8)	115(10)	100(12)

Table 25: Average number of BiCGStab iterations per timestep when using Y_2, with convergence criterion $\|r\|_2 < 10^{-14}$.

in table 25 was constant in N when the Reynolds number was small, but increased with N when the Reynolds number was large. The increase in iterations as a function of N for larger Reynolds numbers was of approximately the same magnitude as for the Y_1-preconditioned system, but Y_2 was more robust with respect to ν than Y_1, at least when $\Delta t = 0.01$ (as it was in all the simulations presented in this and the previous section). The variance was larger larger for the smaller viscosity values.

Tables 22-25 show a very close correlation between the eigenvalue spectrum, measured by $\kappa_2(P^{-1}A)$, of the preconditioned operators, and the average number of iterations required to reach a certain residual norm $\|r_n\|_2 < 10^{-14}$ in the BiCGStab algorithm. Like we saw in section 6.2.1, no such correlation is evident between the 2-norm condition number $\kappa_1(P^{-1}A)$ and the performance of the preconditioners.

6.3 Arterial blood flow simulations

In table 26 we compare the time per BiCGStab iteration for each of the preconditioners. Optimizing the implementation of the preconditioners to minimize computational time was not the focus of this part of the thesis. Using algebraic multigrid to approximate inverse operators, the time it took to apply the preconditioners was an approximately linear function of the number n of unknows in the discrete linear system. Note that we have not compared the time to build the preconditioners each time step. Table 26 should probably be taken with a grain of salt, as the simulations were done over the span of a few days on a machine that was under variable load by multiple users.

Tables 27-28 contain the results from the low Reynolds number flow simulations. The tables show that each preconditioner was able to keep the average iteration number near constant when the grid resolution is increased, also in the case of a three dimensional flow problem with less simple boundary geometry. The average computation time each time step grew approximately linearly with the number of unknowns. The iteration counts and computation time measurements presented are averages over two heart cycles (4000 time steps). In all these simulations the convergence criterion was $\|r\|_2 < 10^{-14}$, where r is the residual vector in the BiCGStab method.

Tables 29-30 contain the results of the moderate Reynolds number simulations. The maximum inlet flow velocity magnitude in these simulations was $605\,\mathrm{mm\,s^{-1}}$, giving a Reynolds number of 400 in the part of the artery preceding the aneurysm. In contrast to the low Reynolds number simulations, here we see a significant improvement in performance on the finest mesh, compared to the two coarser meshes. Whether this is due to the higher velocity, or the smaller Δt (which was 0.0001 here compared to 0.0005 in the low Re simulations) is unknown.

	B_O	P_O	Y_1	Y_2
B_O	1	0.8	0.5	0.5
P_O	1.3	1	0.6	0.7
Y_1	2.2	1.7	1	1.1
Y_2	2.0	1.5	0.9	1

Table 26: Comparison of average computation time on one BiCGStab iteration for each preconditioner, during the arterial blood flow simulations where $\max \mathrm{Re}_{inlet} \approx 200$. The value in cell (i, j) is the ratio of the computation time for preconditioner i by the computation time for preconditioner j. Thus, if the number in cell (i, j) is greater than one then preconditioner i used more time per BiCGStab iteration than preconditioner j. The values were taken from the largest simulations, the mesh referred to as rf2.

Bifurcation Aneurysm, P_O			
Mesh	Iterations	St.dev.	Time
rf0	34	1	18.6
rf1	35	2	35.6
rf2	35	3	63.9

(a)

Bifurcation Aneurysm, B_O			
Mesh	Iterations	St.dev.	Time
rf0	82	7	23.6
rf1	87	10	55.9
rf2	99	16	137.7

(b)

Table 27: Tables of average iterations with standard deviation, and computation time per time step when using the preconditioners B_O and P_O. $\max \boldsymbol{u}_{inlet}(t) = 302\,\mathrm{mm\,s^{-1}}$ and $\Delta t = 0.0005$.

Bifurcation Aneurysm, Y_1			
Mesh	Iterations	St.dev.	Time
rf0	30	1	16.9
rf1	28	2	34.2
rf2	29	3	89.0

(a)

Bifurcation Aneurysm, Y_2			
Mesh	Iterations	St.dev.	Time
rf0	27	1	13.6
rf1	26	2	32.9
rf2	27	5	74.4

(b)

Table 28: Tables of average iterations with standard deviation, and computation time per time step when using the two Yosida variants. $\max \boldsymbol{u}_{inlet}(t) = 302\,\mathrm{mm\,s^{-1}}$ and $\Delta t = 0.0005$.

Bifurcation Aneurysm, P_O			
Mesh	Iterations	St.dev.	Time
rf0	39	1	8.3
rf1	40	2	17.8
rf2	34	2	42.2

(a)

Bifurcation Aneurysm, B_O			
Mesh	Iterations	St.dev.	Time
rf0	110	10	20.4
rf1	116	11	42.0
rf2	87	9	91.0

(b)

Table 29: Tables of average iterations with standard deviation, and average computation time per time step when using the preconditioners B_O and P_O. $\max \boldsymbol{u}_{inlet}(t) = 605\,\mathrm{mm\,s^{-1}}$ and $\Delta t = 0.0001$.

Bifurcation Aneurysm, Y_1			
Mesh	Iterations	St.dev.	Time
rf0	68	3	21.8
rf1	62	3	43.2
rf2	34	1	69.6

(a)

Bifurcation Aneurysm, Y_2			
Mesh	Iterations	St.dev.	Time
rf0	48	3	17.2
rf1	50	4	36.2
rf2	29	1	64.4

(b)

Table 30: Tables of average iterations with standard deviation, and average computation time per time step when the two Yosida variants. $\max \boldsymbol{u}_{inlet}(t) = 605\,\mathrm{mm\,s^{-1}}$ and $\Delta t = 0.0001$.

7 Discussion and conclusion

The block diagonal preconditioner B_ϵ was studied extensively in ([6], [26]), and we tested it in this thesis primarily as a basis for comparison. The calculated values $\kappa_2(B_\epsilon A_S)$ we list in section 6.1.1 are approximately equal to the same results in those papers. The iteration numbers in table 18 show that the preconditioner works well in practice, when implemented in FEniCS and cbc.block. From the results in tables 13 and 22 the adaption of the preconditioner to the Oseen equations seems to work well, both keeping the eigenvalues clustered and leading to low iteration numbers that don't increase much when the mesh is refined. The caveat is that if the convective term is too dominant, due to high velocities or too large timesteps, the performance of the preconditioner deteriorates. Tables 27b and 29b shows that although B_ϵ leads to a higher iteration count on average, its simpler construction compared to the other preconditioners considered makes it more competitive when one considers computation time.

The Yosida preconditioner Y_1 is not useful when F is not dominated by $\frac{1}{\Delta t}M$. This means, when convection is absent, that $\frac{1}{\Delta t}$ need to be much larger than ν. When convection is present, there is an additional requirement that $\frac{1}{\Delta t}M$ also should dominate the convective term $C(\boldsymbol{u}_1)$. This is evident from tables 20 and 24. In section 6.3 we see that Y_1, in the right conditions, can work approximately as well as the other preconditioners in 3D simulations with high kinematic viscosity and velocity if the time step is small enough.

The Yosida preconditioner variant Y_2 performed well in the structured tests. It was shown to not be as robust as B_ϵ and P_ϵ, with eigenvalues and iteration numbers varying more with different parameter values and grid size. Table 28b shows that Y_2 was the most efficient of the preconditioners in the low Reynolds number 3D case in terms of number of iterations. This can be partly explained by the Reynolds number being small during most of the simulation, as can be seen from the inlet velocity profile in figure 9a, if the Y_2 preconditioner performs better with a low time step and low Reynolds number. In the moderate Reynolds number flow simulation Y_2 performed best in terms of iterations on the finest mesh, but used more time than P_ϵ.

The overall best performing preconditioner, according to the results presented in the previous chapter, is the projection method based preconditioner P_ϵ suggested in [7]. Using it results in the tightest eigenvalue spread and on average the lowest number of iterations with the BiCGStab algorithm. The number of iterations when using P_ϵ also vary little when velocity varies in time, contrary to B_ϵ. Tables 19 and 23 show the number of iterations when using P_O staying below 25 for all parameter combinations, except when convection becomes too dominant. A limit on the number of iterations per time step is also from table 27a, where the average is about 35 for all mesh resolutions. The results of the moderate Reynolds flow number simulations are more ambiguous, with the average iteration count being 39 and 40 on the two coarser meshes, and 34 on the finest mesh.

As expected none of the preconditioners stay order optimal, in the sense of the number of BiCGStab iterations being independent of the grid resolution, when the flow problem being solved is dominated by convection. This means that when velocities are large or viscosity is small, i.e. when the Reynolds number is of order $> 10^3$, using a coupled solver with one of the four preconditioners tested in this thesis is not feasible for finer grids. While the results of the moderate reynolds number flow show the performance of each preconditioner increasing with mesh refinement, the time step in those simulations was so small that the problem was not dominated by the convection. Experiments with using larger magnitude velocity boundary conditions, without decreasing Δt, all failed due to BiCGStab not converging within 400 iterations.

The tables of condition numbers for the preconditioned operators, when compared with the tables of eigenvalue ratios and iteration numbers, suggest that $\kappa_1(P^{-1}A) = \sigma_{\max}/\sigma_{\min}$ is not a useful indicator of preconditioner performance in the preconditioned BiCGStab algorithm. Conversely, the ratio $\kappa_2(P^{-1}A)$ of the largest eigenvalue by the smallest, by moduli, is seen to be a good indicator of preconditioner performance. This is consistent with what is argued for B_ϵ in section 4.2 in [6]. We did not see any clear relationship between the distribution in the complex plane of the eigenvalues of the preconditioned operators, plotted in section 6.1.2, and the performance of the preconditioners.

In [21] it is argued that for both the CGS and GMRES algorithms the speed of convergence to a given error norm is determined not by eigenvalues or singular values, but by the pseudospectrum of the coefficient matrix. It is possible that this is the case for the BiCGStab algorithm as well, but constraints on time and resources limited our investigation in this direction. Preliminary calculations suggest that random perturbations of the preconditioned operators do not lead to much greater perturbations in their eigenvalues.

This author is not aware of other work on the relationship between the properties of the preconditioned discrete Oseen equations and the convergence of the BiCGStab method. For work on the GMRES method applied to the preconditioned Oseen problem, and some other approaches to solving the discrete Navier-Stokes equations within a finite element framework, we refer to chapter 8 in the book by Elman, Silvester and Wathen [3].

8 A. Source code

We only include the implementation of the preconditioners used for the simulations in sections 6.2 and 6.3 here. The rest of the source code is available on the web address http://www.bitbucket.org/krisbrox/thesis.

8.1 A.1 cbc.block implementation of preconditioners

```
from block import *
from dolfin import *
from block.algebraic.petsc import collapse, ILU, InvDiag
from block.algebraic.petsc import AMG, SOR, LumpedInvDiag

class B_eps():
    def __str__(self):
        return 'B_eps'
    def generate_prec(self, AA, p, q, u, v, P, bcu=None, bcp=None,
                      u_1=None, dim=None):
        mu = P.mu;   dt = P.dt
        rho= P.rho; nu = mu/rho

        [[A, B], [C, _]] = AA

        if bcp == None:
            Kp_dt     = assemble(dt*(dot(grad(p), grad(q))*dx + p*q*dx))
        else:
            Kp_dt     = assemble(dt*(dot(grad(p), grad(q))*dx))
        I_p_nu    = assemble((1/nu) * p*q*dx)

        # Dirichlet bcs for the pressure
        if bcp != None:
            for bc in bcp:
                bc.apply(I_p_nu)
                bc.apply(Kp_dt)

        Sp = AMG(Kp_dt) + AMG(I_p_nu)
        P  = block_mat([[AMG(A),      0],
                        [0,           Sp]])
        return P
```

```
class P_eps():
    def __str__(self):
        return 'P_eps'

    def generate_prec(self, AA, p, q, u, v, P, bcu=None,
                      bcp=None, u_1=None, dim=None):
        rho = P.rho; dt = P.dt; mu = P.mu;
        nu = mu/rho

        [[A, G], [D, _]] = AA

        if bcp == None:
            Kp     = assemble(dot(grad(p),grad(q))*dx+p*q*dx)
            Kp_dt = assemble(dt*(dot(grad(p),grad(q))*dx+p*q*dx))
        else:
            Kp     = assemble(dot(grad(p), grad(q))*dx)
            Kp_dt = assemble(dt * (dot(grad(p), grad(q))*dx))
        I_p_nu    = assemble((1/nu) * p*q*dx)

        # Set bcs for the preconditioner
        if bcp != None:
            for bc in bcp:
                bc.apply(I_p_nu)
                bc.apply(Kp)
                bc.apply(Kp_dt)

        Sp = AMG(Kp_dt) + AMG(I_p_nu)


        P1 = block_mat([[1,          G*AMG(Kp)],
                        [0,                 Sp]])
        P2 = block_mat([[1,                  0],
                        [D,                 -1]])
        P3 = block_mat([[AMG(A),             0],
                        [0,                  1]])
        P = P1*P2*P3
        return P
```

```
class P_eps_NS():
    def __str__(self):
        return 'P_eps_NS'

    def generate_prec(self, AA, p, q, u, v, P, bcu=None,
                      bcp=None, u_1=None, dim=None):
        rho = P.rho; dt = P.dt; mu = P.mu;
        nu  = mu/rho

        [[F, G], [D, _]] = AA

        if bcp == None:
            Kp    = assemble(dot(grad(p), grad(q))*dx + p*q*dx)
            Kp_dt = assemble(dt*(dot(grad(p), grad(q))*dx + p*q*dx))
        else:
            Kp    = assemble(dot(grad(p), grad(q))*dx)
            Kp_dt = assemble(dt*(dot(grad(p), grad(q))*dx))
        Mp          = assemble(p*q*dx)
        Mp_nu       = assemble((1/nu) * p*q*dx)
        Mu          = assemble(dot(u,v)*dx)
        C           = assemble(inner(dot(u_1, nabla_grad(p)), q)*dx)

        if bcp != None:
            for bc in bcp:
                bc.apply(Mp_nu)
                bc.apply(Kp)
                bc.apply(Kp_dt)
                bc.apply(C)

        Sp1 = SOR(Mp_nu)
        Sp2 = AMG(Kp_dt)
        Sp3 = AMG(Kp)*C
        Sp = Sp1+Sp2+Sp3

        P1 = block_mat([[1,          G*AMG(Kp)],
                        [0,                 Sp]])
        P2 = block_mat([[1,                  0],
                        [D,                 -1]])
        P3 = block_mat([[AMG(F),             0],
                        [0,                  1]])
        P = P1*P2*P3
```

```
return P
```

```
class B_eps_NS():
    def __str__(self):
        return 'B_eps_NS'

    def generate_prec(self, AA, p, q, u, v, P, u_1=None,
                      bcu=None, bcp=None, dim=None):
       mu = P.mu; dt = P.dt; rho = P.rho;
       nu = mu/rho

       [[A, B], [C, _]] = AA

       if bcp == None:
         Kp     = assemble((dot(grad(p),grad(q))*dx+p*q*dx))
         Kp_dt = assemble(dt*(dot(grad(p),grad(q))*dx+p*q*dx))
       else:
         Kp     = assemble((dot(grad(p),grad(q))*dx))
         Kp_dt = assemble(dt*(dot(grad(p),grad(q))*dx))

       conv     = assemble(inner(dot(u_1,nabla_grad(p)),q)*dx)
       I_p_nu   = assemble((1/nu)*p*q*dx)

       if bcp != None:
           for bc in bcp:
               bc.apply(I_p_nu)
               bc.apply(Kp)
               bc.apply(Kp_dt)
               bc.apply(conv)

       Sp1 = SOR(I_p_nu)
       Sp2 = AMG(Kp_dt)
       Sp3 = AMG(Kp)*conv
       Sp  = Sp1 + Sp2 + Sp3

       P   = block_mat([[AMG(A),          0],
                        [0,              Sp]])
       return P
```

```
class Y_2():
   def __str__(self):
      return 'Yosida_2'

   def generate_prec(self, AA, p, q, u, v, P, bcu=None,
                     bcp=None, u_1=None, dim=3):

      [[F, G], [D, Q]]        = AA

      Fp = AMG(F)
      Mp = assemble(p*q*dx)
      if bcp != None:
         for bc in bcp:
            bc.apply(Mp)

      if dim == 2:
         S = D*InvDiag(F)*G + Mp
      elif dim == 3:
         S = D*InvDiag(F)*G

      Sp = AMG(collapse(S))

      p1 = block_mat([[Fp, 0],[0, 1]])
      p2 = block_mat([[1, -G],[0, 1]])
      p3 = block_mat([[F, 0],[0, 1]])

      p4 = block_mat([[1, 0],[0, -Sp]])
      p5 = block_mat([[1, 0],[-D, 1]])
      p6 = block_mat([[Fp, 0],[0, 1]])

      P = p1*p2*p3*p4*p5*p6
      return P
```

```
class Y_1():
    def __str__(self):
        return 'Yosida_1'

    def generate_prec(self, AA, p, q, u, v, P, bcu=None,
                      bcp=None, u_1=None, dim=3):
        mu = P.mu
        dt = P.dt

        [[F, G],
         [D, Q]]  = AA

        Fp = AMG(F)

        Mp = assemble(p*q*dx)
        Mudt = assemble((1/dt)*inner(u,v)*dx)

        if bcu != None:
            for bc in bcu:
                bc.apply(Mudt)
        if bcp != None:
            for bc in bcp:
                bc.apply(Mp)

        S = D*LumpedInvDiag(Mudt)*G + Mp
        Sp = AMG(collapse(S))

        p1 = block_mat([[Fp, 0],[0, 1]])
        p2 = block_mat([[1, -G],[0, 1]])
        p3 = block_mat([[F, 0],[0, 1]])

        p4 = block_mat([[1, 0],[0, -Sp]])
        p5 = block_mat([[1, 0],[-D, 1]])
        p6 = block_mat([[Fp, 0],[0, 1]])

        P = p1*p2*p3*p4*p5*p6
        return P
```

References

[1] V.D. Tsakanikas. *Intravascular Imaging: Current Applications and Research Developments: Current Applications and Research Developments.* Advances in Bioinformatics and Biomedical Engineering:. IGI Global, 2011.

[2] Simone Deparis, Gwendol Grandperrin, and Alfio Quarteroni. Parallel preconditioners for the unsteady navier-stokes equations and applications to hemodynamics simulations. *Computers and Fluids*, 92:253–273, 2014.

[3] Howard Elman, David J Silvester, and Andrew J Wathen. *Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics.* Oxford University Press, 2005.

[4] Michele Benzi and Gene H Golub. A preconditioner for generalized saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):20–41, 2004.

[5] Timo Heister and Gerd Rapin. Efficient augmented lagrangian-type preconditioning for the oseen problem using grad-div stabilization. *International Journal for Numerical Methods in Fluids*, 71(1):118–134, 2013.

[6] Kent-Andre Mardal and Ragnar Winther. Uniform preconditioners for the time-dependent stokes equations. *Numerische Mathematik*, 98:305–327, 2004.

[7] Mingchao Cai. Analysis of some projection method based preconditioners for models of incompressible flow. *Applied Numerical Mathematics*, 90:77–90, 2015.

[8] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method.* Springer, 2012.

[9] Kent-Andre Mardal and Joachim Berdal Haga. Block preconditioning of systems of pdes. In Anders Logg, Kent-Andre Mardal, and Garth Wells, editors, *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, pages 643–655. Springer Berlin Heidelberg, 2012.

[10] F.M. White. *Viscous Fluid Flow.* McGraw-Hill series in mechanical engineering. McGraw-Hill, 1991.

[11] Hans Petter Langtangen, Kent-Andre Mardal, and Ragnar Winther. Numerical methods for incompressible viscous flow. *Advances in Water Resources*, 25(8):1125–1146, 2002.

[12] Lawrence C. Evans. *Partial Differential Equations (Graduate Studies in Mathematics, V. 19) GSM/19.* American Mathematical Society, 1998.

[13] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems.* Studies in Mathematics and its Applications. Elsevier Science, 1978.

[14] Hans Petter Langtangen, Kent-Andre Mardal, and Ragnar Winther. Numerical methods for incompressible viscous flow. *Advances in Water Resources*, 25(8–12):1125 – 1146, 2002.

[15] Ivo Babuška. The finite element method with lagrangian multipliers. *Numerische Mathematik*, 20(3):179–192, 1973.

[16] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 8(R2):129–151, 1974.

[17] Kent-Andre Mardal, Joachim Schöberl, and Ragnar Winther. A uniform inf–sup condition with applications to preconditioning. *arXiv preprint arXiv:1201.1513*, 2012.

[18] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

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

[20] P.R. Graves-Morris. The breakdowns of bicgstab. *Numerical Algorithms*, 29(1-3):97–105, 2002.

[21] Noël M. Nachtigal, Satish C. Reddy, and Lloyd N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM Journal on Matrix Analysis and Applications*, 13(3):778–795, 1992.

[22] A. Pyzara, B. Bylina, and J. Bylina. The influence of a matrix condition number on iterative methods' convergence. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pages 459–464, Sept 2011.

[23] Trygve K Nilssen, Gunnar A Staff, and Kent-Andre Mardal. Order optimal preconditioners for fully implicit runge-kutta schemes applied to the bidomain equations. *Numerical Methods for Partial Differential Equations*, 27(5):1290–1312, 2011.

[24] Jörg Liesen and Petr Tichý. Convergence analysis of krylov subspace methods. *GAMM-Mitteilungen*, 27(2):153–173, 2004.

[25] J. Bergh and J. Löfström. *Interpolation spaces: an introduction.* Grundlehren der mathematischen Wissenschaften. Springer, 1976.

[26] Kent-Andre Mardal and Ragnar Winther. Preconditioning discretizations of systems of partial differential equations. *Numerical Linear Algebra with Applications*, 18:1–40, 2000.

[27] Alfio Quarteroni, Fausto Saleri, and Alessandro Veneziani. Factorization methods for the numerical approximation of navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 188(1–3):505 – 526, 2000.

[28] David Kay, Daniel Loghin, and Andrew Wathen. A preconditioner for the steady-state navier–stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, 2002.

[29] J.W. Pearson, M. Stoll, and A.J. Wathen. Regularization-robust preconditioners for time-dependent pde-constrained optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1126–1152, 2012.

[30] John W. Eaton, David Bateman, and Søren Hauberg. *GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations.* CreateSpace Independent Publishing Platform, 2009. ISBN 1441413006.

[31] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, and Hong Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2015.

[32] J Jiang, K Johnson, K Valen-Sendstad, K-A Mardal, O Wieben, and Strother C. Flow characteristics in a canine aneurysm model: A comparison of 4d accelerated phase-contrast mr measurements and computational fluid dynamics simulations. *Medical Physics*, 38(11), 2011.

[33] Øyvind Evju and Martin S Alnæs. cbcflow. `http://www.bitbucket.org/simula_cbc/cbcflow`, 2015.