ILU and Machine Learning Based Preconditioning For The Discretized Incompressible Navier-Stokes Equations

by
Rita Stanaityte

A dissertation submitted to the Department of Mathematics,

College of Natural Sciences and Mathematics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Mathematics

University of Houston
May 2020

ACKNOWLEDGMENTS

First and foremost, I would like to express sincere gratitude to my research supervisor, Prof. Maxim Olshanskii. Without his assistance and dedicated involvement in every step, this work would have never been accomplished. I would like to thank you very much for your support and understanding over these years.

Completing this work would have been all the more difficult if not for the support from my colleagues at the Department of Mathematics of University of Houston. I am indebted to them for their help. In particular, I would like to thank Kyle Williams for his help with writing the code and generating datasets discussed in Chapter 7.

Nobody has been more important to me in the pursuit of this project than my family. I would like to thank my parents and my sister, whose love and guidance are with me in whatever I pursue. I am particularly thankful to my grandmother. Her pedagogical talent is greatly responsible for my love of mathematics. I would like to thank my best friends, Flora and Zhofrua, whom I miss dearly and who are always my best support.

I am also grateful to my friends for providing support and friendship that I needed. To Anna, for helping me survive all the stress from these years and not letting me give up. To Vladimir, for listening, offering me advice and commenting on earlier versions of the manuscript. I cannot forget friends who went through hard times together and cheered me on: Jea, Prajakta, Alan and many others.

Last, but certainly not least, I would like to express gratitude to my husband, Vitalii, who has given me the extra strength and motivation to get things done.

ABSTRACT

Motivated by the numerical solution of the incompressible Navier-Stokes equations, this dissertation studies numerical properties of threshold incomplete LU factorizations for nonsymmetric saddle-point matrices. We consider preconditioned iterative Krylov-subspace methods, such as GMRES, to solve large and sparse linear algebraic systems that result from Galerkin finite element (FE) discretizations of the linearized Navier-Stokes equations. The corresponding preconditioners are used to accelerate the convergence of the GMRES method. Stabilized and unstabilized finite element methods are used for the Navier-Stokes problem leading to systems of algebraic equations of a saddle point type, which has a 2×2-block structure. Numerical experiments for model problems of a driven cavity flow and flow over a backward-facing step illustrate the performance of one-parameter and two-parameter ILU factorizations as preconditioners.

We also introduce a Machine Learning (ML) based approach for building ILU factorizations for preconditioning. For this purpose, we use the tools well-developed in the scope of image segmentation. Image Segmentation is the process of partitioning an image into separate and distinct regions. The process has some similarities to building patterns for ILU-type preconditioner. In our interpretation, the segmented regions represent a non-zero pattern for L and U factors. We applied a convolutional neural network with the benchmark U-net architecture to predict non-zero patterns for ILU-type factorizations and further use the resulting preconditioners to solve the discrete linearized Navier-Stokes system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

This research is motivated by the numerical solution of the Navier-Stokes equations governing the flow of viscous incompressible Newtonian fluids. The system of Navier-Stokes equations describes the dynamics of fluid flow in terms of its velocity, pressure, and density. These equations were derived independently by G.G. Stokes and M. Navier, in the early 1800s. The Navier-Stokes system is a set of coupled differential equations, and for some given model flow problems, its solution can be found using analytical tools. However, for most practical flow problems, these equations are too difficult to solve analytically. Numerical modeling of viscous incompressible fluid flows is also challenging. In particular, the discretization of the Navier-Stokes equations by any conventional discretization method, like the Finite Element method (FE), results in a very large system of algebraic equations. Solving this system of equations is computationally expensive. Moreover, solving equations numerically is known to get harder for higher values of Reynolds number (small values of kinematic viscosity and/or high velocities). Since the fluid flows are ubiquitous in physics, engineering, biology and everyday life, there is still a pressing need for new methods and tools to solve such problems numerically.

In this dissertation, we study and use matrix factorization as a tool to improve iterative solvers for systems of linear algebraic equations with non-symmetric saddle-point matrices. Our work is focused on practical use and improvement of the preconditioning techniques originally introduced by Kaporin [23] for symmetric positive definite (SPD) matrices. Here it is applied to non-symmetric matrices and the generalized minimal residual method (GMRES) as a solver. Numerical analysis and experiments in [26] and [25] with the inf-sup stable Galerkin finite element methods for the incompressible Navier-Stokes equations demonstrate the robustness and efficiency of this approach.

In the present dissertation, we extend the method and analysis to the system of algebraic equations resulting from the stabilized formulations of the Navier-Stokes equations

$$\begin{cases} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega \\ \qquad\qquad\qquad div(\mathbf{u}) = 0 \text{ in } \Omega, \end{cases} \tag{1}$$

where $\mathbf{u}$ is the flow velocity, and p is the pressure field, with appropriate boundary conditions on $\partial\Omega$, the boundary of the Lipschitz domain $\Omega \subset R^3$. Implicit time discretization or linearization of the Navier-Stokes in Picard fixed-point iteration, and further application of a FE method results in the problem of the form

$$\underbrace{\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix}}_{\mathcal{A}} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2}$$

where A is sparse $n \times n$ positive definite matrix, which combines the discretization of the diffusion, convection, and time-dependent terms; u and p represent the discrete velocity and pressure, respectively; B^T is the discrete gradient, B is the discrete (negative) divergence, C is the matrix resulting from possible pressure stabilization terms, f and g contain forcing and boundary terms. We study several preconditioning techniques that improve the convergence of the GMRES iterations for (2).

In the last two decades, a lot of work has been done to develop preconditioned iterative methods in application to incompressible flow problems. One can find an excellent review of preconditioning techniques for saddle-point problems arising in computation fluid dynamics in [3]. One of the standard techniques to design a preconditioner for the matrix $\mathcal{A}$ from (2) is based on building individual preconditioners for the submatrix A and Schur complement matrix $S = BA^{-1}B^T + C$ and combining them in a 2×2 block matrix. It was shown in [14] that Krylov subspace methods demonstrate a noticeable improvement of performance for solving the Navier-Stokes equations when using the pressure convection-diffusion [24] and a least-square commutator [11] preconditioners for the pressure Schur complement of the system. In the same paper mesh-independent convergence rates were observed for specific flow problems for which previous versions of the methods did not exhibit such behavior. Thus, one of the most popular techniques to improve convergence of iterative methods in incompressible fluid dynamics is based on block 2×2 preconditioners. Important subclasses are formed by block diagonal and block triangular preconditioners for the full system $\mathcal{A}\mathbf{u} = \mathbf{b}$. Block diagonal preconditioners have been extensively studied in [37]; also in the context of applications beyond fluid dynamics. Block triangular preconditioners were first suggested in [10] and gain a lot of attention over the last two decades. This class of preconditioners includes some of the most effective solvers for saddle point problems, both symmetric and nonsymmetric. Paper [31] reviews possible advantages and difficulties of using various Schur complement preconditioners, recalls existing eigenvalue bounds for the preconditioned Schur complement and proves such for the block triangular preconditioner. Comparing to the Stokes problem, the situation is more complicated for nonsymmetric saddle point problems resulting from the discretization of the Oseen equations. For steady problems with small Reynolds numbers, using a pressure mass matrix to approximate the Schur complements is usually sufficient and results in convergence rate independent of mesh size. This uniform convergence is observed for both block diagonal and block triangular preconditioners. Such preconditioners were also used in [13] for the linear systems arising from Newton's method applied to the Navier-Stokes equations.

The area of developing robust preconditioners for fluid problems has expanded in recent years. In particular, robust preconditioners of the block triangular type for the Oseen equations are of interest [12]. A block triangular preconditioner based on the pressure convection-diffusion operator was also used in [38] with good results as a smoother for a multigrid method applied to the Oseen problem. It was shown in [36] that block preconditioners result in an efficient approach for the solution of incompressible Navier-Stokes equation discretized by finite elements and linearized by Newton or Picard's methods.

However, another feasible approach for building preconditioners is based on the incomplete elementwise factorization of the original matrix $\mathcal{A}$. In [41], new reordering techniques were introduced for the degrees of freedom that make the application of ILU preconditioner to saddle point problems competitive with the most advanced block preconditioners. The principle that the preconditioning matrix should have the same 2×2 block structure as the original saddle point matrix is called constraint preconditioning. Constrained preconditioners have been widely used in the solution of saddle point systems arising from mixed finite element formulations of elliptic partial differential equations [16]. This is one of the cases when it is sometimes possible to find an easily invertible -

diagonal matrix that is spectrally equivalent to A.

Bai et al. [1] introduce the Hermitian and skew-Hermitian preconditioning (HSS) as a stationary iterative method, where they show convergence for non-Hermitian positive definite systems. The application of HSS as a preconditioner for rather general saddle point problems has been studied in [3]. Unfortunately the rate of convergence of the HSS iteration is rather slow, even with the optimal choice of a parameter. Because of that, Benzi and Golub proposed in [2] that GMRES or other Krylov subspace methods should be used to accelerate the convergence of the HSS method. Another type of iterative solvers for the Navier-Stokes equations in rotation form has been introduced and studied in [29] and [30]. Although the rotation form is not widely used in practice, it has some advantages over the (standard) convective form.

One more type of preconditioners was introduced by Benzi and Olshanskii in [4] in 2006 based on the idea of augmenting the A block of the saddle-point matrix with a term depending on the constraint. The authors named it the augmented Lagrangian based approach. Different variants of the augmented Lagrangian (AL)-based block-triangular preconditioners were studied in [5]. This type of preconditioners is used to accelerate the convergence of the GMRES applied to various finite element discretizations of the Oseen problem in two and three space dimensions. The preconditioner is based on an algorithm that relies on a highly specialized multigrid method involving a custom prolongation operator and a special smoothing iteration. To generalize the preconditioner to three dimensions, in [17] alternative finite elements were proposed for the velocity and prolongation operators for which the preconditioner works robustly. Field-of-Value convergence analysis of the GMRES with (AL)-preconditioner for Oseen problem can be found in [4]. AL-preconditioning remains an active area of research.

In [26] a matrix decompostion of the form

$$A = LU + LR_u + R_\ell U - E, \tag{3}$$

is introduced and investigated, where U and L are upper and lower triangular matrices and R_u

and R_l are strictly upper and lower triangular matrices respectively, E is the corresponding error matrix. Given two small parameters $0 < \tau_2 \leq \tau_1$, the off-diagonal elements of U and L are either zero or have absolute values greater than τ_1. The absolute values of R_u and R_l entries are either zero or belong to $(\tau_2, \tau_1]$.

We study the numerical performance of the method for a set of linear algebraic systems that appear in the simulation of driving cavity flow and flow over backward-facing step. We conduct case studies of steady incompressible flow in a lid-driven square cavity for $1/1000 \leq \nu \leq 1$ and flow over backward-facing step for $1/400 \leq \nu \leq 1$. The statistics we are interested in are the level of fill-in of U and L factors and the average iteration numbers. We perform various qualitative and quantitative comparisons to determine the effect of Reynolds number and mesh size for four different finite elements discretizations of the governing equations. Furthermore, we compare the performance of two-parameter factorization $ILU(\tau_1, \tau_2)$ and a standard one parameter factorization for different values of discretization, stabilization and threshold parameters.

In Chapter 6, we determine the optimal threshold parameter and show superiority of $ILU(\tau_1, \tau_2)$ comparing its performance to some state-of-the-art block preconditioners. We analyze the results of using GMRES with $ILU(\tau_1, \tau_2)$ and GMRES with pressure convection-diffusion (PCD) or the least-squares commutator (LSC) as a preconditioner. We examine iteration numbers and building times for various problem sizes and Reynolds numbers.

In Chapter 7 we introduce a Machine Learning (ML) based approach for building ILU factorizations for preconditioning. For this purpose, we employ ML tools developed in the scope of image segmentation. In the last decade, astounding results were shown for image segmentation tasks. We noticed that image segmentation problems is similar to a certain extent to finding a pattern based on matrix entries. It refers to the process of associating each entry in a matrix to a label. We consider a U-net architecture suggested in [34], which achieves an excellent performance on different segmentation applications. U-Net was first designed for medical image segmentation, but it was later successfully used in many other fields and for different tasks. In Chapter 7, we propose a deep learning algorithm based on U-net architecture for computing the ILU preconditioners' pattern as an output of a neural network.

The rest of this dissertation is organized as follows. Chapter 2 discusses the physical background of the Navier-Stokes equations. In Chapter 3, we consider discretization in time and linearization techniques. In Chapter 4, the discretization of the incompressible Navier-Stokes by the finite element discretization is introduced. Chapter 5 deals with ILU type and block preconditioners. In Chapter 6, some of these preconditioners are compared for solving several standard benchmark problems. Lastly, in Chapter 7, we present deep learning techniques for preconditioners' design and in Chapter 8 conclusions.

2 Navier-Stokes equations

2.1 Mathematical model

To obtain the equation of motion for a fluid, we need to start with conservation laws applied to a material volume $W_0 \subset \Omega$ in a domain occupied by fluid. Let us introduce Eulerian coordinates of a point in Ω by $x = (x_1, x_2, x_3)$. Let $\boldsymbol{X}$ be a material point in Ω at $t = t_0$, where t_0 is fixed and from $(0, T)$, and ξ is the Eulerian coordinates of the point $\boldsymbol{X}$ at time t. Then we define a mapping:

$$t \to \boldsymbol{X}_\xi(t), t \in (t_0 - \delta, t_0 + \delta), \tag{4}$$

where $\delta > 0$ is sufficiently small such that all functions presented below will be well-defined and $\mathbf{X}_\xi$ describes the coordinates of the point $\mathbf{X}$. Then (4) describes the trajectory of the point $\boldsymbol{X}$.

Therefore, the trajectory of the point $\boldsymbol{X}$ in a velocity field $\mathbf{u} = \mathbf{u}(x, t) \in \mathrm{R}^3$ is given by the solution of the system:

$$\frac{d}{dt}\boldsymbol{X}_\xi(t) = \mathbf{u}(\boldsymbol{X}_\xi(t), t), t \in (t_0 - \delta, t_0 + \delta), \boldsymbol{X}_\xi(t_0) = \xi \tag{5}$$

There are two ways to describe the motion of a fluid (Figure 1):

- Eulerian specification, we record the evolution of the flow properties at every fixed point in space as time varies.
- Lagrange specification, where we follow fluid particles (material points) as they travel in the flow.

For the given material volume W_0 at time $t = t_0$, let

$$W(t) := \{\boldsymbol{X}_\xi(t) : \xi \in W_0\} \tag{6}$$

First we consider the conservation of mass. For the given material volume, the conservation of

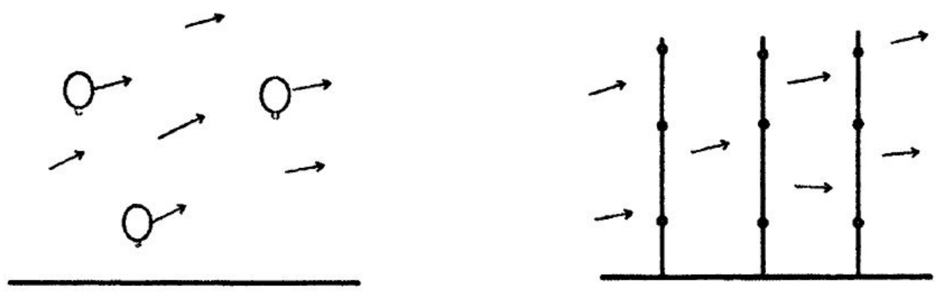

Figure 1: 1) Lagrange approach: the fluid flow properties are determined by tracking the motion and properties of the particles as they move in time. 2) Eulerian approach: parameters are measured with monitoring devices on a masts

mass can be written as

$$\frac{d}{dt}\int_{W(t)} \rho dx = 0, \tag{7}$$

where ρ is the density of the fluid.

Another necessary notion we need further in this thesis is that of material derivative. The material derivative is the rate of change of an intensive property f on a particle passively advected by the velocity field $\mathbf{u}$. The material derivative is defined as:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla) f. \tag{8}$$

Also, we need another important theorem, which is known as the Reynolds Transport Theorem. It provides a way to transfer equations for conservation laws from the Lagrangian point of view to the Eulerian point of view using a control surface and a control volume. For a scalar smooth function $f = f(x,t)$ the Reynold's transport theorem results in the identity:

$$\begin{aligned} \frac{d}{dt}\int_{W(t)} f(x,t)dx &= \int_{W(t)} \frac{df}{dt}(x,t) + f div(\mathbf{u}(x,t))dx = \\ &= \int_{W(t)} \frac{\partial f}{\partial t}(x,t) + \mathbf{u}\cdot\nabla f(x,t) + f div(\mathbf{u}(x,t))dx \\ &= \int_{W(t)} \frac{\partial f}{\partial t}(x,t) + div(f\mathbf{u})(x,t)dx \end{aligned} \tag{9}$$

Using the formulas (9) in (7) we get:

$$\int_{W(t)} \frac{\partial \rho}{\partial t} + div(\rho \mathbf{u}) dx = 0 \tag{10}$$

Since (10) holds for any material volume $W(t)$, with the assumption that $\rho = const$ we get the continuity equation

$$div(\mathbf{u}) = 0 \tag{11}$$

Next, we consider the conservation of momentum. Newton's second law applied to a material volume $W(t)$ leads to the following relation:

$$\frac{d}{dt} \int_{W(t)} \rho dx = F(t), \tag{12}$$

where F(t) are forces acting on W(t). We decompose forces acting on the volume to external force $F_1(t) = \int_{W(t)} \rho \mathbf{g} dx$, i.e. gravity, and $F_2(t)$, which describes internal forces. If $\mathbf{a}$ is the internal force, then $F_2(t) = \int_{\partial W(t)} \mathbf{a} \cdot \mathbf{n} ds$. F_2 represents forces, which are acting on the surface, imposed by the outside fluid 'tugging' or 'pushing' on the surface by means of friction. In 3D, each of the 3 sets of surface planes bounding an element experiences a 3-component force, giving 9 components all together. These form the stress tensor σ, defined so that the force f exerted per unit area across a surface element, is equal to $\sigma \cdot \mathbf{n}\ \mathbf{ds} = \mathbf{n} ds$, where $\mathbf{n} = \mathbf{n}(x,t)$ is the outer normal unit on $\partial W(t)$ (Figure 2). We conclude that $F_2(t) = \int_{\partial W(t)} \sigma \mathbf{ds}$.
One more important theorem is needed. It is the Gauss divergence theorem:

$$\int_{\Omega} div(\mathbf{a}) d\Omega = \int_{\partial\Omega} \mathbf{a} \cdot \mathbf{n} d\Gamma \tag{13}$$

The boundary conditions are applied by evaluating the boundary integral in (13), if possible.

Implementing the divergence theorem, we arrive at the updated Newton's second law for the

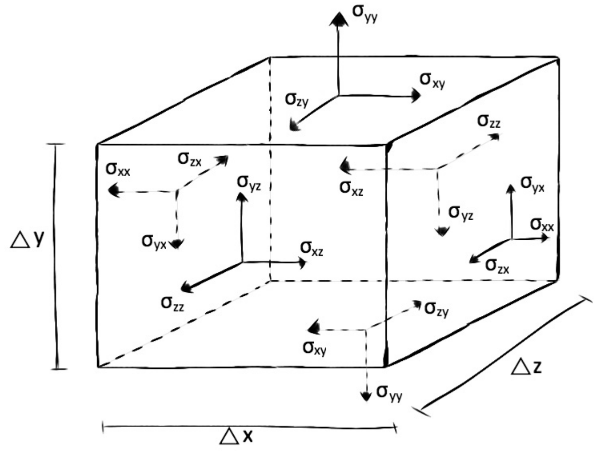

Figure 2: Surface stresses on a fluid element.

forces:

$$\frac{d}{dt}\int_{W(t)} \rho dx = F_1(t) + F_2(t) = \int_{W(t)} \rho\mathbf{g} + div(\sigma)dx \tag{14}$$

Applying the Reynold's Transport theorem to the left-hand side of (14), we get

$$\int_{W(t)} \frac{\partial \rho u_i}{\partial t} + div(\rho u_i(u))dx = \int_{W(t)} \rho g_i + div(\sigma_i)dx \tag{15}$$

We define $div(\sigma)$ as a row-wise divergence of sigma. In Cartesian coordinates, the operator is

$$div(\sigma) = \begin{pmatrix} \frac{\partial\sigma_{11}}{\partial x_1} + \frac{\partial\sigma_{12}}{\partial x_2} + \frac{\partial\sigma_{13}}{\partial x_3} \\ \frac{\partial\sigma_{21}}{\partial x_1} + \frac{\partial\sigma_{22}}{\partial x_2} + \frac{\partial\sigma_{23}}{\partial x_3} \\ \frac{\partial\sigma_{31}}{\partial x_1} + \frac{\partial\sigma_{32}}{\partial x_2} + \frac{\partial\sigma_{33}}{\partial x_3} \end{pmatrix} \tag{16}$$

Since (15) is satisfied for any material volume in Ω, we obtain the following identity, in the vector notation:

$$\frac{\partial \rho\mathbf{u}}{\partial t} + div(\rho\mathbf{u}\otimes\mathbf{u}) = \rho\mathbf{g} + div(\sigma), \tag{17}$$

where $\mathbf{u}\otimes\mathbf{u} = (u_i u_j)_{1\le i,j\le 3} \in \mathrm{R}^3$

Using that $div(\rho\mathbf{u}\otimes\mathbf{u}) = \rho(\mathbf{u}\cdot\nabla)\mathbf{u}$ for $\mathbf{u}$ s.t. $div\mathbf{u} = 0$ and $\rho = const$,we obtain momentum equation:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = \rho \mathbf{g} + div(\sigma) \tag{18}$$

The conservation equations derived above, in addition to a few assumptions about the forces and the behaviour of fluids, lead to the equations of motion for fluids. We assume the fluid is homogeneous in the sense that the relationship between stress and rate of strain is the same throughout any given sample. We also assume that the fluid is isotropic, i.e., that there is no preferred direction in space insofar as the relationship between stress and rate of strain is concerned. These assumptions define Stokesian Fluid. Now we make one further assumption. We assume that the relationship between stress and rate of strain is linear. This defines a Newtonian Fluid.

We introduce the stress tensor σ_{ij} as follows: σ_{ij} is the i-component of the internal force on a surface element ds that has a normal $\mathbf{n}$ pointing in the j-direction. The stress tensor σ_{ij} can be decomposed into a hydrostatic pressure term P that only acts normal to the surface $(i = j)$ and a shear stress tensor τ_{ij}:

$$\sigma_{ij} = -P\delta_{ij} + \tau_{ij} \tag{19}$$

The number of unknowns in the conservation of momentum equation can be reduced by replacing the shear stress tensor τ_{ij} with an expression containing the deformation rate tensor ϵ_{ij}, which is a function of spatial derivatives of u.
Assume that the shear stress τ_{ij} on a fluid element is linearly related to the deformation rate ϵ_{ij}, then

$$\tau_{ij} = \mu\epsilon_{ij}. \tag{20}$$

There is also a normal component of deformation but for incompressible fluids this term is zero. Then we get:

$$\epsilon_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{21}$$

Using (21),(19), and (20) in the momentum equation (18), we arrive at the general form of the momentum equation:

$$\rho\frac{\partial}{\partial t}(u_j) + \rho u_k \frac{\partial}{\partial x_k}(u_j) = \frac{\partial}{\partial x_i}(-P\delta_{ij}) + \mu(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}) + \rho f_j. \tag{22}$$

To solve fluid flow problems, we require both the momentum equation (18) and the continuity equation (11), which constitute a complete system subject to the boundary and initial conditions.

Using equations referred above, (11), (22), and kinematic viscosity formula $\nu = \frac{\mu}{\rho}$, we obtain system, which is known as the Navier-Stokes equations for incompressible fluid flow:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} - \nu\Delta\boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p = \mathbf{f}; \\ div(\boldsymbol{u}) = 0; \\ \boldsymbol{u}(0) = \mathbf{u_0} \end{cases} \tag{23}$$

in a bounded Lipschitz domain $\Omega \subset R^3$, and on a time interval $(0, T)$ $(T < \infty)$, for sufficiently smooth initial data $\mathbf{u_0}$.

The flow equations require the set of conditions that act at the domain boundary. In this work, we consider the following boundary conditions on $\partial\Omega = \Gamma_D \cup \Gamma_N$ with $\Gamma_D \cap \Gamma_N = \emptyset$:

- Dirichlet (or no-slip) boundary condition:

$$u = 0 \text{ on } \Gamma_D \tag{24}$$

  The no-slip condition for viscous fluids assumes that at a solid boundary, the fluid will have zero velocity relative to the boundary.

- Neumann boundary condition:

$$\nu\frac{\partial \mathbf{u}}{\partial n} - \mathbf{n}p = 0, \text{ on } \Gamma_N, \tag{25}$$

where **n** denotes outward-pointing normal to the boundary.

The physical meaning of the Neumann boundary condition, which fixes the component of the stress tensor, is that the tangential "slip" velocity, rather than zero, is proportional to the tangential stress.

2.2 Non-dimensionalization of the Navier-Stokes equations

For any hydrodynamic problem the velocity field depends on the coordinates, time and physical parameters of this problem. Then every change of parameters will make changes to entire flow field. To avoid this problem for our experiments, we write the Navier-Stokes equation in dimensionless form. For this we will use characteristic scales for all quantities in a system.

The scaling parameters are L (characteristic length), V (characteristic speed), ρ_1 (characteristic viscosity). Then $x = L\overline{x}$, $\rho = \rho_1\overline{\rho}$, $u = V\overline{u}$, $f = \frac{V^2}{L}\bar{f}$, $p = \rho_1 V^2 \bar{p}$, $\mu = \rho_1 V L \bar{\mu}$. In non-dimensional form, our problem will have only one dimensionless parameter called the Reynolds number

$$Re = \frac{\rho_1 LV}{\mu_1}. \tag{26}$$

The Reynolds number is the ratio of the inertial forces and the viscous forces. For low values of Re, the flow is tipically laminar.In this case the viscous force is comparably more important, and disturbances in the flow are damped out by viscosity. Thus, it is difficult for disturbances to grow and sustain themselves. When Re increased, the flow exhibits more chaotic behaviour resulting in the developed turbulent flows for high Re number. In this thesis we will consider only laminar flows.

The final result is the non-dimensional form of the Navier-Stokes equation (23).

$$\begin{aligned} \frac{\partial \bar{\boldsymbol{u}}}{\partial t} - \frac{1}{Re}\Delta\bar{\boldsymbol{u}} + \bar{\boldsymbol{u}} \cdot \nabla\bar{\boldsymbol{u}} + \nabla\bar{p} = \bar{\mathbf{f}} \\ div(\bar{\mathbf{u}}) = 0 \end{aligned} \tag{27}$$

where over-bar means non-dimensional variables.

2.3 Steady-state Navier-Stokes equations.

The steady-state or stationary Navier–Stokes equations describe flows in the equilibrium state. Such flow fields can be expected in practice if:

- all data of the Navier-Stokes equations (23) do not depend on the time.
- the viscosity ν is sufficiently large, or equivalently, the Reynolds number Re is sufficiently small.

Here we consider boundary-value problems in a domain $\overline{\Omega} = \Omega \cup \partial\Omega$. Where Ω is a bounded domain in R^d with Lipschitz boundary. The steady-state Navier-Stokes equations take the form

$$\begin{cases} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega \\ \qquad\qquad div(\mathbf{u}) = 0 \text{ in } \Omega, \end{cases} \tag{28}$$

with boundary conditions

$$-\nu(\nabla\mathbf{u})\cdot\mathbf{n} + p\mathbf{n} = 0 \text{ on } \Gamma_N \text{ and } \mathbf{u}(x) = \mathbf{w} \text{ on } \Gamma_D, \tag{29}$$

where $\partial\Omega = \Gamma_N \cup \Gamma_D$, n stands for normal component and outward normal direction, $\nu > 0$ is the kinematic viscosity and $\mathbf{u}$, p is the velocity of the fluid and pressure.

2.4 Weak formulation.

Before applying the FEM to solve equation (28) with the given boundary conditions, it is necessary to transform it into a more suitable form. To do that, we will use a weak formulation of the problem.

The weak formulation of equation (28) can be derived by multiplying (28) by a so-called test function v and integrating over the domain. The choice of the class of functions to which v belongs determines whether (28) has a solution for ν sufficiently large or f sufficiently small. It is a common practice to apply integration by parts to equation (28) in order to get rid of the second derivative term. Integration by parts uses the Gauss divergence theorem (13). The resulting weak formulation is the following one: Find $\mathbf{u} \in H^1_E$ and $p \in L_2(\Omega)$ such that:

$$\begin{cases} \nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} + \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{u})\mathbf{v} - \int_\Omega p(\nabla \cdot \mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} \text{ for all } \mathbf{v} \in H^1_{E_0}, \\ \int_\Omega q(\nabla \cdot \mathbf{u}) = 0 \text{ for all } q \in L_2(\Omega), \end{cases} \tag{30}$$

where test space $H^1_{E_0}$ and solution H^1_E defined as:

$H^1_{E_0} := \{\mathbf{v} \in H^1(\Omega) | \vec{v} = 0 \text{ on } \partial\Omega_D\}$

$H^1_E := \{\mathbf{u} \in H^1(\Omega) | \vec{u} = \vec{v} \text{ on } \partial\Omega_D\}$

and $\nabla \mathbf{u} : \nabla \mathbf{v}$ represents the component wise scalar product.

The classical result about well-posedness of (30) is known [19].

Theorem. Let $n \leq 4$ and let Ω be a bounded domain of R^n with a Lipschitz continious boundary Γ. Given a function $\mathbf{f}$ in $(H^{-1}(\Omega))^n$ and in addition $\frac{N}{\nu^2}||f||^* < 1$, where $N = \sup_{\mathbf{u},\mathbf{v},\mathbf{w} \in V} \frac{|a_1(\mathbf{w}; \mathbf{u}, \mathbf{v})|}{|\mathbf{u}|_{1,\Omega} |\mathbf{v}|_{1,\Omega} |\mathbf{w}|_{1,\Omega}}$ and $||f||^* = \sup_{\mathbf{v}V} \frac{|<\mathbf{f}, \mathbf{v}>|}{||\mathbf{v}||_{H^1_{E_0}}}$, then the Navier-Stokes equation has a unique solution $(\mathbf{u}, p)$ in $V \times L^2_0(\Omega)$. Where a_1 is the trilinear form $a_1(\mathbf{w}; \mathbf{u}, \mathbf{v}) = \Sigma^n_{i,j=1} \int_\Omega w_j \frac{\partial u_i}{\partial x_j} v_i dx$.

3 Discretization in time

We now consider discretization in time. For this, we will implement a simple Crank-Nicolson scheme for time-stepping. For linear evolution PDE's this method is known to be unconditionally stable; hence it is also thought to be a good method for some nonlinear PDE's [40]. For the fully implicit Crank-Nicolson scheme (implicit for both linear and nonlinear terms), Heywood and Rannacher [22] proved that it is almost unconditionally stable and convergent, i.e., stable and convergent when

$$\Delta t h^{-d} \leq C_0, d = 2, 3 \tag{31}$$

and Tone [33] provided the convergence proof under the condition

$$\Delta t h^{-2-d/2} \leq C_0, d = 2, 3 \tag{32}$$

for some positive constant C_0 depending on the data (Δt denotes the step size in the time direction and d is the space dimension for $\Omega \in R^d$).

3.1 Crank-Nicolson scheme

This thesis focuses on the Crank-Nicolson scheme for solving the time-dependent Navier-Stokes equations in the case of $d = 2$.

A midpoint CrankNicolson scheme for NSE reads as follows: given $\mathbf{u}^0$, $\mathbf{u}^1$, and p^1, for each k=1,2,... find velocity $\mathbf{u}^{k+1}$ and pressure p^{k+1} satisfying:

$$(\frac{\partial \mathbf{u}}{\partial t})^{k+1/2} + [\mathbf{u} \cdot \nabla \mathbf{u}]^{k+1/2} = \nu \Delta \mathbf{u}^{k+1/2} - \nabla p^{k+1/2} + \mathbf{f}^{k+1/2}, \tag{33}$$

$$\nabla \cdot \mathbf{u}^{k+1} = 0, \tag{34}$$

where $\mathbf{u}^{k+1/2} = \frac{\mathbf{u}^{k+1} + \mathbf{u}^k}{2}$.

For the computational efficiency, we want to have approximation of the convection velocity $\mathbf{u}^{k+1/2}$. A second order central scheme may the be used for the temporal derivative:

$$(\frac{\partial \mathbf{u}}{\partial t})^{k+1/2} = \frac{\mathbf{u}^{k+1} - \mathbf{u}^{k}}{\Delta t} + \mathcal{O}(\Delta t^2), \tag{35}$$

yields approximate solutions at time $t^{m+1} = t^m + \Delta t$. The final nonlinear scheme reads:

$$\begin{cases} \frac{1}{\Delta t}(\mathbf{u}^{k+1} - \mathbf{u}^{k}) + (\mathbf{u} \cdot \nabla \mathbf{u})^{k+1/2} = \nu \Delta \mathbf{u}^{k+1/2} - \nabla p^{k+1/2} + \mathbf{f}^{k+1/2} \\ \nabla \cdot \mathbf{u}^{k+1} = 0 \end{cases} \tag{36}$$

We still need to approximate the advective term using nonlinear iterations, which will be discussed in the next section.

3.2 Nonlinear iteration

There are two standard techniques to the nonlinearity in the Navier-Stokes equation:

- Picard iteration
- Newton iteration

Picard iteration is a widely used technique for solving the nonlinear equation that governs fluid dynamics. The method is simple to implement and computationally cheap but has been known to fail or converge slowly. The Newton method is more complex and expensive (on a per-iteration basis) than Picard.

3.2.1 Picard linearization

The non-linear system (28) is often solved using a Picard method. It is well known [19] that the Picard scheme is convergent when the initial approximation satisfies $\mathbf{u_0} \leq Re||\mathbf{f}||_{-1}$, where $||\cdot||_s$ defines the Sobolev norm of order s. The sufficient condition for convergence [12] is the same as sufficient condition for uniqueness of steady solution: $\nu > (c_2||\mathbf{f}||_{-1})^{1/2}/c_1$, where c_1, c_2 are continuity and coercivity constants. Picard type method (defined below) for solving the nonlinear system leads to a linear system of algebraic equations of the form:

$$\begin{bmatrix} A & C \\ C^T & 0 \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} b_n \\ 0 \end{bmatrix},$$

where x and y are vectors of coefficients for FE velocity and pressure unknowns. Matrix A is non-symmetric positive definite, and the complete matrix has block structure. Many solvers for linear systems benefit from these, and our method presented below does too. Unfortunately, the Picard method diverges when Re is not small enough. To handle the idea of higher Re numbers, we use the following idea from [27]. We discretized equation by using a coarse mesh and then applying a few Picard type iteration steps, as a defect correction.

We are solving the Navier-Stokes equation with a few Picard iterations which leads to Oseen problem on each iteration. Thus, given an "initial guess" $(\mathbf{u_0}, p_0) \in (H^1_E \times L_2(\Omega))$, a sequence of iterates is computed $(\mathbf{u_i}, p_i) \in (H^1_E \times L_2(\Omega))$, $i = 0, .., n$ The system (23) is often solved in practical calculations using the method, given as follows.
We start computations with the nonlinear residual associated with the weak formulation (30). This is a pair of functionals $R_k(\mathbf{v})$, $r_k(q)$ satisfying:

$$R_k = \int_\Omega \mathbf{f} \cdot \mathbf{v} - \int_\Omega \mathbf{u_k} \cdot \nabla \mathbf{u_k} \cdot \mathbf{v} - \nu \int_\Omega \nabla \mathbf{u_k} : \nabla \mathbf{v} + \int_\Omega p_k (\nabla \cdot \mathbf{v}), \tag{37}$$

$$r_k = -\int_\Omega q(\nabla \cdot \mathbf{u_k}) \tag{38}$$

We have the following linear problem. For all $\mathbf{v} \in H^1_{E_0}$ and $q \in L_2(\Omega)$, find $\delta \mathbf{u_k} \in H^1_{E_0}$ and $\delta p_k \in L_2(\Omega)$ satisfying:

$$\begin{cases} \int_\Omega \mathbf{u_k} \cdot \nabla \delta \mathbf{u_k} \cdot \mathbf{v} + \nu \int_\Omega \nabla \delta \mathbf{u_k} : \nabla \mathbf{v} - \int_\Omega \delta p_k (\nabla \cdot \mathbf{v}) = R_k(\mathbf{v}) \\ \int_\Omega q(\nabla \cdot \delta \mathbf{u_k}) = r_k(q) \end{cases} \tag{39}$$

The solution of (39) is the Picard correction. Updating the previous iterate using $\mathbf{u_{k+1}} = \mathbf{u_k} + \delta \mathbf{u_k}$, $p_{k+1} = p_k + \delta p_k$ defines the next iterate in the sequence.

If we substitute $\delta \mathbf{u_k} = \mathbf{u_{k+1}} - \mathbf{u_k}$ and $\delta p_k = p_k - p_{k+1}$ into (39) then we obtain an explicit definition for the new iterate: for all $\mathbf{v} \in H^1_{E_0}$ and $q \in L_2(\Omega)$ find $\mathbf{u_{k+1}} \in H^1_{E_0}$ and $p_{k+1} \in L_2(\Omega)$ such that:

$$\int_\Omega \mathbf{u_k} \cdot \nabla \mathbf{u_{k+1}} \cdot \mathbf{v} + \nu \int_\Omega \nabla \mathbf{u_{k+1}} : \nabla \mathbf{v} - \int_\Omega p_{k+1} (\nabla \cdot \mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v}, \tag{40}$$

$$\int_\Omega q(\nabla \cdot \mathbf{u_{k+1}}) = 0; \tag{41}$$

We see that the Picard iteration corresponds to a simple fixed-point iteration strategy for solving system (28), with the part of the convection coefficient evaluated at the current velocity.

3.2.2 Newton linearizaton

Another standard method of solving nonlinear system is the Newton's method, which followed by solving linear system on each step. Unfortunately, for higher Re it makes some difficulties for multilevel solvers.

The corrections $\delta \mathbf{u_k} \in \mathbf{H}^1_{E_0}$ and $\delta p_k \in \mathbf{L}_2(\Omega)$ should satisfy

$$\int_\Omega (\delta \mathbf{u_k} + \mathbf{u_k}) \cdot \nabla (\mathbf{u_k} + \mathbf{u_k}) \cdot \mathbf{v} - \int_\Omega (\mathbf{u_k} \cdot \nabla \mathbf{u_k}) \cdot \mathbf{v} + \nu \int_\Omega \nabla \delta \mathbf{u_k} : \nabla \mathbf{v} - \int_\Omega \delta p_k (\nabla \cdot \mathbf{v}) = R_k(\mathbf{v}) \tag{42}$$

$$\int_{\Omega} q(\nabla \cdot \delta \mathbf{u_k}) = r_k(q) \tag{43}$$

The solution of (42) is the Newton correction. So we are updating the previous iteration by formulas $\mathbf{u_{k+1}} = \mathbf{u_k} + \delta \mathbf{u_k}$ and $p_{k+1} = p_k + \delta p_k$, which define next iterate.

The Newton iteration method converges faster but within a smaller radius of convergence than the Picard technique. It may be preferable to apply the Newton technique when a close guess to the solution is available. For certain problems, a mixed or joint approach may be considered in which the Picard iteration method is used to improve the initial solution guess before the algorithm switches to the Newton iteration.

4 Finite element approximation

4.1 The Galerkin finite element method

The goal in this chapter is the setup of discretization method, where piecewise polynomial $\mathbf{u_h}$ approximates a vector function $\mathbf{u}$: $\Omega \to R$.

We approximate $\mathbf{u}$ using Galerkin discretization on a finite dimensional space H_E^{1}. We split the given domain Ω into subdomains of the same shape on which we define finite elements functions. The discrete formulation of (23) is: find $\mathbf{u_h} \in X_E^h$ and $p_h \in M^h$ such that

$$\begin{aligned} \nu \int_\Omega \nabla \mathbf{u_h} : \nabla \mathbf{v_h} + \int_\Omega (\mathbf{u_h} \cdot \nabla \mathbf{u_h}) \cdot \mathbf{v_h} - \int_\Omega p_h (\nabla \cdot \mathbf{v_h}) = \int_\Omega \mathbf{f} \cdot \mathbf{v_h} \text{ for all } \mathbf{v_h} \in X_0^h \\ \int_\Omega q_h (\nabla \cdot \mathbf{u_h}) = 0 \text{ for all } q_h \in M^h, \end{aligned} \tag{44}$$

where $\mathbf{u_h} : \nabla \mathbf{v_h}$ is the componentwise scalar product. To construct an approximation, we assume that Finite Element velocity space $S_0^h \subset H_{E_0}^1$ is an n-dimensional vector space of test functions for which $\{\phi_1, ..., \phi_n\}$ is a basis. To satisfy boundary conditions, we extend this set by additional functions $\phi_{n+1}, ..., \phi_{n+n\partial}$ and select fixed coefficients u_j, so that the function $\Sigma u_j \phi_j$ interpolates the boundary data,

$$\mathbf{u_h} = \Sigma_{j=1}^n u_j \phi_j + \Sigma_{n+1}^{n+n\partial} u_j \phi_j \tag{45}$$

We also construct a set of scalar (pressure) basis functions ψ_k, which is a basis for $M_h \subset L_2(\Omega)$, and set

$$p_h = \Sigma_{k=1}^{n_p} p_k \psi_k, \tag{46}$$

We define matrix A, the vector-Laplacian matrix, and the matrix B, the divergence matrix, the vector-convection matrix N and the additional matrix W resulting from the Newton derivative as follows:

$$A = [a_{ij}],\ a_{ij} = \int_\Omega \nabla \phi_{\mathbf{i}} : \phi_{\mathbf{j}} \tag{47}$$

$$B = [b_{ij}],\ b_{ij} = -\int_\Omega \psi_k \nabla \cdot \phi_{\mathbf{j}} \tag{48}$$

$$N = [n_{ij}],\ n_{ij} = \int_\Omega (\mathbf{u_h} \cdot \nabla \phi_{\mathbf{j}}) \cdot \phi_{\mathbf{i}} \tag{49}$$

$$W = [w_{ij}],\ w_{ij} = \int_\Omega (\phi_{\mathbf{j}} \cdot \nabla \mathbf{u_h}) \cdot \phi_{\mathbf{i}} \tag{50}$$

So the matrix formulation of Navier-Stokes equation is the following:

$$\begin{pmatrix} \nu A + N + W & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{51}$$

Where matrix A is symmetric and positive definite and Newton derivative matrix N is skew-symmetric if $\mathbf{u_h}$ is point-wise divergence free and $\mathbf{u_h} \cdot \mathbf{n}$=0 on the bondary. Matrix B has full rank for the case of open boundary conditions and 1-rank deficient for enclosed flows. Matrix W is sign indefinite. The right-hand-side vectors are the nonlinear residuals associated with the current discrete solutions.

$$f = [f_i], f_i = \int_\Omega \mathbf{f_i} \cdot \phi_{\mathbf{i}} - \int_\Omega \mathbf{u_h} \cdot \nabla \mathbf{u_h} \cdot \phi_{\mathbf{i}} - \nu \int_\Omega \nabla \mathbf{u_h} : \nabla \phi_{\mathbf{i}} + \int_\Omega p_h (\nabla \cdot \phi_{\mathbf{i}}) \tag{52}$$

$$g = [g_k], g_k = \int_\Omega \psi_k (\nabla \cdot \vec{u_h}) \tag{53}$$

For Picard iteration (51) changes to:

$$\begin{pmatrix} \nu A + N & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{54}$$

The mathematical motivation for finite element approximation is the observation that a smooth function can often be approximated to arbitrary accuracy using piecewise polynomial functions. The idea is to choose basis functions ϕ_j that are locally nonzero on a mesh of triangles or tetrahedra or a grid of rectangles. We will discuss some types of rectangular elements below.

4.2 Stable rectangular elements

The discrete analogue of (30) (refereed to as uniform inf-sup stability) requires the existence of a positive constant γ independent of h such that, for any grid,

$$\inf_{q_h \neq 0} \sup_{\vec{v_h} \neq 0} \frac{(q_h, \nabla \cdot \vec{v_h})}{||\vec{v_h}||_{1,\Omega} ||q_h||_{0,\Omega}} \geq \gamma > 0, \tag{55}$$

Where $v_h \in S_0^h$, $q_h \in M^h$ and $||\mathbf{v}||_{1,\Omega} = (\int_\Omega \mathbf{v} \cdot \mathbf{v} + \nabla \mathbf{v} : \nabla \mathbf{v})^{1/2}$ is a norm for functions in $H^1_{E_0}$, $||q||_{0,\Omega} = ||q - (1/|\Omega|) \int_\Omega q||$ is a quotient space norm.

This condition guarantees that the solvability condition holds for any admissible grid, and it is also crucial for establishing optimal a priori error bounds. Stable rectangular elements $Q_2 - Q_1$ are known as the Taylor-Hood method, where Q_1 is bilinear element. On a rectangular grid each function from Q_1 is of the form $(ax + b)(cy + d)$. Q_2 is a biquadratic finite element approximation on a rectangular grid. Its nodal degrees of freedom consist of vertices, four midside additional node points and centroid. The resulting approximation on each element is from $span\{1, x, y, xy, x^2, y^2, xy^2, x^2y, x^2y^2\}$. In this case, our linear system still has the same form as in (92). It is well-known that these elements satisfy the discrete inf-sup condition (55) and that there is a unique numerical solution .

4.3 Inf-sup unstable rectangular elements.

If $B^T p = 0$ for some nonzero pressure mode p, then $\gamma = 0$ in (92) and so the finite element method is unstable. Such p corresponds to some spurious pressure mode different from 1. Unstable FEM can be stabilized to make it practical. The basic idea behind stabilization is to relax the incompressibility constraint in a special way so that spurious pressure modes are no longer in a null space of the resulting coefficient matrix. Stabilized discrete formulation of (23) is: find $\mathbf{u_h} \in X_E^h$

and $p_h \in M^h$ such that

$$\begin{aligned} \nu \int_\Omega \nabla \mathbf{u_h} : \nabla \mathbf{v_h} + \int_\Omega (\mathbf{u_h} \cdot \nabla \mathbf{u_h}) \cdot \mathbf{v_h} - \int_\Omega p_h (\nabla \cdot \mathbf{v_h}) = \int_\Omega \mathbf{f} \cdot \mathbf{v_h} \text{ for all } \mathbf{v_h} \in X_0^h \\ \int_\Omega q_h (\nabla \cdot \mathbf{u_h}) + \beta \int_\Omega \nabla q_h \cdot \nabla q_h = 0 \text{ for all } q_h \in M^h, \end{aligned} \tag{56}$$

As a result of stabilization discrete solutions $\mathbf{u_h}$ and p_h satisfy rigorous error bounds. From a linear algebra viewpoint, the stabilization provides a consistent regularization of the singular matrix that arises in the unstabilized case. It introduces a stabilization parameter $\beta > 0$ together with stabilization matrix C in place of a zero block in the discrete system (suggesting that C should be a symmetric positive-semidefinite matrix):

$$\begin{pmatrix} \nu A + N & B^T \\ B & -\beta C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{57}$$

with the following stability condition:

$$null(S_\beta) = \text{span}\{1\}, \tag{58}$$

where $S_\beta = BA^{-1}B^T + \beta C$ is a symmetric positive-semidefinite matrix.

To define stabilization for our problem, we will use $\beta = \frac{1}{\nu}$.

A finite element spatial discretization of the system results in large, sparse systems of the form,

$$\mathcal{A}x = b \Longleftrightarrow \begin{pmatrix} \nu A + N & B^T \\ B & -\frac{1}{\nu} C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{59}$$

where u and p represent the discrete velocity and pressure, arising from the Picard iteration applied to the Navier-Stokes equations. The coefficient matrices are non-symmetric. The Krylov subspace methods can be used to solve these systems. For our case we choose GMRES. Thus, we can proceed directly to the design of preconditioning.

5 Preconditioners

The convergence rate of iterative methods depends on spectral properties of the coefficient matrix. Hence one may attempt to transform the linear system into one that is equivalent in the sense that it has the same solution, but has more favorable spectral properties. A preconditioner is a matrix that defines such a transformation.

For instance, if a matrix M approximates the coefficient matrix A in some way, the transformed system can be written in the following form:

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}. \tag{60}$$

It has the same solution as the original system $A\mathbf{x} = \mathbf{b}$, but the spectral properties of its coefficient matrix may be more favorable. It is a useful approach only if computing $M^{-1}\mathbf{v}$ for an arbitrary vector $\mathbf{v}$ is cheap. Such a matrix M is called a preconditioner or, more precisely, a left preconditioner.

In the case of a right preconditioner, one solves:

$$AM^{-1}\mathbf{u} = \mathbf{b}, \text{ where } \mathbf{x} = M^{-1}\mathbf{u} \tag{61}$$

5.1 ILU factorization

In numerical analysis, LU decomposition is the factorization of a matrix as the product of a lower triangular matrix and an upper triangular matrix. An incomplete factorization instead generates triangular matrices L, U such that $A \approx LU$ rather than $A = LU$. Solving for $LUx = b$ can be done quickly but does not yield the exact solution to $Ax = b$. So, we instead use the matrix $M = LU$ as a preconditioner in iterative solvers.

To introduce the basic implementation of incomplete factorization, we will first desribe the $ILU(0)$ factorization as in [35]. It is the technique with no fill-in as a modified version of Gaussian elimination. The main idea is to drop elements of L and U which are outside the sparsity pattern of A. The sparsity pattern of A is defined as follows

$$P \subset \{(i,j) | i \neq j; 1 \leq i,j \leq n\}, \tag{62}$$

where n is a dimension of matrix A. By the definition, L and U together have the same number of non-zero elements as matrix A. So, in general, $ILU(p)$ preconditioner is thus defined as a factorization that retains all elements with level up to p. A level of fill-in is attributed to each element that is processed by Gaussian elimination, and dropping will be based on the value of the level of fill-in [35].The initial level of fill-in for element a_{ij} is defined by the formula

$$lev(i,j) = \begin{cases} 1, & \text{if } a_{ij} \neq 0 \text{ or } i = j. \\ \infty, & \text{otherwise.} \end{cases} \tag{63}$$

Updated level of fill-in we compute by the formula

$$min\{lev(i,j), lev(i,k) + lev(k,j) + 1\} \tag{64}$$

There is also a modification of this algorithm. Instead of using fill-in criteria, we can use ILU(τ) algorithm with threshold. We are applying a dropping rule to an element, which means we are replacing the element by zero if it satisfies a set of criteria. A dropping rule can be applied to a whole row by applying the same rule to all elements of the row.

Incomplete LU factorizations of A can be written in the form $A = LU - E$ with an error matrix E. How small the norm of the matrix E is ruled by a threshold parameter $\tau > 0$. For positive definite matrices A one can choose such a small τ that the product LU of its incomplete triangular factors L and U is also positive definite and so estimates from [20] can be applied to assess the numerical stability of the incomplete factorization. Let $c_A = \lambda_{\min}(A_{\mathrm{S}})$, where $A_{\mathrm{S}} = \frac{1}{2}(A + A^T)$. The sufficient condition for numerical stability is $\tau < c_A n^{-1}$ ($n = dim(A)$). In practice, however,

larger τ are used, and in the case of non-symmetric matrices non-positive or close to zero pivots may appear, and breakdown of an algorithm may happen. Now let us take a look at the situation with ILU factorization for saddle-point matrices with positive definite (1,1)-block, as in the given problem. It was observed that for symmetric saddle-point systems that the block factorization can be used to construct an incomplete factorization.
To ameliorate the performance of the preconditioning in some situations, we consider the two-parameter Tismenetsky-Kaporin variant of the threshold ILU factorization. Below we consider an extension of the Tismenetsky-Kaporin factorization to the case of non-symmetric and saddle-point matrices.

Given a matrix $A \in R^{n\times n}$, the two-parameter factorization can be written as

$$A = LU + LR_u + R_\ell U - E, \tag{65}$$

(see [26]), where R_u and R_ℓ are strictly upper and lower triangular matrices, while U and L are upper and lower triangular matrices, respectively. Given two small parameters $0 < \tau_2 \le \tau_1$ the off-diagonal elements of U and L are either zero or have absolute values greater than τ_1, the absolute values of R_ℓ and R_u entries are either zero or belong to $(\tau_2, \tau_1]$; entries of the error matrix are of order $O(\tau_2)$. We say that it is the $ILU(\tau_1,\ \tau_2)$ factorization of A. The well-known one parameter $ILU(\tau)$ factorization can be viewed as (65) with $R_u = R_\ell = 0$ and $\tau_1 = \tau_2 = \tau$. The important improvement the two-parameter ILU factorization gives over the one-parameter $ILU(\tau)$ is that the fill-in of L and U is ruled by the first threshold parameter τ_1, while the quality of the resulting preconditioner is mainly defined by τ_2, once $\tau_1^2 < \tau_2$ holds [25]. It means that if we choose $\tau_2 = \tau_1^2 = \tau$, then the level of fill-in of $ILU(\tau_1, \tau_2)$ may be similar to $ILU(\tau)$ but the rate of convergence of preconditioned Krylov subspace method will be comparable to $ILU(\tau^2)$.

In what follows, the algorithm makes no specific use of the block structure of the matrix A, but can formally be applied to a generic nonsymmetric $A \in R^{n\times n}$. Thus, we denote by A below some given nonsymmetric square matrix.

The derivation of the $ILU(\tau_1, \tau_2)$ preconditioner in the SPD case assumes such a scaling of the matrix and unknowns that all diagonal elements are equal to 1. For the performance of the method, it was found [26], that it is very important to rescale a given matrix A (see below). We can use the condition number to judge whether a given non-singular matrix A is ill-conditioned or not, which is defined as:

$$Cond(A) = ||A||||A^{-1}||, \tag{66}$$

where the matrix norm is the Frobenius norm. At the same time the value of Cond(A) measures the sensitivity of the solution x to numerical perturbation of a matrix and the right-hand side of (4.3). The property $Cond(\alpha A) = Cond(A)$ for every scalar $\alpha \neq 0$ shows that it is impossible to reduce $Cond(A)$ by multiplying all equations by a scalar. However, it is possible to diminish $Cond(A)$ by multiplying every row and every column of the matrix A by suitable scaling factor l_i and r_i, respectively. The corresponding vectors are denoted by $l \in R^n$ and $r \in R^n$.

In this project, we propose a simple procedure to find matrices $diag(l)$ and $diag(r)$ only through a few operations, which are given explicitly. Thus, we look for scaling vectors $l, r \in R^n$ such that the matrix $A' = diag(l) \cdot A \cdot diag(r)$ has nearly balanced Euclidean norms of rows and columns. For this purpose, we will apply the Sinkhorn algorithm to the nonnegative matrix $F = [a_{kj}^2]$, $k, j = 1 \dots n$. One iteration of the algorithm is as follows:

$$\begin{aligned} diag(r^{(k+1)}) &= diag(F^T l^{(k)})^{-1}, \\ diag(l^{(k+1)}) &= diag(F r^{(k+1)})^{-1}. \end{aligned} \tag{67}$$

For the starting vector we take $l^{(0)} = \{1...1\}$. In all experiments in the next sections, we performed five iterations of (67) to find the scaling vectors, before any incomplete factorization is computed. If factorization is computed for the scaled matrix A' so that $L'U' \approx A'$, then we have to rescale the triangular factors for the original matrix:

$$LU \approx A, \; L = (diag(l))^{-1} L', \; U = U'(diag(r))^{-1} \tag{68}$$

5.2 The $ILU(\tau_1, \tau_2)$ algorithm pseudo-code

Assume we are given a sparse matrix A whose elements are a_{ij}, left and right scaling diagonal matrices $D_L = diag(l)$ and $D_R = diag(r)$, threshold parameters $0 < \tau_2 \leq \tau_1 < 1$. For a matrix A, P(A) denotes the subset of indexes (i,j) such that $a_{ij} = 0$. We will use below the notation R for the upper triangular error factor R_u (factor R_l is not computed in the course of factorization) and $\mathbf{v} \in R^n$ for an auxiliary vector (initially $\mathbf{v} = \{0, \ldots, 0\}$).

```
(1) Main loop by rows of $A$ to compute the rows of $L$ and $U$:
    for $i = 1, \ldots, n$:
(2) Initialize the row accumulator vector $v$ by the $i$th row of the balanced matrix $A$:
        for $j = 1, \ldots, n$ and if $(i, j) \notin P(A)$:
            $v_j := (D_L)_{ii} a_{ij} (D_R)_{jj}$
        end for
(3) Loop over all already computed rows of $U$:
        for $k = 1, \ldots, i - 1$ and if $v_k \neq 0$:
(4) Update the accumulator vector:
            $v_k := v_k / U_{kk}$
            if $|v_k| > \tau_2$ then
                for $j = k + 1, \ldots, n$ and if $(k, j) \notin P(U)$:
                    $v_j := v_j - v_k U_{kj}$
                end for
            end if
            if $|v_k| > \tau_1$ then
                for $j = k + 1, \ldots, n$ and if $(k, j) \notin P(R)$:
                    $v_j := v_j - v_k R_{kj}$
                end for
            end if
        end for
(5) Rescale the $i$th row of $U$:
        $\lambda_i := \max_{k=i,\ldots,n} |v_k|$
        if $\lambda_i < \tau_2$ then
            $\lambda_i := \tau_2$
        end if
        for $j = i, \ldots, n$ and if $v_j \neq 0$:
            $v_j := v_j / \lambda_i$
        end for
(6) Compute the $i$th row of $L$:
        $L_{ii} = \lambda_i$
        for $j = 1, \ldots, i-1$ and if $|v_j| > \tau_1$:
            $L_{ij} := v_j$
        end for
(7) Compute the $i$th row of $U$ and $R$:
        if $|v_i| < \tau_2$ then
            $v_i := \tau_2 \cdot \mathrm{sign}(v_i)$
        end if
        $U_{ii} = v_i$
        for $j = i+1, \ldots, n$ and if $v_j \neq 0$:
            if $|v_j| > \tau_1$ then
                $U_{ij} := v_j$
            else if $|v_j| > \tau_2$ then
                $R_{ij} := v_j$
            end if
        end for
(8) Clear nonzero elements of the row accumulator $v$:
        for $j = 1, \ldots, n$ and if $v_j \neq 0$:
            $v_j := 0$
        end for
    end for
(9) Perform the final re-scaling of the incomplete factors $L$ and $U$:
```

```
for $i = 1, \dots, n$:
    for $j = 1, \dots, i$ and if $(i,j) \notin P(L)$:
        $L_{ij} := L_{ij}/(D_L)_{ii}$
    end for
    for $j = i, \dots, n$ and if $(i,j) \notin P(U)$:
        $U_{ij} := U_{ij}/(D_R)_{jj}$
    end for
end for
```

5.3 LU factorization

The 2×2-block matrix from (59) is in general not sign definite and if $C = 0$, its diagonal has zero entries. A potential source of instabilities in (4.3) is the presence of dominating convection terms. This necessitates stabilization of the discrete system, if the mesh is not sufficiently fine to resolve all scales in the solution. The stabilization leads to the modification of (1, 2)-block of the matrix. It follows that (1, 2)-block is not equal to the transpose of the (2, 1)-block B. The algebraic framework of this section admits a generic positive semi-definite matrix C. An LU factorization of such matrices often requires pivoting (rows and columns permutations) for stability reasons. However, exploiting the block structure and the properties of blocks A and C, one readily checks that the LU factorization

$$\mathcal{A} = \begin{pmatrix} A & \widetilde{B}^T \\ B & -C \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & -U_{22} \end{pmatrix} \tag{69}$$

with lower (upper) triangle matrices L_{11}, L_{22} (U_{11}, U_{22}) exists without pivoting, once $\det(A) \neq 0$ and there exist LU factorizations for the (1,1)-block

$$A = L_{11}U_{11}$$

and the Schur complement matrix $\widetilde{S} := BA^{-1}\widetilde{B}^T + C$ is factorized as

$$\widetilde{S} = L_{22}U_{22}.$$

Decomposition (69) then holds with $U_{12} = L_{11}^{-1}B^T$ and $L_{21} = BU_{11}^{-1}$.

Assume A is positive definite. Then the LU factorization of A exists without pivoting. Its numerical stability (the relative size of entries in factors L_{11} and U_{11}) may depend on how large is the skew-symmetric part of A comparing to the symmetric part. To make this statement more precise, we denote $A_{\mathrm{S}} = \frac{1}{2}(A+A^T)$, $A_{\mathrm{N}} = A - A_{\mathrm{S}}$ (similar notation will be used to denote symmetric and skew-symmetric parts of other matrices) and let

$$C_A = \|A_{\mathrm{S}}^{-\frac{1}{2}}A_{\mathrm{N}}A_{\mathrm{S}}^{-\frac{1}{2}}\|.$$

Here $|M|$ denotes the matrix of absolute values of M-entries. The following bound on the size of elements of L_{11} and U_{11} holds (see eq.(3.2) in [26]):

$$\frac{\||L_{11}||U_{11}|\|_F}{\|A\|} \leq n\left(1 + C_A^2\right). \tag{70}$$

If $C \geq 0$, $\widetilde{B} = B$, and matrix B^T has the full column rank, then the positive definiteness of A implies that the Schur complement matrix is also positive definite. However, this is not the case for a general block $\widetilde{B} \neq B$. In the application studied in [25], block $\widetilde{B}^T$ is a perturbation of B^T. The analysis below shows that the positive definiteness of $\widetilde{S}$ and the stability of its LU factorization is guaranteed if the perturbation $E = \widetilde{B} - B$ is not too large. The size of the perturbation will enter our bounds as the parameter ε_E defined as

$$\varepsilon_E := \|A_{\mathrm{S}}^{-\frac{1}{2}}E^T\|.$$

For the ease of analysis we introduce further notations:

$$S = BA^{-1}B^T + C, \quad \widehat{A}_N = A_{\mathrm{S}}^{-\frac{1}{2}} A_{\mathrm{N}}\, A_{\mathrm{S}}^{-\frac{1}{2}}.$$

We shall repeatedly make use of the following identities:

$$\begin{aligned}
(A^{-1})_{\mathrm{S}} &= \frac{1}{2}\left(A^{-1} + A^{-T}\right) = A_{\mathrm{S}}^{-\frac{1}{2}}(I - \widehat{A}_N^2)^{-1} A_{\mathrm{S}}^{-\frac{1}{2}}, \\
(A^{-1})_{\mathrm{N}} &= \frac{1}{2}\left(A^{-1} - A^{-T}\right) = A_{\mathrm{S}}^{-\frac{1}{2}}(I + \widehat{A}_N)^{-1}\widehat{A}_N(I - \widehat{A}_N)^{-1} A_{\mathrm{S}}^{-\frac{1}{2}}.
\end{aligned} \tag{71}$$

From the identities

$$\langle (Sq, q)\rangle = \langle (Bv, q)\rangle + \langle (Cq, q)\rangle = \langle (v, B^T q)\rangle + \langle (Cq, q)\rangle = \langle (Av, v)\rangle + \langle (Cq, q)\rangle,$$

which are true for $q \in R^m$ and $v := A^{-1}B^T q \in R^n$, we see that S is positive definite, if A is positive definite. For $\widetilde{S}$ we then compute:

$$\begin{aligned}
\langle \widetilde{S}q, q\rangle &= \langle Sq, q\rangle + \langle A^{-1}E^T q, B^T q\rangle \\
&= \langle Sq, q\rangle + \langle A_{\mathrm{S}}^{\frac{1}{2}} A^{-1} E^T q, A_{\mathrm{S}}^{-\frac{1}{2}} B^T q\rangle \\
&= \langle Sq, q\rangle + \langle A_{\mathrm{S}}^{\frac{1}{2}} A^{-1} E^T q, (I - \widehat{A}_N)(I - \widehat{A}_N)^{-1} A_{\mathrm{S}}^{-\frac{1}{2}} B^T q\rangle \\
&= \langle Sq, q\rangle + \langle \left((I + \widehat{A}_N) A_{\mathrm{S}}^{\frac{1}{2}} A^{-1} A_{\mathrm{S}}^{\frac{1}{2}}\right) A_{\mathrm{S}}^{-\frac{1}{2}} E^T q, (I - \widehat{A}_N)^{-1} A_{\mathrm{S}}^{-\frac{1}{2}} B^T q\rangle.
\end{aligned}$$

We employ identities (71) to get

$$\begin{aligned}
(I + \widehat{A}_N) A_{\mathrm{S}}^{\frac{1}{2}} A^{-1} A_{\mathrm{S}}^{\frac{1}{2}} &= (I + \widehat{A}_N) A_{\mathrm{S}}^{\frac{1}{2}}((A^{-1})_{\mathrm{S}} + (A^{-1})_{\mathrm{N}}) A_{\mathrm{S}}^{\frac{1}{2}} \\
&= (I + \widehat{A}_N)((I - \widehat{A}_N^2)^{-1} + (I + \widehat{A}_N)^{-1}\widehat{A}_N(I - \widehat{A}_N)^{-1}) \\
&= (I - \widehat{A}_N)^{-1} + \widehat{A}_N(I - \widehat{A}_N)^{-1} \\
&= (I + \widehat{A}_N)(I - \widehat{A}_N)^{-1}.
\end{aligned}$$

Noting $\|(I-\widehat{A}_N)^{-1}\| \le 1$ for a skew-symmetric $\widehat{A}_N$, it was estimated that

$$\begin{aligned}
\langle \widetilde{S}q, q\rangle &\ge \langle Sq,q\rangle - \|(I+\widehat{A}_N)(I-\widehat{A}_N)^{-1}\|\|A_{\mathrm{S}}^{-\frac{1}{2}}E^Tq\|\|(I-\widehat{A}_N)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq\| \\
&\ge \langle Sq,q\rangle - \|(I+\widehat{A}_N)\|\|A_{\mathrm{S}}^{-\frac{1}{2}}E^T\|\|q\|\|(I-\widehat{A}_N)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq\| \\
&\ge \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\|(I-\widehat{A}_N)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq\| \\
&= \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\langle (I-\widehat{A}_N)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq, (I-\widehat{A}_N)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq\rangle^{\frac{1}{2}} \\
&= \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\langle A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq, (I+\widehat{A}_N)^{-1}(I-\widehat{A}_N)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq\rangle^{\frac{1}{2}} \\
&= \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\langle A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq, (I-\widehat{A}_N^2)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq\rangle^{\frac{1}{2}} \qquad (72)\\
&= \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\langle B^Tq, A_{\mathrm{S}}^{-\frac{1}{2}}(I-\widehat{A}_N^2)^{-1}A_{\mathrm{S}}^{-\frac{1}{2}}B^Tq\rangle^{\frac{1}{2}} \\
&= \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\langle B(A^{-1})_{\mathrm{S}}B^Tq, q\rangle^{\frac{1}{2}} \\
&= \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\langle BA^{-1}B^Tq, q\rangle^{\frac{1}{2}} \\
&= \langle Sq,q\rangle - (1+C_A)\varepsilon_E\|q\|\langle Sq, q\rangle^{\frac{1}{2}} \\
&\ge \left(1-(1+C_A)\varepsilon_E\lambda_{\min}^{-\frac{1}{2}}(S_{\mathrm{S}})\right)\langle Sq,q\rangle.
\end{aligned}$$

Hence, we conclude that $\widetilde{S}$ is positive definite then it holds

$$\kappa := (1+C_A)\varepsilon_E c_S^{-\frac{1}{2}} < 1 \qquad (73)$$

where $c_S := \lambda_{\min}(S_{\mathrm{S}})$.

If $\widetilde{S}$ is positive definite, the factorization $\widetilde{S} = L_{22}U_{22}$ satisfies the stability bound similar to (70):

$$\frac{\||L_{22}||U_{22}|\|_F}{\|\widetilde{S}\|} \le m\left(1+\|\widetilde{S}_{\mathrm{S}}^{-\frac{1}{2}}\widetilde{S}_{\mathrm{N}}\widetilde{S}_{\mathrm{S}}^{-\frac{1}{2}}\|^2\right),$$

where $\widetilde{S}_{\mathrm{S}} = \frac{1}{2}(\widetilde{S}+\widetilde{S}^T)$, $\widetilde{S}_{\mathrm{N}} = \widetilde{S}-\widetilde{S}_{\mathrm{S}}$.

The quotients $C_A = \|A_{\mathrm{S}}^{-\frac{1}{2}}A_{\mathrm{N}}A_{\mathrm{S}}^{-\frac{1}{2}}\|$ and $\|\widetilde{S}_{\mathrm{S}}^{-\frac{1}{2}}\widetilde{S}_N\widetilde{S}_{\mathrm{S}}^{-\frac{1}{2}}\|$ are largely responsible for the stability of the LU factorization for (59). The following lemma from [25] shows the estimate of $\|\widetilde{S}_{\mathrm{S}}^{-\frac{1}{2}}\widetilde{S}_N\widetilde{S}_{\mathrm{S}}^{-\frac{1}{2}}\|$ in terms of C_A, ε_E and c_S.

Let $A \in R^{n\times n}$ be positive definite and (73) be satisfied, then it holds

$$\|\widetilde{S}_{\rm S}^{-\frac{1}{2}}\widetilde{S}_N\widetilde{S}_{\rm S}^{-\frac{1}{2}}\| \le \frac{(1+\varepsilon_E c_S^{-\frac{1}{2}})C_A}{1-\kappa}. \tag{74}$$

To estimate the entries of U_{12} and L_{21} factors in (69) we repeat the arguments from [26] and arrive at the following bound

$$\frac{\|U_{12}\|_F+\|L_{21}\|_F}{\|U_{11}\|\|\widetilde{B}\|_F+\|L_{11}\|\|B\|_F} \le \frac{m(1+C_A)}{c_A}$$

with $c_A := \lambda_{\min}(A_{\rm S})$.

Summary of the results of this section is in the following theorem.

Assume matrix A is positive definite, C is positive semidefinite, and the inequality (73) holds with $\varepsilon_E = \|A_{\rm S}^{-\frac{1}{2}}(\widetilde{B}-B)^T\|$, $C_A = \|A_{\rm S}^{-\frac{1}{2}}A_{\rm N}A_{\rm S}^{-\frac{1}{2}}\|$, and $c_S = \lambda_{\min}(S_{\rm S})$, then the LU factorization for (69) exists without pivoting. The entries of the block factors satisfy the following bounds

$$\begin{aligned}
\frac{\||L_{11}||U_{11}|\|_F}{\|A\|} &\le n\left(1+C_A^2\right),\\
\frac{\||L_{22}||U_{22}|\|_F}{\|\widetilde{S}\|} &\le m\left(1+\frac{(1+\varepsilon_E c_S^{-\frac{1}{2}})C_A}{1-\kappa}\right),\\
\frac{\|U_{12}\|_F+\|L_{21}\|_F}{\|U_{11}\|\|\widetilde{B}\|_F+\|L_{11}\|\|B\|_F} &\le \frac{m(1+C_A)}{c_A}
\end{aligned}$$

with κ from (73).

The above analysis indicates that the LU factorization for (59) exists if the (1,1) block A is positive definite and the perturbation of the (1,2)-block is sufficiently small. The stability bounds depend on the constant C_A which measures the ratio of skew-symmetry for A, the ellipticity constant c_A, the perturbation measure ε_E and the minimal eigenvalue of the symmetric part of the unperturbed Schur complement matrix S.

5.4 PCD and LSC preconditioners

We consider another two successful approaches for approximating the Schur complement: the pressure convection-diffusion (PCD) preconditioner [12] and the least-squares commutator (LSC) preconditioner [12]. Here we briefly outline the key ideas and provide basic forms of the preconditioners.

5.4.1 The pressure convection-diffusion preconditioner

Pressure convection-diffusion preconditioner in [11] appears to be robust with respect to mesh refinement and more robust with respect to kinematic viscosity ν than the simply scaled mass matrix preconditioner. Some deterioration of its performance happens when $\nu \to 0$ for many problems and discretizations. The approach to building the PCD preconditioner is often motivated by considering the commutator ε between the divergence operator and the convection-diffusion operator.

Let

$$\mathcal{F} = -\nabla \cdot (2\mu \mathbf{D}) + \rho \mathbf{w_h} \cdot \nabla \tag{75}$$

Suppose that we have an analogous operator to $\mathcal{F}$ on the pressure space, denoted by $\mathcal{F}_p$, so that

$$\varepsilon = \nabla \cdot \mathcal{F} - \mathcal{F}_p \nabla. \tag{76}$$

The idea is then to use a discrete version of ε, and deduce the Schur complement approximation by setting the discrete commutator to 0. Upon equating the discrete commutator to be zero, one obtains an approximation to S in terms of finite element matrices. To correctly scale the discrete operators in the discrete commutator, one uses the finite element mass matrices:

$$Q_{i,j} = \int_{\Omega} \phi_j \cdot \phi_i. \tag{77}$$

The discrete commutator is then defined by

$$\varepsilon_h = (Q^{-1}B)(Q^{-1}A) - (Q^{-1}F_p)(Q^{-1}B), \tag{78}$$

where F_p represents the discrete counterpart of $\mathcal{F}_p$. A and B are matrices arising from the discretization of the Navier-Stokes equation. After letting ε_h to be zero, this can be rearranged to give the approximation

$$BA^{-1}B^T \approx QF_p^{-1}BQ^{-1}B^T \tag{79}$$

In the continuous case we can define F_p as

$$F_{p;i,j} = \int_\Omega \mu \nabla\psi_j \cdot \nabla\psi_i + \int_\Omega \rho(\mathbf{w}_h \cdot \nabla\psi_j) \cdot \psi_i. \tag{80}$$

This can be written as $F_p = A_p + N_p$. Further the scaled Laplacian term $BQ^{-1}B^T$ is replaced by the sparse pressure Laplacian A_p, where

$$A_{p;i,j} = \int_\Omega \nabla\psi_j \cdot \nabla\psi_i. \tag{81}$$

This yields the PCD approximation to the Schur complement matrix:

$$\hat{S}^{-1} = A_p^{-1}F_pQ^{-1} \tag{82}$$

PCD shows improved convergence properties for solving problems with inflow or outflow boundary conditions [12].

5.4.2 The least-squares commutator preconditioner

The results are a little bit different for the least square commutator as a preconditioner. The performance is sensitive to mesh size, with some degradation when the mesh is refined. The elimination of the mesh dependence can be observed if one uses an adjustment for boundary conditions. The scaling for LSC is observed to be an important factor, and the choice of the scaling is necessary for this type of preconditioner to be efficient [13]. Another way to define F_p is the LSC preconditioner, which chooses F_p so that the discrete commutator is small in a least square sense. It turns out that it is more convenient to consider the adjoint of the commutator and choose F_p by minimizing each individual vector norm of the columns of F_p^T; for further details see [12]. This minimization problem reduces to the weighted least square problem:

$$\min ||[Q^{-1}A^TQ^{-1}B^T]_j - Q^{-1}B^TQ^{-1}[F_p^T]_j||_Q, \tag{83}$$

for each column j. Using the normal equations, F_p can be given as

$$F_p = (BQ^{-1}A^TQ^{-1}B^T)(BQ^{-1}B^T)^{-1}Q \tag{84}$$

We get the Schur complement approximation

$$BA^{-1}B^T = (BQ^{-1}B^T)(BQ^{-1}AQ^{-1}B^T)^{-1}(BQ^{-1}B^T). \tag{85}$$

In practice one replaces Q with its diagonal $T = diag(Q)$, and so one constructs the sparse scaled Laplacian $BT^{-1}B^T$. The LSC Schur complement approximation then takes the form

$$\hat{S}^{-1} = (BT^{-1}B^T)(BT^{-1}AT^{-1}B^T)^{-1}(BT^{-1}B^T) \tag{86}$$

For both preconditioners mesh-independent convergence rates have been observed. Eigenvalue bounds for both methods can be found in [12] and, while for PCD they are h-independent, for LSC the known bounds depend on h. For both the PCD and LSC methods, all known bounds depend on the Reynolds number. Numerical tests in [7] show iteration counts for both methods that are mildly dependent on the Reynolds number. Results in the next section will illustrate the main properties of PCD and LCS preconditioners.

6 Numerical results

In this chapter, we demonstrate the performance of the $ILU(\tau)$ and $ILU(\tau_1, \tau_2)$ preconditioners for different values of discretization, stabilization and threshold parameters. As a testbench, we applied the given method to several problems, such as the flow over the backward-facing step and the driven-cavity flow. For numerical tests, we use the MATLAB implementation of ILU(τ_1,τ_2)(algorithm from section 5.2) . The optimal values of ILU thresholds $\tau_1 = 0.03$, $\tau_2 = 7\tau_1^2$ are taken from [26] where detailed analysis of ILU(τ_1,τ_2) and ILU(τ):= ILU(τ,τ) preconditioners for the Oseen systems without stabilization is given. In all experiments, we use the GMRES method with the right preconditioner defined by the ILU(τ_1,τ_2) or ILU(τ) factorizations. The stopping criterion in all experiments is the decrease of the residual by ten orders of magnitude. The $ILU(\tau_1, \tau_2)$ preconditioner was implemented as described above, and for $ILU(\tau)$ the build-in MATLAB procedure $ilu(\tau)$ was used. All results are computed for matrices preprocessed using five iterations of the balancing algorithm, described on (67). The results will be presented in tables, using the following legend: T_{build} and T_{it} show CPU time spent for building preconditioners and iterations. $T_{CPU} = T_{build} + T_{it}$ and #it is the number of GMRES iterations needed to satisfy the stopping criterion. The fill-in ratio is computed through the following formulas:

$$fill_{LU} = (nz(L) + nz(U))/nz(A), \tag{87}$$

where $nz(A) = \Sigma_{ij} sign|A_{ij}|$ is the number of non-zero entries in A.

One reasonable way [21] to judge the quality of a numerical solution is to compute it for meshes of different sizes and show that the FE solution converges to the true solution of the NSE in some norm.

6.1 Flow over backward-facing step

In this section, we consider two-dimensional backward-facing step flow [28],[18],[6]. It is a popular benchmark problem for incompressible flow problems. We provide results obtained by thorough numerical computations for various expansion ratios defined below and a range of Reynolds numbers. The schematic picture of the test set-up is given in Figure 3. The left boundary is an inflow, and the right boundary is an outflow. The bottom boundary is a solid wall, and for our test problems, the top boundary is a solid wall too .

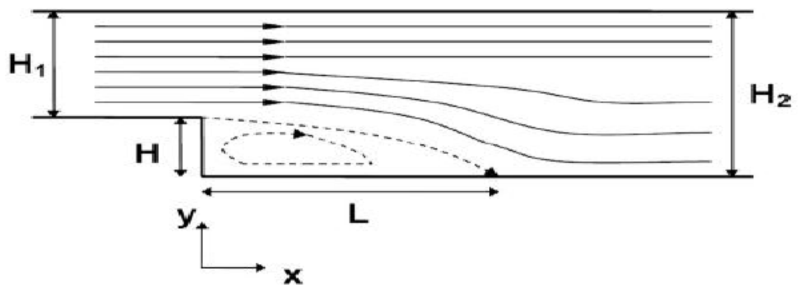

Figure 3: Flow over backward-facing step

Most numerical studies of backward-facing step flows were carried out for a limited number of relevant parameters such as Reynolds number $\mathbf{Re} = \frac{UD}{\nu}$ and expansion ratio H_2/H. Here $D = 2H_1$ denotes the hydraulic diameter of the inlet channel with height H_1; H_2 is the channel height in the expanded region, and $\mathbf{U}$ is the maximum of the inlet velocity. This configuration provides a convenient simple geometric shape for a detailed examination of the rich character of a vortical flow. We used parabolic inflow boundary condition (a Poiseuille flow profile), and natural outflow boundary condition (a no-flow (zero velocity) condition), i.e., the Neumann condition (25) is applied at the outflow boundary. We resolve the model using a sequence of successively refined meshes and compare the computed results for these different meshes.

To discretize the problem, we build several rectangular subdivisions of Ω. First, three increasingly fine meshes with regular rectangular elements are constructed. We compute our results using the sequence of meshes. The coarsest mesh was build by the uniform subdivision for Ω into n rectangles with side $h = 2^{1-l}$, where l is the grid refinement level.

Finer meshes were obtained by a regular global refinement process. The geometric paremeters were set as $L = 15$, $H_1/H = \frac{3}{2}$, and for the maximum inlet velocity we take $U = 1$.

In all experiments in this section, the resulting linear algebra systems are solved by the preconditioned GMRES method with $ILU(\tau_1, \tau_2)$ preconditioner. Based on the results in Tables 1 and 2 below, we chose the optimal value for τ_1, which we use further to compare results with other different parameters. Comparing the fill-in level and time to realize the process, the optimal value for τ_1 was found to be 0.03 (which agrees with [26]). The computations were run with the finest mesh 2 (grid level $l = 3$) for $\nu = 1/100$. For smaller values of τ_1 we observe the increase of fill-in and T_{build}, but the decrease of iteration numbers and T_{it}. We run the same set of experiments with meshes 3 and 4 (grid level $l = 4, 5$), and observed almost the same results in terms of iteration numbers and the dependence of computational times of parameters τ_1 and τ_2. Thus we can say that the optimal value of τ_1 is grid-independent as can be observed from Tables (3) and (4).

We observe large assembling time T_{build}, since the factorization was implemented in MATLAB rather than using efficient external implementation. These times would be significantly shorter, if an external pre-compiled implementation of $ILU(\tau_1, \tau_2)$ is used.

Table 1: Backward step flow, $Q_2 - Q_1$, grid level $l = 5$. The dependence of two-parameter $ILU(\tau_1, \tau_2)$ perfomance on the choice of threshold parameter. Results are shown for $\nu = 1/100$.

τ_1	fill-in	iter	T_{built}	T_{iter}	T_{CPU}
0.1	0.82	90	773.5	0.2	773.52
0.08	0.9	116	993	0.086	993.086
0.07	0.96	112	588.5	0.07	588.57
0.06	1.067	115	823.78	0.12	823.792
0.05	1.45	85	620.45	0.122	620.577
0.04	1.607	55	685	0.47	685.47
0.03	2.075	40	709.4	0.029	709.429
0.02	2.47	24	804	0.022	804.022
0.01	2.81	19	926.4	0.03	926.43
0.007	3.51	14	1113.8	0.022	1113.822
0.005	4.2	12	1284.9	0.012	1284.912

For smaller values of τ we observe the increase of $fill - in$ and T_{build} but the decrease of iteration

Table 2: Backward step flow, $Q_2 - Q_1$ finite elements, grid level $l = 4$. The dependence of two-parameter $ILU(\tau_1, \tau_2)$ perfomance on the choice of threshold parameter. Results are shown for $\nu = 1/100$.

τ_1	fill-in	iter	T_{built}	T_{iter}	T_{CPU}
0.1	zero	pivot	—	—	—
0.08	1.08	97	71.94	0.038	71.978
0.07	1.16	117	73.37	0.025	73.395
0.06	1.27	120	85.12	0.02	85.122
0.05	1.49	120	96.6	0.38	96.98
0.04	1.75	122	111.89	0.019	111.909
0.03	1.97	119	124.69	0.016	124.709
0.02	2.26	95	139.28	0.016	139.296
0.01	2.95	118	183.39	0.02	183.392
0.007	3.12	88	190.44	0.033	190.473
0.005	3.5	67	233.44	0.022	233.462

numbers and T_{it}. So we observe that the optimal value for τ is close to the one reported in [26]. So we decided to keep the optimal value for $\tau_1 = 0.03$.

In tables below (Tables 7 and 8), we compare the same parameters as above for different meshes and also compare results for $ILU(\tau_1, \tau_2)$ with the block-triangle preconditioner from [26] with the PCD approximation of the pressure Schur complement (see Section 5.4) and AMG preconditioner [9] for the (1,1) block of the system. We shall denote this preconditioner as PCD-AMG. From results in Tables 3 and 4 we see that the two-parameter ILU factorization has advantage for certain mesh and problem parameters of the one of the best performing state-of-the-art preconditioner. We run experiments with $\tau = 0.03$, different meshes and 4 types of finite elements.

Table 3: The dependence of two-parameter $ILU(\tau_1, \tau_2)$ performance on the choice of the mesh size. Results are shown for $\nu = 1/100$, $\tau_1 = 0.03$ and $\tau_2 = 7\tau_1^2$.

$Q_1 - Q_1$	8x24	16x48	32x96
T_{build}	159	3738	130245
T_{it}	0.03	0.27	1.69
it	29	76	259
fill-in	1.69	1.99	2.85
PCD-AMG	64	56	56

$Q_1 - P_0$	8x24	16x48	32x96
T_{build}	112	2602	98221
T_{it}	0.06	0.37	2.13
it	103	118	414
fill-in	1.89	1.93	2.25
PCD-AMG	62	59	56

Table 4: The dependence of two-parameter $ILU(\tau_1, \tau_2)$ performance on the choice of the mesh size. Results are shown for $\nu = 1/100$, $\tau_1 = 0.03$ and $\tau_2 = 7\tau_1^2$.

$Q_2 - Q_1$	8x24	16x48	32x96
T_{build}	119	2231	60348
T_{it}	0.07	0.36	2.47
it	119	301	583
fill-in	1.97	1.64	1.84
PCD-AMG	72	75	63

$Q_2 - P_1$	8x24	16x48	32x96
T_{build}	171	3440	97676
T_{it}	0.09	0.32	0.71
it	176	56	139
fill-in	3.06	2.24	2.39
PCD-AMG	167	160	93

We do not see a big difference in the number of iterations for stable and stabilized finite elements. The iteration numbers increase for finer meshes as expected for a method based on an ILU factorization. For PCD-AMG, the numbers of iterations are almost the same for all mesh sizes and elements.

Table 5: The performance of $ILU(\tau)$ $Q_2 - Q_1$ and $Q_1 - Q_1$, using a 16×48 uniform grid (grid level $l = 4$), for the backward step flow, $\tau = 0.07$ without balancing. The results are shown for various values of viscosity ν.

$Q_2 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	zero	pivot	—	—
$\nu = 1/10$	0.045	0.33	119	0.72
$\nu = 1/50$	0.24	0.37	76	2.31
$\nu = 1/100$	0.65	0.30	45	4.33
$\nu = 1/200$	0.62	0.56	85	5.07
$\nu = 1/250$	0.54	0.54	89	5.05
$\nu = 1/300$	zero	pivot	—	—
$\nu = 1/400$	zero	pivot	—	—

$Q_1 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	0.11	1.02	404	0.74
$\nu = 1/10$	0.15	0.37	175	0.94
$\nu = 1/50$	0.25	0.6	144	1.18
$\nu = 1/100$	div	—	—	—
$\nu = 1/200$	div	—	—	—
$\nu = 1/250$	16.7	1.54	14	66.08
$\nu = 1/300$	20.17	1.21	9	74.13
$\nu = 1/400$	—	zero	pivot	—

To see benefits of using $ILU(\tau_1, \tau_2)$, we will compare convergence results with those for $ILU(\tau)$. We show in Tables 5 and 6 iteration numbers for all values of ν, from $\nu = 1$ to $\nu = \frac{1}{400}$. Level of fill-in increases while the value of ν decreases. The number of iterations decreases till $\nu = \frac{1}{100}$ and then increases again. If we compare the numbers of iterations in Table 5 with those in the Table 6, we can see that the application of balancing is very important. For $ILU(\tau)$ we did not see any effect of balancing. Also $ILU(\tau)$, after ν becomes more than 1/50, shows huge level of fill-in and solution does not converge. The level of fill-in for $ILU(\tau_1, \tau_2)$ is two times smaller compare to $ILU(\tau)$ for $Q_2 - Q_1$ elements. Therefore, with almost the same number of iterations $ILU(\tau_1, \tau_2)$ is much cheaper in realization.

Table 6: The dependence of performance of $ILU(\tau_1, \tau_2)$ on the choice of kinematic viscosity parameter. For $Q_2 - Q_1$ and $Q_1 - Q_1$ finite elements, using a 16×48 uniform grid (grid level $l = 4$), for the backward step flow, $\tau_1 = 0.03$ with balancing. The results are shown for various values of viscosity ν.

$Q_2 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	1476	0.19	57	0.95
$\nu = 1/10$	2726	0.5	25	2.28
$\nu = 1/50$	2167	0.16	22	1.74
$\nu = 1/100$	4362	1.16	297	1.64
$\nu = 1/200$	2044	0.36	233	1.84
$\nu = 1/250$	1904	0.35	246	1.94
$\nu = 1/300$	1806	0.35	365	1.98
$\nu = 1/400$	zero	pivot	—	—

$Q_1 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	4328	2.99	563	2.12
$\nu = 1/10$	8058	2.58	90	4.45
$\nu = 1/50$	4761.2	0.37	55	2.48
$\nu = 1/100$	4031	0.32	76	1.99
$\nu = 1/200$	3489	0.68	125	2.02
$\nu = 1/250$	3337	0.74	215	2.09
$\nu = 1/300$	3256	1.01	225	2.19
$\nu = 1/400$	2737	0.81	361	2.35

We also examine the difference in performance of PCD-AMG and LSC-AMG, both methods, are already implemented in IFISS, to see if we have any improvement over ILU(τ_1, τ_2). From the Tables 7 and 8 we can see that PCD-AMG and LSC-AMG preconditioners do not change the number of iterations for finer meshes and have almost the same number of iterations for both types of finite elements. If we compare with Table 6, on the two finest grids used, we can say that LSC and PSD are more efficient and require less number of iterations. It is also easy to see that the dependence on the kinematic viscosity as a parameter is the following: the larger iteration number can be seen as smaller the viscosity parameter.

Table 7: Backward facing step, $Q_1 - Q_1$, grid level $l = 4, 5$, comparison of PCD-AMG and LSC-AMG.

$l = 4$	PSD		LSC	
ν	*iter*	T_{build}	iter	T_{build}
1/10	21	2.05	21	5.533
1/100	29	2.9	37	8.9
1/400	104	10.5	132	32.7

$l = 5$	PSD		LSC	
ν	*iter*	T_{build}	iter	T_{build}
1/10	21	1.01	21	2.53
1/100	26	1.22	30	3.5
1/400	58	2.7	87	10.2

Table 8: Backward facing step, $Q_2 - Q_1$, grid level $l = 4, 5$, comparison of PCD-AMG and LSC-AMG.

$l = 4$	PSD		LSC	
ν	*iter*	T_{build}	iter	T_{build}
1/10	23	1.9	10	1.23
1/100	30	2.67	21	2.45
1/400	73	7.07	64	8.03

$l = 5$	PSD		LSC	
ν	*iter*	T_{build}	iter	T_{build}
1/10	22	1	11	7.13
1/100	30	1.28	16	9.8
1/400	54	23.3	44	26.4

6.2 The driven cavity flow

This problem is often employed to evaluate numerical methods and to validate codes for solving the Navier-Stokes equations [8], [32],[39],[15]. With a simple geometry of a cavity problem, applying a numerical method to this flow problem in terms of coding is quite easy and straightforward. Another reason for the problem to be so popular is that it is rather easy to generate solutions exhibiting global features such as vortexes and layers one would expect in a flow.

Here the flow domain is a square; see Figure 4. Along all walls except the top one, the velocity is required to vanish. Along the top wall, the normal velocity component vanishes, and the tangential component is prescribed to a constant. Therefore, the model is a flow in a square cavity with the lid moving from left to right. A difficulty associated with the driven cavity flow is that the flow contains (at the upper corners) strong, nonphysical singularities (Figure 5). The effect of these singularities can be mitigated in various ways by smoothing out the transition between the boundary conditions at the corners.

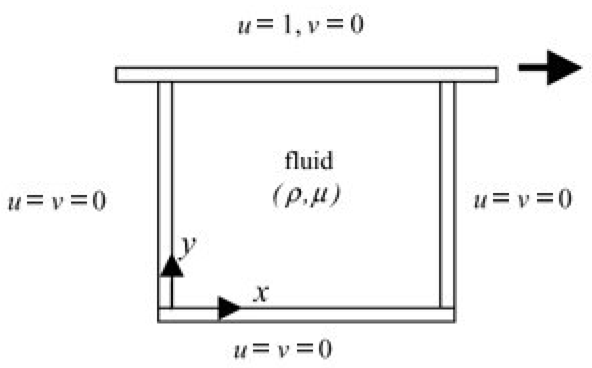

Figure 4: Cavity flow

We used a square domain Ω and enclosed flow boundary condition as illustrated in Figure 4.

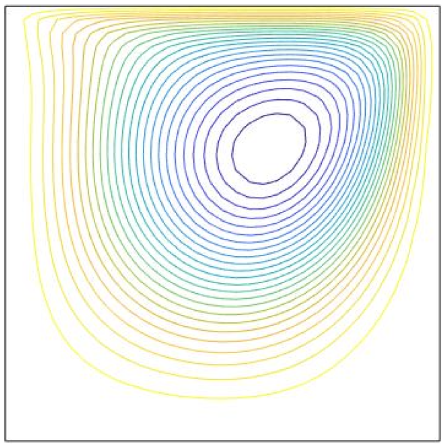

Figure 5: Streamlines of the flow solution to the cavity flow

We have three different computational models:

$$\begin{aligned} \{y = 1, -1 \leq x \leq 1 | u_x = 1\}, &\text{ a leaky cavity} \\ \{y = 1, -1 < x < 1 | u_x = 1\}, &\text{ a watertight cavity} \\ \{y = 1, -1 \leq x \leq 1 | u_x = 1 - x^4\}, &\text{ a regularized cavity.} \end{aligned} \tag{88}$$

For the dissertation, the regularized cavity problem was solved on uniform grids with grid levels $l = 3, 4, 5$, and the wide range of Reynolds numbers (Re) increasing from zero to 1000 was used to verify the suitability of the proposed numerical algorithm.

To choose optimal threshold parameter τ_1, we run set of experiments for the cavity model with $Q_2 - Q_1$ finite element discretization and kinematic viscosity $\nu = 1/100$ and sequence of parameters τ_1 in range from 0.1 to 0.005. We can see from Tables 10 and 9 that the optimal $\tau_1 = 0.03$ leads to a good balance between observed values as level of fill-in and rate of convergence. We observe that the performance of the preconditioner does not change significantly for finer mesh, so the optimal value of τ_1 was found the same. Therefore we will run $ILU(\tau_1, \tau_2)$ with this optimal value in experiments below.

In another set of tests, the effect of problem size was investigated, using three different grids with 64×64, 32×32, 16×16, 8×8 grid cells. Tables 11 and 12 show results of numerical simulations

Table 9: Lid driven cavity flow, $Q_2 - Q_1$, grid level $l = 4$. The dependence of two-parameter $ILU(\tau_1, \tau_2)$ performance on the choice of threshold parameter. Results are shown for $\nu = 1/100$.

τ_1	fill-in	iter	T_{built}	T_{iter}	T_{CPU}
0.1	—	—	—	—	—
0.08	—	—	—	—	—
0.07	1.22	13	2.18	0.085	2.265
0.06	1.29	11	2.28	0.008	2.288
0.05	1.4	9	2.41	0.012	2.422
0.04	1.5	9	3.01	0.027	3.037
0.03	1.61	8	2.92	0.013	2.933
0.02	1.75	7	3.094	0.002	3.096
0.01	2.03	5	3.48	0.001	3.481
0.007	2.12	5	3.28	0.001	3.281
0.005	2.21	4	3.99	0.001	3.991

Table 10: Cavity flow, $Q_2 - Q_1$, grid level $l = 5$. The dependence of two-parameter $ILU(\tau_1, \tau_2)$ performance on the choice of threshold parameter. Results are shown for $\nu = 1/100$.

τ_1	fill-in	iter	T_{built}	T_{iter}	T_{CPU}
0.1	1.07	23	15.67	0.32	15.99
0.08	1.22	21	16.34	0.015	16.355
0.07	1.33	20	17.2	0.005	17.205
0.06	1.46	18	18.6	0.005	18.605
0.05	1.64	16	21.37	0.014	21.384
0.04	1.88	14	29.57	0.008	29.578
0.03	2.13	12	29.78	0.002	29.782
0.02	2.45	9	38.6	0.022	38.622
0.01	3.033	7	60.9	0.008	60.908
0.007	3.35	6	71.62	0.004	71.624
0.005	3.65	5	52.21	0.013	52.34

with $\nu = 1/100$ for both stabilized and stable finite elements. We did not see any difference when the stabilization parameter β was changed, so we keep it equal to 1/3 as it is given by default in IFISS. As the problem size increases, we can see how the iterations number increases as well. This result is expected, since refining a mesh usually increases the condition number of the matrix problem, therefore reducing the rate of convergence of the GMRES method. The level of fill-in increases rapidly for the level grid $l = 6$. We can also observe that $ILU(\tau_1, \tau_2)$ is more efficient for stable finite elements discretization than the scheme with preconditioning PCD-AMG in terms of the number of iterations.

Next, we run a set of numerical experiments to show dependence on the viscosity parameter

Table 11: Lid-driven cavity flow,grid level 3. The dependence of two-parameter $ILU(\tau_1, \tau_2)$ performance on the choice of the grid level. Results are shown for $\nu = 1/100$ and $\tau_1 = 0.03$ and different finite elements.

$Q_1 - Q_1$	64x64	32x32	16x16	8x8
T_{build}	29054	1176	46	4
T_{it}	0.8	0.09	0.04	0.05
it	75	27	15	11
fill-in	6.42	4.51	3.28	2.13
AMG	46	42	39	33

$Q_1 - P_0$	64x64	32x32	16x16	8x8
T_{build}	19873	782	34	3
T_{it}	0.3	0.32	0.04	0.04
it	82	42	19	24
fill-in	5.71	3.84	3.14	2.01
AMG	48	39	38	32

Table 12: Lid-driven cavity flow. The dependence of two-parameter $ILU(\tau_1, \tau_2)$ performance on the choice of the grid level. Results are shown for $\nu = 1/100$ and $\tau_1 = 0.03$ and different finite elements.

$Q_2 - Q_1$	64x64	32x32	16x16	8x8
T_{build}	12040	512	25	2.5
T_{it}	0.2	0.06	0.04	0.04
it	40	16	11	9
fill-in	4.125	2.971	2.08	1.69
AMG	47	44	37	30

$Q_2 - P_1$	64x64	32x32	16x16	8x8
T_{build}	19832	775	38	4
T_{it}	0.3	0.17	0.03	0.038
it	55	19	12	10
fill-in	4.19	3.21	2.82	2.73
AMG	58	49	44	41

for stable and stabilized finite elements as we have done in a previous test. From Tables 13 and 14, which show dependence on the choice of kinematic viscosity parameter in range from 1/32 to 1/1024, we can see that $ILU(\tau_1, \tau_2)$ works better for stabilized finite elements and decreases number of iteration in two times. Furthermore, the level of fill-in also two times less for stabilized elements.

For the purpose of comparison, we include results of preconditioned $GMRES$ iteration with the built-in MATLAB $ILU(\tau)$ preconditioner. We next look at the $\tau-$dependence of the build-in $ILU(\tau)$ preconditioner. We run this computations for finest mesh 6 and $\nu = 1/100$ The results are presented in Table 15. We observed that that for smaller values of τ fill-in and T_{build} increase, but iteration number and T_{it} decrease. So the optimal τ value is found to be 0.08 or 0.07. We can observe from Tables 9 and 10 that two-parameter preconditioner works much better in terms of the number of iterations and decreases it comparing to $ILU(\tau)$ for τ closer to 0.1 in three times and for τ less than 0.02 in two times.

The next series of experiments (Tables 16 and 17) shows that the application of balancing is also important in practice for the cavity problem. In all experiments in Table 17 we performed

Table 13: Lid driven cavity flow, $Q_1 - Q_1$ finite elements, grid level $l = 5$, $\tau_1 = 0.03$. The dependence of two parameter $ILU(\tau_1, \tau_2)$ performance on the choice of kinematic viscosity parameter.

	$\nu = 1/32$	$\nu = 1/128$	$\nu = 1/512$	$\nu = 1/1024$
T_{build}	1634.64	894	662.6	592.9
T_{it}	0.459	0.07	0.130	0.26
fill-in	5.59	4.87	4.62	4.59
iter	23	24	82	231

Table 14: Lid driven cavity flow, $Q_2 - Q_1$ finite elements, grid level $l = 5$, $\tau_1 = 0.03$. The dependence of two parameter $ILU(\tau_1, \tau_2)$ performance on the choice of kinematic viscosity parameter.

	$\nu = 1/32$	$\nu = 1/128$	$\nu = 1/512$	$\nu = 1/1024$
T_{build}	645.41	447.63	426.308	small pivot
T_{it}	0.103	0.058	0.065	
fill-in	3.16	2.77	3.01	
iter	16	11	30	

five iterations to find the scaling vectors before any incomplete factorization was computed. The computations were run on the finest mesh 4 and with optimal values for τ and τ_1 from Tables 9 and 15. We can see results similar to those found in Table 5 for the backward-facing step problem. We conclude that the balancing is very important and in the cavity flow, it decreases the number of iterations by a factor of 3. Without this preprocessing $ILU(\tau)$ fails to converge for small values of kinematic viscosity when we used stabilized finite elements.

The two-parameter ILU factorization leads to a more efficient preconditioner in terms of memory usage (fill-in) and iteration counts, but with more expensive set-up stage, compared to the standard ILU(τ). We do not consider the T_{build} values to conclude about the efficiency of the $ILU(\tau_1, \tau_2)$ preconditioner because of the same realization of this algorithm in Fortran [26] results in much shorter built times. Our version was implemented in Matlab, and because of some build-in functions, the execution takes much more time.

In the next set of test experiments, we compare the difference of using the convection-diffusion preconditioner with (PCD) with AMG for the velocity block and the least-squares commutator (LSC) preconditioner. These preconditioners are quite robust with respect to grid size and viscosity parameter. We see from the Tables 18, 19 below that PCD preconditioner works much better in

Table 15: Lid driven cavity flow, $Q_2 - Q_1$ finite elements, $\nu = 1/100$, grid level $l = 6$. The dependence of one parameter $ILU(\tau)$ perfomance on the choice of threshold parameter.

τ	iter	T_{built}	T_{iter}	T_{CPU}	fill in
0.1	90	0.018	0.123	0.141	0.77
0.08	74	0.042	0.06	0.102	0.96
0.07	74	0.058	0.064	0.122	1.09
0.06	57	0.063	0.057	0.12	1.248
0.05	53	0.079	0.044	0.123	1.506
0.04	40	0.107	0.032	0.139	1.77
0.03	29	0.131	0.029	0.16	2.155
0.02	23	0.21	0.023	0.233	2.93
0.01	16	0.325	0.02	0.345	5.12
0.007	13	0.372	0.0181	0.39	6.85
0.005	11	0.54	0.02	0.56	9.15
0.003	9	0.88	0.023	0.9	14.08
0.002	8	1.43	0.022	1.45	18.7
0.001	8	1.7	0.03	1.74	26.83

terms of number of iterations and also does not depend on the choice of finite elements. Mesh-independence of the GMRES convergence rate is again apparent. The increase in iterations is required as the kinematic viscosity parameter decreases. It is easy to see that LSC does not work with stable finite elements $Q_2 - Q_1$. For stabilized finite elements, we still observe that PCD works much better in terms of T_{build} and number of iterations. For PCD, iteration time does not increase that much as for LSC as the viscosity coefficient decreases. If we compare PCD and LSC with $ILU(\tau_1, \tau_2)$, we can see that two-parameter ILU works better in terms of number of the iterations and does not have any problem with stabilized finite elements. In most experiments $ILU(\tau_1, \tau_2)$ needs about half of iteration number spent by other methods to reach the same accuracy.

Results show that the choice of parameters and preconditioning methods can significantly influence the rate of convergence, but there is no single precondition that performs uniformly well in all cases.

Table 16: Lid driven cavity flow, $Q_2 - Q_1$ and $Q_1 - Q_1$ finite elements, $\nu = 1/100$, The dependence of one parameter $ILU(\tau)$ preconditioner performance on the choice of kinematic viscosity parameter with $\tau = 0.07$ and without balancing.

$Q_2 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	zero	pivot		
$\nu = 1/10$	0.0016	0.055	23	0.75
$\nu = 1/50$	0.005	0.054	15	1.43
$\nu = 1/100$	0.07	0.05	17	1.75
$\nu = 1/200$	0.008	0.05	27	2.57
$\nu = 1/250$	0.007	0.06	35	2.66
$\nu = 1/300$	0.007	0.07	52	2.75
$\nu = 1/400$	0.008	0.09	60	2.87

$Q_1 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	0.002	0.066	45	0.75
$\nu = 1/10$	0.005	0.09	30	0.9
$\nu = 1/50$	0.004	0.05	24	1.12
$\nu = 1/100$	0.03	0.14	57	7.01
$\nu = 1/200$	0.02	0.55	622	6.15
$\nu = 1/250$	0.02	4.04	div	5.88
$\nu = 1/300$	0.018	3.54	div	5.58
$\nu = 1/400$	0.02	3.9	div	5.97

Table 17: Lid driven cavity flow, $Q_2 - Q_1$ and $Q_1 - Q_1$ finite elements, grid level $l = 5$, The dependence of two-parameter preconditioner $ILU(\tau_1, \tau_2)$ performance on the choice of kinematic viscosity with $\tau_1 = 0.03$ and balancing.

$Q_2 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	20.43	0.23	24	1.01
$\nu = 1/10$	34.46	0.05	9	2.43
$\nu = 1/50$	28.6	0.05	8	2.28
$\nu = 1/100$	23.87	0.05	11	2.08
$\nu = 1/200$	24.3	0.05	14	2.13
$\nu = 1/250$	25.8	0.05	15	2.18
$\nu = 1/300$	26.05	0.06	17	2.23
$\nu = 1/400$	27.6	0.06	23	2.42
$\nu = 1/1000$	small	pivot	.	

$Q_1 - Q_1$	T_{build}	T_{it}	it	fill-in
$\nu = 1$	59	0.3	24	2.17
$\nu = 1/10$	96	0.07	13	4.2
$\nu = 1/50$	57	0.05	13	3.4
$\nu = 1/100$	46	0.06	15	3.2
$\nu = 1/200$	35	0.05	22	3.01
$\nu = 1/250$	35.6	0.05	24	3
$\nu = 1/300$	51.2	0.05	26	2.99
$\nu = 1/400$	33	0.06	45	2.92
$\nu = 1/1000$	28.7	0.45	78	2.74

Table 18: Lid driven cavity flow, $Q_1 - Q_1$ finite elements, grid level $l = 6, 5$ comparison modified pressure convection-diffusion and boundary-adjusted least-squares commutator preconditioners.

	PCD		LSC	
ν	*iter*	T_{build}	iter	T_{build}
1/10	19	0.28	30	0.163
1/100	36	0.27	54	0.34
1/1000	127	0.49	180	1.177

	PCD		LSC	
ν	*iter4*	T_{build}	iter	T_{build}
1/10	21	0.104	30	0.193
1/100	36	0.135	54	0.348
1/1000	127	0.63	158	4.85

Table 19: Lid driven cavity flow, $Q_2 - Q_1$ finite elements, grid level $l = 6, 5$ comparison modified pressure convection-diffusion and boundary-adjusted least-squares commutator preconditioners.

$l = 6$	PCD		LSC	
ν	*iter*	T_{build}	iter	T_{build}
1/10	8	0.064	div	—
1/100	29	0.08	div	—
1/1000	64	1.8	div	—

$l = 5$	PCD		LSC	
ν	*iter*	T_{build}	iter	T_{build}
1/10	29	0.028	div	—
1/100	41	0.037	div	—
1/1000	162	0.408	div	—

7 Deep learning in application to ILU preconditioning

In this chapter, we bridge deep learning networks and preconditioner design. We propose a convolutional neural network with a specific architecture to predict a fill-in pattern for ILU factors further used in preconditioning.

7.1 General Static Pattern ILU

Incomplete LU factorization gives as an output a sparse lower triangular matrix L and a sparse upper triangular matrix U in a way such that it has non-zero entries in some locations. In a traditional approach, the non-zero pattern is chosen in advance [35]. Although the only restriction on the fill-in is that it has to include diagonal elements (as it was shown in [35]), in a standard approach the pattern is based on the fill-in structure of the original matrix. Such traditional approach does not work for saddle point problems, where the (2,2)-block can be zero in the original matrix but is densely filled in the LU decomposition. For any pattern P, which satisfies the condition for diagonal elements

$$P \subset \{(i,j); 1 \leq i,j \leq n\}, \tag{89}$$

then pattern-based ILU factorization has the following algorithm:

```
for $k = 1, \dots, n-1$
    for $i = k+1, \dots, n$ and if $(i,k) \in P$:
        $a_{ik} := a_{ik}/a_{kk}$
        for $j = k+1, \dots, n$ and if $(i,j) \in P$:
            $a_{ij} = a_{ij} - a_{ik} * a_{kj}$
        End
    End
End
```

Based on the method described above, we decided to build a neural network that can predict a suitable and effective pattern for the ILU decomposition without resorting to the fill-in structure of the given matrix.

To generate a set of patterns and to train a neural network, we apply the following algorithm.

Algorithm 7.1:

1. Compute exact LU decomposition for a given sparse matrix A.
2. Find indices of $3N$ non-diagonal entries with the largest absolute value in L, where N is the dimension of the matrix A. Next, do the same for U.
3. Add diagonal elements to two sets of indices above to get a full set of indices.
4. Construct a binary matrix with a non-zero pattern from the obtained set of indices.

This resulting set of binary matrices is further used to define an objective of training.

7.2 Training set

To generate the training set as an input for this experiment, we take the lid-driven cavity problem as it was set up in Chapter 6. We modify the Navier-Stokes equation as follows:

$$\begin{cases} \alpha\mathbf{u} - \nu\Delta\mathbf{u} + (\mathbf{w}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega \\ div(\mathbf{u}) = 0 \text{ in } \Omega, \end{cases} \tag{90}$$

Note that for the purpose of training we replace the inertia term $(\mathbf{u}\cdot\nabla)\mathbf{u}$ by $(\mathbf{w}\cdot\nabla)\mathbf{u}$, where $\mathbf{w}$ has two components in the form of Fourier series:

$$\begin{aligned} w_1 &= \Sigma_{ij=1}^{3}\alpha_{ij} sin(ix)\cdot cos(jy) \\ w_2 &= \Sigma_{ij=1}^{3}\beta_{ij} sin(ix)\cdot sin(jy). \end{aligned} \tag{91}$$

We take random uniform distribution for coefficients α_{ij} in the interval $[0,10]$ and $\alpha = 0.01$.

For a given set of coefficients we use $Q_2 - Q_1$ finite elements and apply 5 Picard iterations to generate the matrix formulation of the problem (90):

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{92}$$

Then we apply ILU-factorization algorithm 7.1 to generate a pattern matrix. The pair of the two matrices describes a single data point in the training set.

Before we feed a matrix to the network, we do a preprocessing step. We normalize each of three nonzero blocks using the formula (93):

$$\begin{pmatrix} \alpha & & & & \\ & \ddots & & & \\ & & \alpha & & \\ & & & \beta & \\ & & & & \ddots \\ & & & & & \beta \end{pmatrix} \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \alpha & & & & \\ & \ddots & & & \\ & & \alpha & & \\ & & & \beta & \\ & & & & \ddots \\ & & & & & \beta \end{pmatrix} \tag{93}$$

where $\alpha = 1/\sqrt{||A||}$ and $\beta = \sqrt{||A||}/||B||$ and $|| \cdot ||$ denotes the Frobenius norm.

We generate a training set of 1000 examples. The input matrices and their corresponding patterns are used to train the network (Figure 6 and 7).

7.3 U-net architecture for convolutional networks

In the last decade neural networks have made huge progress in many image recognition tasks. Initially, convolutional networks were used for classification tasks, where input is an image and output is the corresponding label. In 2015 a new efficient algorithm was developed for various biomedical image segmentation problems [34]. This network had an image as input and a binary mask-matrix as output. We choose to apply this type of networks to our matrix formulation of the Navier-Stokes equation (51) and get ILU patterns as output.

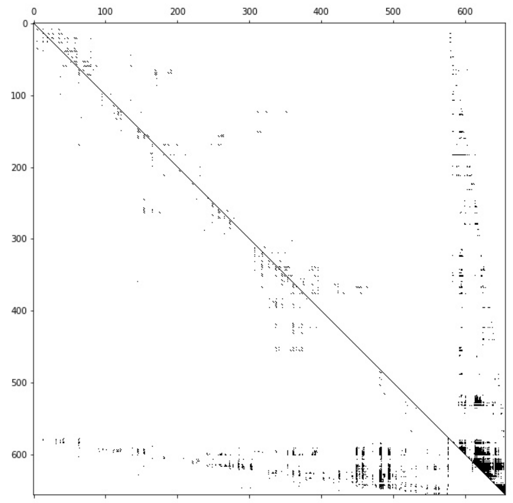

Figure 6: Pattern predicted by CNN

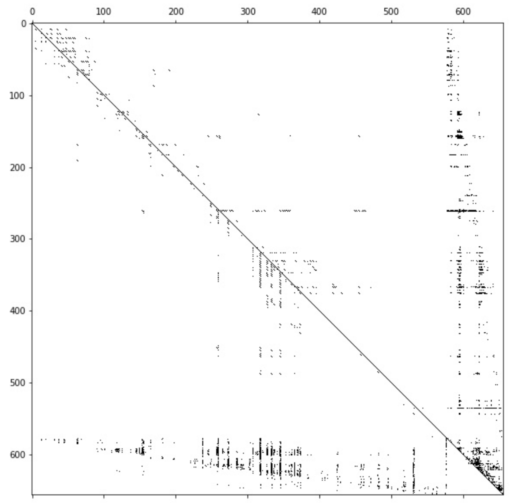

Figure 7: Pattern by algorithm 7.1

7.4 CNN architecture

We use the architecture as it was given in [34], but we modify the dimensions of the input and output layers to satisfy our problem dimensions . It consists of three sections: the contracting path (left side), the bottleneck and the expansive path (right side). The contracting path follows the common architecture of a convolutional network. It consists of the repeated application of couple of 3×3 convolutions followed by 2×2 max pooling. At each step of downsampling, we double the number of feature channels so the architecture can learn patterns effectively. The bottleneck is made from 2 convolutional layers with dropout.

The expansion section is similar to the contraction section. It consists of an upsampling of the feature map followed by a 2×2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3×3 convolutions. The number of expansion blocks is as same as the number of contraction block. In total the net- work has 23 convolutional layers (Figure 8).

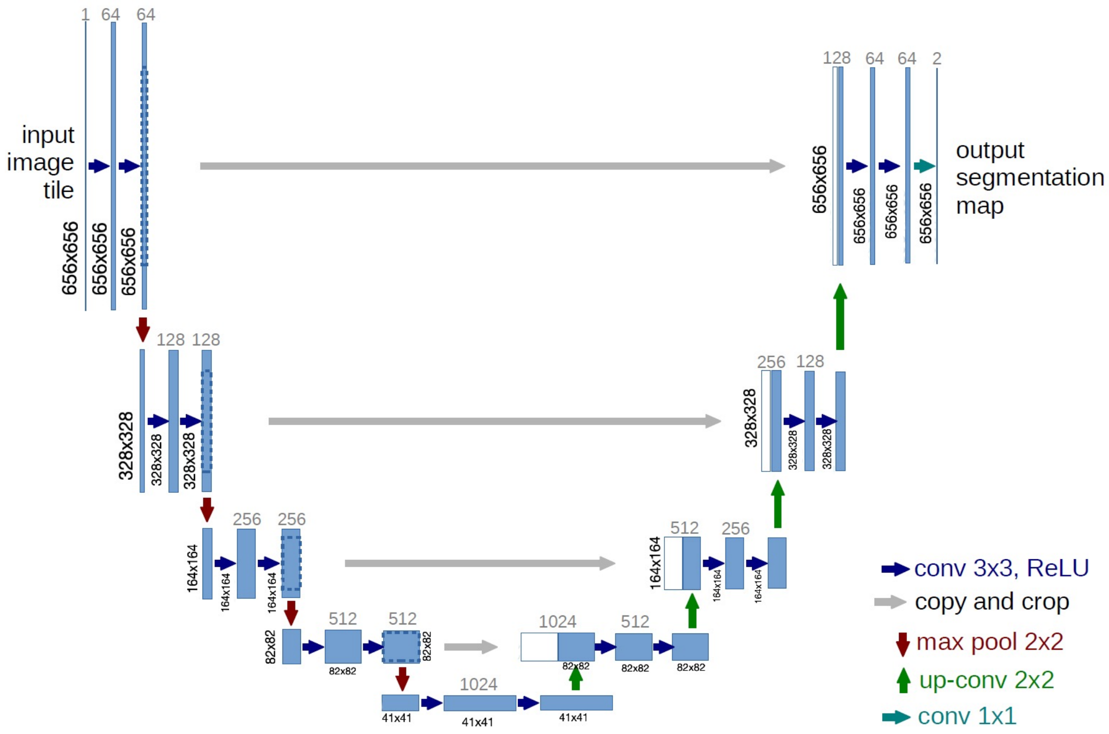

Figure 8: U-net architecture (based on Fig.1 in [34])

7.5 Experiments

To train the U-net, we use Adam optimization algorithm with a learning rate 0.001 and binary cross-entropy loss function. We tested batch sizes $b = 1, 2, 3$ due to the lack of GPU memory and got the best accuracy with batch size $b = 2$.

We test the application of U-net CNN to the general Navier-Stokes formulation (28) for lid-driven cavity problem with different values of kinematic viscosity. The test data for the cavity model is constructed by using $Q_2 - Q_1$ finite elements and mesh level $l = 4$ (16×16) to generate the matrix formulation of the given problem. For the purpose of comparison, we choose $ILU(\tau)$ as preconditioner and optimize threshold parameter τ in a way that it has the same level of fill-in as preconditioner generated using neural network output. We adjust parameter τ for each value of kinematic viscosity to get a comparable level of fill-in. From Table 20, we can see that the rate of convergence of the GMRES method is somewhat lower for ILU generated using the pattern from

U-net and diverges for $\nu < 1/400$.

Table 20: Lid driven cavity flow, $Q_2 - Q_1$finite elements, grid level $l = 4$ comparison of U-net pattern ILU and $ILU(\tau)$.

	U-net		ILU(τ)		
ν	it	fill-in	τ	it	fill-in
1/50	194	0.63	0.15	64	0.466
1/100	253	0.51	0.2	60	0.5
1/200	418	0.45	0.3	239	0.51
1/300	326	0.52	0.27	135	0.53
1/400	div	—	0.23	175	0.87

The conducted experiment shows that our proposed neural network-based method of building preconditioners works for the lid-driven cavity flow problem up to a certain Reynolds number. Currently, it performs worse in terms of convergence speed than a threshold based incomplete LU preconditioning with similar fill-in level but leaves room for improvement. We expect that trying different network architectures, training hyperparameters, and using more powerful hardware could significantly improve on the quality of generated preconditioners.

8 Conclusions

In this dissertation we studied preconditioning techniques for linear algebraic systems that arise from finite element approximation of the Navier-Stokes equations. Our focus was on testing, improving and showing advantages of LU-type preconditioners applied to fluid dynamics problems. We studied the performance of proposed preconditioners in the Krylov subspace method based on two-parameter threshold ILU factorization of non-symmetric saddle point problems. It was observed that the $ILU(\tau_1, \tau_2)$ preconditioner for a suitable choice of parameters has a low fill-in and leads to faster convergent iterations compared to $ILU(\tau)$. Advantages of two-parameter incomplete factorization preconditioners can be seen from numerical results in Chapter 6. $ILU(\tau_1, \tau_2)$ is insensitive to the type of finite element discretization and domain geometry. We found that two-parameter $ILU(\tau_1, \tau_2)$ can be successfully used in practice and improves the performance of the iterative method.

With the increasing applications of deep learning in image analysis and other fields, we tested the image segmentation algorithm applied to a discrete formulation of the Navier-Stokes equation in the matrix-vector form. Image segmentation is a task in which we classify each pixel of an image as belonging to a non-zero pattern. The U-Net convolutional neural network is one of the most well-recognized algorithms to solve that problem. It is effective even with a limited dataset. Our results in Chapter 7 showed that convolutional neural networks can be used for generating patterns for L and U factors for ILU-type preconditioners. It is a first step towards using deep learning in preconditioner design. We expect to improve performance with bigger and more diverse data sets as well as with tailored network architecture.

Bibliography

[1] Z.-Z. Bai, G. H. Golub, and M. K. Ng. Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 24(3):603–626, 2003.

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

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

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

[5] M. Benzi, M. A. Olshanskii, and Z. Wang. Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 66(4):486–508, 2011.

[6] G. Biswas, M. Breuer, and F. Durst. Backward-facing step flows for various expansion ratios at low and moderate Reynolds numbers. *J. Fluids Eng.*, 126(3):362–374, 2004.

[7] N. Bootland, A. Bentley, C. Kees, and A. Wathen. Preconditioners for two-phase incompressible Navier-Stokes flow. *arXiv preprint arXiv:1710.08779*, 2017.

[8] O. Botella and R. Peyret. Benchmark spectral results on the lid-driven cavity flow. *Computers & Fluids*, 27(4):421–433, 1998.

[9] J. Boyle, M. Mihajlovic, and J. Scott. Hsl mi20: an efficient AMG preconditioner. Technical report, Citeseer, 2007.

[10] J. H. Bramble and J. E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Mathematics of Computation*, 50(181):1–17, 1988.

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

[12] H. C. Elman, D. Silvester, and A. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, 2014.

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

[14] H. C. Elman and R. S. Tuminaro. Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations. *Electron. Trans. Numer. Anal*, 35:257–280, 2009.

[15] E. Erturk, T. C. Corke, and C. Gökçöl. Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 48(7):747–774, 2005.

[16] R. E. Ewing, R. D. Lazarov, P. Lu, and P. S. Vassilevski. Preconditioning indefinite systems arising from mixed finite element discretization of second-order elliptic problems. In *Preconditioned Conjugate Gradient Methods*, pages 28–43. Springer, 1990.

[17] P. E. Farrell, L. Mitchell, and F. Wechsung. An Augmented Lagrangian Preconditioner for the 3D stationary Incompressible Navier-Stokes Equations at High Reynolds Number. *SIAM Journal on Scientific Computing*, 41(5):A3073–A3096, 2019.

[18] D. K. Gartling. A test problem for outflow boundary conditions—flow over a backward-facing step. *International Journal for Numerical Methods in Fluids*, 11(7):953–967, 1990.

[19] V. Girault and P.-A. Raviart. Finite element approximation of the Navier-Stokes equations. *Lecture Notes in Mathematics, Berlin Springer Verlag*, 749, 1979.

[20] G. H. Golub and C. v. Loan. *Matrix computations.* Baltimore, MD: Johns Hopkins University Press, 1996.

[21] M. D. Gunzburger. *Finite Element Methods for Viscous Incompressible flows.* Academic Press, 1989.

[22] J. G. Heywood and R. Rannacher. Finite element approximation of the nonstationary Navier-Stokes problem. I. Regularity of solutions and second-order error estimates for spatial discretization. *SIAM Journal on Numerical Analysis*, 19(2):275–311, 1982.

[23] I. E. Kaporin. High quality preconditioning of a general symmetric positive definite matrix based on its U^TU+ U^TR+ R^TU-decomposition. *Numerical Linear Algebra with Applications*, 5(6):483–509, 1998.

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

[25] I. Konshin, M. Olshanskii, and Y. Vassilevski. LU factorizations and ILU preconditioning for stabilized discretizations of incompressible Navier-Stokes equations. *Numerical Linear Algebra with Applications*, 24(3):e2085, 2017.

[26] I. N. Konshin, M. A. Olshanskii, and Y. V. Vassilevski. ILU preconditioners for nonsymmetric saddle-point matrices with application to the incompressible Navier-Stokes equations. *SIAM Journal on Scientific Computing*, 37(5):A2171–A2197, 2015.

[27] W. Layton. A two-level discretization method for the Navier-Stokes equations. *Computers Mathematics with Applications*, 26(2):33 – 38, 1993.

[28] H. Le, P. Moin, and J. Kim. Direct numerical simulation of turbulent flow over a backward-facing step. *Journal of Fluid Mechanics*, 330:349–374, 1997.

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

[30] M. A. Olshanskii and A. Reusken. Navier-Stokes equations in rotation form: A robust multigrid solver for the velocity problem. *SIAM Journal on Scientific Computing*, 23(5):1683–1706, 2002.

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

[32] A. Prasad, C.-Y. Perng, and J. Koseff. Some observations on the influence of longitudinal vortices in a lid-driven cavity flow. In *1st National Fluid Dynamics Conference*, page 3654, 1988.

[33] R. Rannacher. Finite element methods for the incompressible Navier-Stokes equations. In *Fundamental Directions in Mathematical Fluid Mechanics*, pages 191–293. Springer, 2000.

[34] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241. Springer, 2015.

[35] Y. Saad. *Iterative methods for sparse linear systems.* Volume 82. SIAM, 2003.

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

[37] D. Silvester and A. Wathen. Fast iterative solution of stabilised stokes systems part II: using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367, 1994.

[38] D. J. Silvester. Efficient solution of the steady-state Navier-Stokes equations using a multigrid preconditioned Newton-Krylov method. *International Journal for Numerical Methods in Fluids*, 43(12):1407–1427, 2003.

[39] T. E. Tezduyar, S. Mittal, S. Ray, and R. Shih. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering*, 95(2):221–242, 1992.

[40] J. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods.* Texts in Applied Mathematics. Springer, 1 edition, 1999.

[41] M. ur Rehman, C. Vuik, and G. Segal. A comparison of preconditioners for incompressible Navier-Stokes solvers. *International Journal for Numerical Methods in Fluids*, 57(12):1731–1751, 2008.