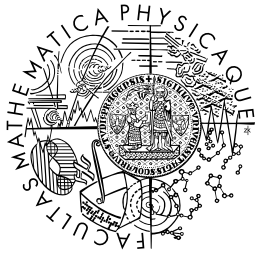


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Erik Mitro

Efficient scalable solvers for incompressible flow problems

Mathematical Institute of Charles University

Supervisor of the master thesis: RNDr. Jaroslav Hron, Ph.D.

Study programme: Mathematics

Study branch: Mathematical modeling in Physics and Technology

Prague 2020

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Hereby, I would like to express my gratitude to my supervisor Jaroslav Hron for his patient guidance, time and advices. I would also like to thank my whole family and my girlfriend for their continued support.

Title: Efficient scalable solvers for incompressible flow problems

Author: Erik Mitro

Institute: Mathematical Institute of Charles University

Supervisor: RNDr. Jaroslav Hron, Ph.D., Mathematical Institute of Charles University

Abstract: In this thesis, the different solution methods for saddle-point systems arising from fluid dynamics are studied. The main emphasis is on Krylov subspace methods with effective preconditioning techniques for saddle-point systems obtained from finite element discretization of the Navier-Stokes equations. Two preconditioning techniques are presented: pressure-convection-diffusion preconditioning (PCD) and least-square commutator preconditioning (LSC). Both preconditioners are validated on two benchmarks: lid-driven cavity and flow around cylinder. From the computational point of view, we focus on comparing the performance of used solvers, with emphasis on our implementation of PCD preconditioning. All numerical simulations are performed by software Firedrake.

Keywords: CFD, FEM, efficient preconditioning

Contents

Introduction

Since saddle-point systems do occur frequently, there is a need to solve such systems effectively. In this thesis, we will study the solution methods for saddle-point systems arising from incompressible flow problems, with emphasis on iterative methods.

Firstly, a basic properties of saddle-point systems will be presented. Afterwards an overview of Krylov subspace methods will be given. For improving the rate of convergence of Krylov subspace methods, block preconditioning techniques will be introduced.

In the second chapter, we will turn our attention to Stokes and Navier-Stokes equations that describe the flow of incompressible fluids. In this thesis, fluids are considered to be incompressible, homogeneous and Newtonian. Using numerical methods as finite element method, we derive the corresponding saddle-point systems. In the case of Stokes problem, the associated saddle-point system is symmetric and indefinite. MINRES method with block diagonal preconditioning will be presented for effective solving. For approximating of the Schur complement matrix, spectrally equivalent pressure mass matrix is used. Distretization of the Navier-Stokes problem leads to nonsymmetric saddle-point system. Krylov subspace method like GMRES can be used for solving such systems. To improve the rate of the convergence of GMRES, block triangular preconditioning will be used. The delicate problem here is the approximation of the Schur complement matrix. Two very popular approximations will be presented: pressure-convection-diffusion (PCD) preconditioning and least-square commutator (LSC) preconditioning.

Finally, we will validate different solvers on two selected benchmarks: lid-driven cavity and flow around a cylinder. For this purpose, we use the software **Firedrake** (Rathgeber et al. [2016]) and PETSc - see Balay et al. [2019], Balay et al. [1997], Dalcin et al. [2011], Karypis and Kumar [1998], Amestoy et al. [2001], Amestoy et al. [2006]. As firedrake's implementation of PCD preconditioning is limited to steady flow problems, we focus on testing our implementation of PCD preconditioning, with emphasis on comparing the performance of iterative solvers using our implementation of PCD preconditioning, iterative solvers using LSC preconditioning and MUMPS (Amestoy et al. [2001] and Amestoy et al. [2019]) for both steady and unsteady cases in two- and three-dimensional space.

1. Saddle-point system

In this chapter we focus on saddle-point systems. This type of linear systems arise in a large number of applications of computational science and engineering, for example, the (linearized) Navier-Stokes equations and other physical problems with conservation laws. Since such systems are typically large and sparse, they are usually solved by iterative methods such as Krylov subspace methods. For accelerating the rate of convergence of Krylov subspace methods, effective preconditioning is needed.

The goal of this chapter is to formally define saddle-point systems, describe their properties and methods of solving. At the end of this chapter we introduce the preconditioning techniques for the saddle-point systems. For more details about saddle-point systems, we refer to the book Rozložník [2018] and to the article Benzi et al. [2005].

1.1 Motivation

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric positive definite matrix and let $\vec{b} \in \mathbb{R}^n$ be given. Consider the following system of linear algebraic equations

$$\mathbf{A}\vec{x} = \vec{b}. \tag{1.1}$$

The problem of solving the system (1.1) is equivalent to the following minimization problem

$$\textit{minimize } \frac{1}{2}\vec{x}^{\mathsf{T}}\mathbf{A}\vec{x} - \vec{b}^{\mathsf{T}}\vec{x}. \tag{1.2}$$

Adding constraints to (1.2) we received the following constrained quadratic optimization problem

$$\begin{aligned} &\textit{minimize } \frac{1}{2}\vec{x}^{\mathsf{T}}\mathbf{A}\vec{x} - \vec{b}^{\mathsf{T}}\vec{x} \\ &\textit{subject to } \mathbf{B}\vec{x} = \vec{c}, \end{aligned} \tag{1.3}$$

where the matrix $\mathbf{B} \in \mathbb{R}^{m\times n},\ n \geq m$, has full row rank and $\vec{c} \in \mathbb{R}^m$ is given. Let us define the corresponding Lagrangian $\mathcal{L}$ for the optimization problem (1.3) as

$$\mathcal{L}(\vec{x}, \vec{y}) = \frac{1}{2}\vec{x}^{\mathsf{T}}\mathbf{A}\vec{x} - \vec{b}^{\mathsf{T}}\vec{x} + (\mathbf{B}\vec{x} - \vec{c})^{\mathsf{T}}\vec{y}. \tag{1.4}$$

The first-order optimality conditions $\nabla_x\ \mathcal{L} = \vec{0},\ \nabla_y\ \mathcal{L} = \vec{0}$ read as

$$\begin{aligned} \vec{0} &= \nabla_x\ \mathcal{L} = \mathbf{A}\vec{x} + \mathbf{B}^{\mathsf{T}}\vec{y} - \vec{b}, \\ \vec{0} &= \nabla_y\ \mathcal{L} = \mathbf{B}\vec{x} - \vec{c} \end{aligned} \iff \begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{c}. \end{pmatrix}. \tag{1.5}$$

System (1.5) is so-called saddle-point system. It can be proved (Propositions 1.2 and 1.3, Rozložník [2018], pages 3-5) that any solution $(\bar{x}, \bar{y})^{\mathsf{T}},\ \bar{x} \in \mathbb{R}^n,\ \bar{y} \in \mathbb{R}^m$ of the the saddle-point system (1.5) is the stationary point of the Lagrangian $\mathcal{L}$ and it satisfies

$$\min_{x\in\mathbb{R}^n}\max_{y\in\mathbb{R}^m}\mathcal{L}(x, y) = \mathcal{L}(\bar{x}, \bar{y}) = \max_{x\in\mathbb{R}^n}\min_{y\in\mathbb{R}^m}\mathcal{L}(x, y). \tag{1.6}$$

Remark 1
The solutions $(\bar{x}, \bar{y})^\mathsf{T}$ *of* (1.5) *are usually called saddle points.*

This is one of examples where saddle-point systems arise. Other problems leading to saddle-point systems come from fluid dynamics, linear elasticity, mixed FEM formulations of 2nd- and 4th-order elliptic PDEs, ... Due to the frequent occurrence of saddle-point systems, the effective methods for their solving are needed. In this thesis we will be interested in saddle-point systems arising from fluid dynamics. Our goal will be to solve such systems by Krylov subspace methods with effective preconditioning.

1.2 Definition

Through this chapter, we will distinguish between two terms - general saddle-point system and saddle-point system. As the name indicates, the saddle-point system is a simplified general saddle-point system. Firstly, let us define the general saddle-point system:

Definition 1 (General saddle-point system)
Let $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ *be real matrix of order* $(n+m)$ *and let* $\vec{b} \in \mathbb{R}^{n+m}$ *be real column vector with* $(n+m)$ *entries, where* $n, m \in \mathbb{N},\ n \geq m$. *We say that the system of linear algebraic equations*

$$\mathcal{A}\vec{x} = \vec{b} \tag{1.7}$$

is ***general saddle-point system****, if* $\mathcal{A}$ *is 2-by-2 block matrix of the form*

$$\mathcal{A} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}_1^\mathsf{T} \\ \boldsymbol{B}_2 & \boldsymbol{C} \end{pmatrix}, \tag{1.8}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B}_1,\ \boldsymbol{B}_2 \in \mathbb{R}^{m\times n}$ *and* $\boldsymbol{C} \in \mathbb{R}^{m\times m}$. *The matrix* $\mathcal{A}$ *is called* ***general saddle-point matrix****.*

Remark 2

1. *Any system of linear algebraic equations can be written as general saddle-point system.*
2. *From the definition* (1) *it is clear that a general saddle-point matrix* $\mathcal{A}$ *is symmetric if and only if* $\boldsymbol{A} = \boldsymbol{A}^\mathsf{T}$, $\boldsymbol{C} = \boldsymbol{C}^\mathsf{T}$ *and* $\boldsymbol{B}_1 = \boldsymbol{B}_2$.

The case when the block $\mathbf{C} = \mathbf{0}$ and the block $\mathbf{B}_1 = \mathbf{B}_2$ is very common and therefore we define this case separately:

Definition 2 (Saddle-point system)
We say that the general saddle-point system $\mathcal{A}\vec{x} = \vec{b}$ *is* ***saddle-point system****, if the block* $\boldsymbol{C} = \boldsymbol{0}$ *and* $\boldsymbol{B}_1 = \boldsymbol{B}_2 = \boldsymbol{B}$, *i.e., the matrix* $\mathcal{A}$ *is of the form*

$$\mathcal{A} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}^\mathsf{T} \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix}. \tag{1.9}$$

In this case, the matrix $\mathcal{A}$ *is called* ***saddle-point matrix****.*

The case when the block $\mathbf{C}$ is not the zero matrix is very important and it arises in the context of stabilized mixed finite element methods. However, we will not consider this case in this thesis.

Remark 3

1. *The problem of finding the solution to the given saddle-point system is usually called saddle-point problem. In the case $\mathcal{A}$ is general saddle-point matrix, the corresponding problem is called general saddle-point problem.*

2. *The symmetry of a saddle-point matrix $\mathcal{A}$ depends only on the symmetry of the block $\boldsymbol{A}$.*

As we will see in the second chapter, the discrete formulations of both the Stokes and (linearized) Navier-Stokes problems lead to a saddle-point systems.

1.3 Basic properties

Before the solution methods for saddle-point systems will be discussed, we turn our attention to study a basic properties of saddle-point matrices such as invertibility, the existence of their factorizations and their spectral properties.

1.3.1 Nonsingularity

Let $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ and $\vec{\mathbf{b}} \in \mathbb{R}^{n+m}$ with $n \geq m$. Consider the saddle-point system

$$\mathcal{A}\vec{x} = \vec{b} \iff \begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix}, \tag{1.10}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite real matrix of order n and the matrix $\mathbf{B} \in \mathbb{R}^{m\times n}$ has full row rank, i.e., $rank(\mathbf{B}) = m$. The following theorem (Theorem 3.2, [Benzi et al., 2005], page 16) establishes necessary and sufficient condition for unique solvability of (1.10):

Theorem 1 (Unique solvability)
Let $\mathcal{A}$ be saddle-point matrix as in (1.10)*. Then*

$$\mathcal{A} \text{ is nonsingular} \iff \mathcal{N}(\boldsymbol{A}) \cap \mathcal{N}(\boldsymbol{B}) = \{\vec{0}\}, \tag{1.11}$$

where $\mathcal{N}$ denotes the null space of a matrix.

Proof. We prove two implications:
$\Rightarrow$: This implication can be proved indirectly. Suppose $\mathcal{N}(\mathbf{A}) \cap \mathcal{N}(\mathbf{B}) \neq \{\vec{0}\}$. Take $\vec{0} \neq \vec{y}_1 \in \mathcal{N}(\mathbf{A}) \cap \mathcal{N}(\mathbf{B})$. Then

$$\mathcal{A}\begin{pmatrix} \vec{y}_1 \\ \vec{0} \end{pmatrix} = \vec{0}$$

and therefore the saddle-point matrix $\mathcal{A}$ is singular.
$\Leftarrow$: Consider $\vec{y} = \begin{pmatrix} \vec{y}_1 \\ \vec{y}_2 \end{pmatrix} \in \mathbb{R}^{n+m}$ such that $\mathcal{A}\vec{y} = \vec{0}$. It is enough to prove that

$\vec{y} = \vec{0}$. The saddle-point system $\mathcal{A}\vec{y} = \vec{0}$ can be equivalently written as

$$\begin{aligned} \mathbf{A}\vec{y}_1 + \mathbf{B}^\mathsf{T}\vec{y}_2 &= \vec{0} \\ \mathbf{B}\vec{y}_1 &= \vec{0}. \end{aligned} \tag{1.12}$$

After pre-multiplying the first equation in (1.12) by $\vec{y}^\mathsf{T}$ and using the second equation in (1.12), we get

$$\vec{y}_1^\mathsf{T}\mathbf{A}\vec{y}_1 = -\vec{y}_1^\mathsf{T}\mathbf{B}^\mathsf{T}\vec{y}_2 = -(\mathbf{B}\vec{y}_1)^\mathsf{T}\vec{y}_2 = \vec{0}.$$

As the matrix $\mathbf{A}$ is symmetric positive semidefinite, it can be written as $\mathbf{A} = \mathbf{A}^{1/2}\mathbf{A}^{1/2}$. Then

$$\vec{y}_1^\mathsf{T}\mathbf{A}\vec{y}_1 = \vec{y}_1^\mathsf{T}\mathbf{A}^{1/2}\mathbf{A}^{1/2}\vec{y}_1 = \vec{0} \tag{1.13}$$

what implies that $\left\|\mathbf{A}^{1/2}\vec{y}_1\right\| = 0$, where $\|\cdot\|$ denotes the standard Euclidean norm. Hence $\mathbf{A}^{1/2}\vec{y}_1 = \vec{0}$ and therefore $\mathbf{A}\vec{y}_1 = \vec{0}$. This together with the second equation in (1.12) implies $\vec{y}_1 \in \mathcal{N}(\mathbf{A}) \cap \mathcal{N}(\mathbf{B})$. As $\mathbf{B}$ has full row rank and $\mathbf{B}^\mathsf{T}\vec{y}_2 = \vec{0}$ from the first equation in (1.12), we get $\vec{y}_2 = \vec{0}$. We have proved that $\vec{y} = \vec{0}$ and thus the saddle-point matrix $\mathcal{A}$ is nonsingular. □

In the case the block $\mathbf{A}$ is symmetric positive definite matrix, the following theorem (Theorem 4.25, Bertaccini [2018], page 232) establishes the necessary condition for nonsingularity of the saddle-point matrix $\mathcal{A}$:

Theorem 2
If the block matrix $\boldsymbol{A}$ is symmetric positive definite and $\boldsymbol{B}$ has full rank, then saddle-point matrix $\mathcal{A}$ is nonsingular.

Proof. See Bertaccini [2018], page 232. □

The assumption of positive definiteness of the matrix $\mathbf{A}$ is needed in the previous theorem: If the matrix $\mathbf{A}$ is of the form

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

then $\mathbf{A}$ is clearly indefinite and the saddle-point matrix

$$\mathcal{A} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & -1 & 1 \\ -1 & 1 & 0 \end{pmatrix},$$

is singular, even if matrix $\mathbf{B} = \begin{pmatrix} -1 & 1 \end{pmatrix}$ has full rank (Example 4.11, Bertaccini [2018], page 232).

For more results on nosingularity in a more general case, we refer to Benzi et al. [2005] and Bertaccini [2018].

1.3.2 Schur complement

Consider the general saddle-point matrix

$$\mathcal{A} = \begin{pmatrix} \mathbf{A} & \mathbf{B}_1^\mathsf{T} \\ \mathbf{B}_2 & \mathbf{C} \end{pmatrix}. \tag{1.14}$$

If the block $\mathbf{A}$ of $\mathcal{A}$ is nonsingular (invertible), then the general saddle-point matrix (1.14) can be factorized as:

$$\mathcal{A} = \begin{pmatrix} \mathbf{A} & \mathbf{B}_1^\mathsf{T} \\ \mathbf{B}_2 & \mathbf{C} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}_2\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} - \mathbf{B}_2\mathbf{A}^{-1}\mathbf{B}_1^\mathsf{T} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B}_1^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{1.15}$$

Remark 4
The factorization (1.15) *is called block triangular factorization.*

Based on the factorization (1.15) it is obvious that the general saddle-point matrix (1.14) is nonsingular if and only if the matrix $\mathbf{C} - \mathbf{B}_2\mathbf{A}^{-1}\mathbf{B}_1^\mathsf{T}$ is nonsingular.

Definition 3 (Schur complement)
Let $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ be general saddle-point matrix of the form as in (1.14)*, where $n, m \in \mathbb{N}$, $n \geq m$. Assume that the block $\boldsymbol{A}$ of $\mathcal{A}$ is nonsingular (invertible). Then the **Schur complement** of the block $\boldsymbol{A}$ in $\mathcal{A}$ is matrix $\boldsymbol{S}$ defined by*

$$\boldsymbol{S} = \boldsymbol{C} - \boldsymbol{B}_2\boldsymbol{A}^{-1}\boldsymbol{B}_1^\mathsf{T}. \tag{1.16}$$

Another commonly used notation for the Schur complement matrix of block $\boldsymbol{A}$ in $\mathcal{A}$ is $\mathcal{A}\backslash\boldsymbol{A}$.

Remark 5

1. *In general, it is more difficult to verify nonsingularity for the Schur complement matrix $\boldsymbol{S}$ than the nonsingularity of general saddle-point matrix* (1.14)*. (Benzi et al. [2005], page 15)*

2. *In the case of saddle-point matrix ($\boldsymbol{C=0}$, $\boldsymbol{B}_1 = \boldsymbol{B}_2 = \boldsymbol{B}$), the Schur complement matrix $\boldsymbol{S}$ of the block $\boldsymbol{A}$ in $\mathcal{A}$ reduces to*

$$\boldsymbol{S} = -\boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^\mathsf{T}. \tag{1.17}$$

What we want to investigate now is **Schur complement method** for solving the following saddle-point system

$$\mathcal{A}\vec{x} = \vec{b} \iff \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix}, \tag{1.18}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\mathbf{B} \in \mathbb{R}^{m\times n}$ has full row rank, i.e., $rank(\mathbf{B}) = m$. Obviously, the matrix $\mathbf{A}$ is nonsingular because it is symmetric positive definite. From the definition of the Schur complement matrix it is clear that the Schur complement matrix $\mathbf{S}$ is nosingular if and only if the matrix $\mathbf{B}^\mathsf{T}$ has full column rank, i.e., $rank(\mathbf{B}^\mathsf{T}) = m$. Additionally, as the matrix $\mathbf{A}$ is symmetric positive definite, the Schur complement matrix $\mathbf{S}$ is symmetric

negative definite and thus nonsingular. Using $rank(\mathbf{B}) = rank(\mathbf{B}^\mathsf{T})$ together with the theorem (2) we get

$$\mathcal{A} \textit{ is nonsingular} \iff \mathbf{S} \textit{ is nonsingular} \iff rank(\mathbf{B}) = m. \tag{1.19}$$

It means that the saddle-point system (1.18) has an unique solution. Now, we derive the Schur complement method. For this purpose, we use the factorization (1.15). As the saddle-point matrix $\mathcal{A}$ of the system (1.18) is nonsingular, we can explicitly write the inverse of $\mathcal{A}$ as

$$\begin{aligned} \mathcal{A}^{-1} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix}^{-1} = \\ = \begin{pmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\mathbf{S}^{-1}\mathbf{B}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\mathbf{B}\mathbf{A}^{-1} & \mathbf{S}^{-1} \end{pmatrix}^{-1} \end{aligned} \tag{1.20}$$

Then we can explicitly express the solution $\vec{x}$ of the saddle-point system (1.18) as

$$\begin{aligned} \vec{x} = \begin{pmatrix} \vec{x_1} \\ \vec{x_2} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} \vec{b_1} \\ \vec{b_2} \end{pmatrix} = \\ = \begin{pmatrix} (\mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\mathbf{S}^{-1}\mathbf{B}\mathbf{A}^{-1})\vec{b_1} - \mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\mathbf{S}^{-1}\vec{b_2} \\ -\mathbf{S}^{-1}\mathbf{B}\mathbf{A}^{-1}\vec{b_1} + \mathbf{S}^{-1}\vec{b_2} \end{pmatrix} \end{aligned} \tag{1.21}$$

Let us first assume that $\vec{b_2} = 0$. Then (1.21) reduces to

$$\vec{x} = \begin{pmatrix} \vec{x_1} \\ \vec{x_2} \end{pmatrix} = \begin{pmatrix} (\mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\mathbf{S}^{-1}\mathbf{B}\mathbf{A}^{-1})\vec{b_1} \\ -\mathbf{S}^{-1}\mathbf{B}\mathbf{A}^{-1}\vec{b_1} \end{pmatrix} \tag{1.22}$$

In this case the vector $\vec{x_2}$ is the solution of the system

$$\mathbf{S}\vec{x_2} = -\mathbf{B}\mathbf{A}^{-1}\vec{b_2} \iff \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\vec{x_2} = \mathbf{B}\mathbf{A}^{-1}\vec{b_1}, \tag{1.23}$$

which is the system with the symmetric positive definite matrix $\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\mathsf{T}$ as the Schur complement matrix $\mathbf{S}$ is symmetric negative definite. If the vector $\vec{x_2}$ has been computed from (1.23), then we can compute the vector $\vec{x_1}$ as

$$\vec{x_1} = \mathbf{A}^{-1}(\vec{b_1} - \mathbf{B}^\mathsf{T}\vec{x_2}) \tag{1.24}$$

Thus for the right-hand side vector $\vec{b} = (\vec{b_1}\ \vec{0})^\mathsf{T}$ we can express the solution $\vec{x}$ of (1.18) as

$$\vec{x} = \begin{pmatrix} \mathbf{A}^{-1}(\vec{b_1} - \mathbf{B}^\mathsf{T}\vec{x_2}) \\ -\mathbf{S}^{-1}\mathbf{B}\mathbf{A}^{-1}\vec{b_1} \end{pmatrix}. \tag{1.25}$$

If $\vec{b_2} \neq 0$ in (1.21), then the vector $\vec{x_2}$ is the solution of the system

$$\mathbf{S}\vec{x_2} = \vec{b_2} - \mathbf{B}\mathbf{A}^{-1}\vec{b_2}. \tag{1.26}$$

Once the solution $\vec{x_2}$ has been computed from (1.26), the vector $\vec{x_1}$ is determined by (1.24). This method is called **Schur complement method**. For more details about Schur complement method, see Rozložník [2018], pages 34-36.

Instead of (1.15), we can consider the equivalent **LU** factorization of the saddle-point matrix $\mathcal{A}$ of the system (1.18):

$$\mathcal{A} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}. \tag{1.27}$$

Then the saddle-point system (1.18) can be written as

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 - \mathbf{B}\mathbf{A}^{-1}\vec{b}_1 \end{pmatrix}, \tag{1.28}$$

which is block upper triangular system. It can be solved by block backward substitution. This leads to solving two systems - (1.24) and (1.26). For solving these systems, we can use direct method - see subsection 1.4.1. In the case when forming or factorization of Schur complement **S** is too expensive, we can use iterative methods (for example conjugate gradient method (CG)), which need Schur complement matrix **S** only in the form of matrix-vector products. Another possible problem is ill-conditioning of (1.26). It can be solved by preconditioning. However, if we use iterative methods, we do not need explicitly the Schur complement **S** and therefore it is difficult to find suitable preconditioner for the system (1.26) in general. It can be solved by approximating of the Schur complement matrix **S** as we will see later.

1.3.3 Spectral properties

In this part we resume briefly the spectral properties of the saddle-point systems. Let $\mathcal{A}$ be saddle-point matrix as in (1.18). The factorization (1.15) of the saddle-point matrix $\mathcal{A}$ can be rewritten as

$$\begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{1.29}$$

It implies that $\mathcal{A}$ is congruent to the block diagonal matrix $\begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}$, i. e., there exists invertible matrix $\mathcal{Q}$, in our case $\mathcal{Q} = \begin{pmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$, such that $\begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} = \mathcal{Q}\mathcal{A}\mathcal{Q}^\mathsf{T}$. Hence, using Sylvester's law of inertia, the saddle-point matrix $\mathcal{A}$ is indefinite with n positive and m negative eigenvalues as $\mathbf{S} \in \mathbb{R}^{m\times m}$ is symmetric negative definite and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric positive definite.

The following theorem (Proposition 3.5, Rozložník [2018], page 28) gives us eigenvalue bounds for saddle-point matrix $\mathcal{A}$:

Theorem 3 (Eigenvalue bounds)
Let $\mathcal{A} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}^\mathsf{T} \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$ *be a saddle-point matrix, where* $\boldsymbol{A}$ *is symmetric positive definite and* $\boldsymbol{B}$ *has full row rank. Let* $0 < \lambda_1(\boldsymbol{A}) \leq ... \leq \lambda_n(\boldsymbol{A})$ *be eigenvalues of* $\boldsymbol{A}$ *and let* $0 < \sigma_1(\boldsymbol{B}) \leq ... \leq \sigma_m(\boldsymbol{B})$ *be singular values of* $\boldsymbol{B}$*. Then*

$$sp(\mathcal{A}) \subset E^+ \cup E^-, \tag{1.30}$$

where

$$E^{-} = \left[\frac{1}{2}\left(\lambda_1(\boldsymbol{A}) - \sqrt{\lambda_1^2(\boldsymbol{A}) + 4\sigma_m^2(\boldsymbol{B})}\right),\ \frac{1}{2}\left(\lambda_n(\boldsymbol{A}) - \sqrt{\lambda_n^2(\boldsymbol{A}) + 4\sigma_1^2(\boldsymbol{B})}\right)\right]$$
$$E^{+} = \left[\lambda_1(\boldsymbol{A}),\ \frac{1}{2}\left(\lambda_n(\boldsymbol{A}) + \sqrt{\lambda_n^2(\boldsymbol{A}) + 4\sigma_m^2(\boldsymbol{B})}\right)\right].$$

Proof. See Rozložník [2018], pages 28-29. □

The theorem (3) implies that if $\lambda_n(\mathbf{A})$ and $\sigma_m(\mathbf{B})$ are kept constant, then the condition number

$$\kappa(\mathcal{A}) = \frac{|\lambda_n(\mathcal{A})|}{|\lambda_1(\mathcal{A})|}$$

of the saddle-point matrix $\mathcal{A}$ satisfies either

$$\lim_{\lambda_1(\mathbf{A})\to 0} \kappa(\mathcal{A}) = \infty$$

or

$$\lim_{\sigma_1(\mathbf{B})\to 0} \kappa(\mathcal{A}) = \infty,$$

meaning that $\kappa(\mathcal{A})$ is not bounded. For mixed finite element formulations of elliptic PDEs, both $\lambda_1(\mathbf{A})$ and $\sigma_1(\mathbf{B})$ go to zero as the mesh size parameter h goes to zero (Benzi et al. [2005], page 27). Thus $\kappa(\mathcal{A})$ grows like $O(h^{-p})$ for some positive value of p, which implies substantial problem of the rate of convergence of iterative methods as the size of the problem increases. One of possible techniques, how to reduce or even eliminate the dependency on the mesh size parameter h, is preconditioning, which will be discussed later. For further details, see Benzi et al. [2005] page 27-28.

1.4 Direct and iterative methods

This section is devoted to direct and iterative methods for solving large and sparse saddle-point systems. Emphasis will be placed on Krylov subspace methods which are attractive for solving such a systems.

1.4.1 Direct methods

In general, direct methods for solving the system of linear equations are methods based on factorization (variant of Gaussian elimination) of coefficient matrix of this system. Typical factorizations are for instance $\mathbf{LU}$ factorization, Cholesky factorization $\mathbf{LL}^{\mathsf{T}}$ for symmetric positive definite matrices and $\mathbf{LDL}^{\mathsf{T}}$ factorization.

Now we present application of direct methods for solving saddle-point systems. For this purpose, consider the following saddle-point system

$$\mathcal{A}\vec{x} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \vec{\mathbf{b}} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix}, \tag{1.31}$$

where the block $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric positive definite matrix and $\mathbf{B} \in \mathbb{R}^{m\times n}$ has full row rank, $n,\ m \in \mathbb{N},\ n \geq m$. Let us show direct methods based on block $\mathbf{LDL}^{\mathsf{T}}$ factorization and $\mathbf{LU}$ factorization for (1.31):

Block $\mathbf{LDL}^\mathsf{T}$ factorization

Let us consider the block $\mathbf{LDL}^\mathsf{T}$ factorization of the saddle-point matrix $\mathcal{A}$

$$\mathcal{A} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{1.32}$$

where $\mathbf{L} \in \mathbb{R}^{(n+m)\times(n+m)}$ is unit lower triangular matrix and $\mathbf{D} \in \mathbb{R}^{(n+m)\times(n+m)}$ is block diagonal matrix with diagonal blocks of dimension 1 or 2. As the block $\mathbf{A}$ of $\mathcal{A}$ is symmetric positive definite, Cholesky factorization of $\mathbf{A}$ reads as

$$\mathbf{A} = \mathbf{L}_A\mathbf{L}_A^\mathsf{T},$$

where $\mathbf{L}_A$ is lower triangular matrix with positive diagonal entries. Then (1.32) can be written as

$$\mathcal{A} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{L}_A^{-\mathsf{T}}\mathbf{L}_A & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{L}_A\mathbf{L}_A^\mathsf{T} & \mathbf{0} \\ \mathbf{0} & -\mathbf{B}\mathbf{L}_A^{-\mathsf{T}}\mathbf{L}_A\mathbf{B}^\mathsf{T} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L}_A^{-\mathsf{T}}\mathbf{L}_A\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{1.33}$$

Since $\mathbf{A}$ is symmetric positive definite, the Schur complement matrix $\mathbf{S}$ is symmetric negative definite and therefore the matrix

$$-\mathbf{S} = (\mathbf{L}_A^{-1}\mathbf{B}^\mathsf{T})^\mathsf{T}(\mathbf{L}_A^{-1}\mathbf{B}^\mathsf{T}) \tag{1.34}$$

is symmetric positive definite. Let

$$-\mathbf{S} = \mathbf{L}_S\mathbf{L}_S^\mathsf{T}, \tag{1.35}$$

be Cholesky factorization of the matrix $(-\mathbf{S})$. Then (1.33) can be written as

$$\mathcal{A} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T} = \begin{pmatrix} \mathbf{L}_A & \mathbf{0} \\ (\mathbf{L}_A^{-1}\mathbf{B}^\mathsf{T})^\mathsf{T} & \mathbf{L}_S \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{L}_A^\mathsf{T} & \mathbf{L}_A^{-1}\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{L}_S^\mathsf{T} \end{pmatrix}. \tag{1.36}$$

Based on (1.36), the saddle-point system (1.31) can be written as

$$\begin{pmatrix} \mathbf{L}_A^\mathsf{T} & \mathbf{L}_A^{-1}\mathbf{B}^\mathsf{T} \\ \mathbf{0} & -\mathbf{L}_S^\mathsf{T} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{L}_A^{-1}\vec{b}_1 \\ \mathbf{L}_S^{-1}(\vec{b}_2 - (\mathbf{L}_A^{-1}\mathbf{B}^\mathsf{T})^\mathsf{T}(\mathbf{L}_A^{-1}\vec{b}_1)) \end{pmatrix}. \tag{1.37}$$

It is a system with upper triangular matrix which can be solved by backward substitution. For more details, we refer to Rozložník [2018], pages 45-46.

LU factorization

Assume that the $\mathbf{LU}$ factorization of block $\mathbf{B}$ of $\mathcal{A}$ is of the form

$$\mathbf{B} = \mathbf{L}\begin{pmatrix} \mathbf{U} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_1 & \mathbf{0} \\ \mathbf{L}_2 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{U} \\ \mathbf{0} \end{pmatrix}, \tag{1.38}$$

where $\mathbf{L} \in \mathbb{R}^{m\times m}$ is lower triangular matrix and $\mathbf{U} \in \mathbb{R}^{n\times n}$ is upper triangular matrix. Then the saddle-point matrix $\mathcal{A}$ can be factorized as

$$\mathcal{A} = \begin{pmatrix} \mathbf{U}^\mathsf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{L} \end{pmatrix} \begin{pmatrix} \mathbf{U}^{-\mathsf{T}}\mathbf{A}\mathbf{U}^{-1} & \begin{pmatrix} \mathbf{I} & \mathbf{0} \end{pmatrix} \\ \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^\mathsf{T} \end{pmatrix}. \tag{1.39}$$

After substituting (1.39) into the saddle-point system (1.31) and some algebraic manipulation we find that

$$\begin{pmatrix} \mathbf{U}^{-\mathsf{T}}\mathbf{A}\mathbf{U}^{-1} & \begin{pmatrix}\mathbf{I} & \mathbf{0}\end{pmatrix} \\ \begin{pmatrix}\mathbf{I} \\ \mathbf{0}\end{pmatrix} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{U}\vec{x}_1 \\ \mathbf{L}^{\mathsf{T}}\vec{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{U}^{-\mathsf{T}}\vec{b}_1 \\ \mathbf{L}^{-1}\vec{b}_2 \end{pmatrix}, \tag{1.40}$$

what is equivalent to the block lower triangular system

$$\begin{pmatrix} \begin{pmatrix}\mathbf{I} \\ \mathbf{0}\end{pmatrix} & \mathbf{0} \\ \mathbf{U}^{-\mathsf{T}}\mathbf{A}\mathbf{U}^{-1} & \begin{pmatrix}\mathbf{I} & \mathbf{0}\end{pmatrix} \end{pmatrix} \begin{pmatrix} \mathbf{U}\vec{x}_1 \\ \mathbf{L}^{\mathsf{T}}\vec{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{L}^{-1}\vec{b}_2 \\ \mathbf{U}^{-\mathsf{T}}\vec{b}_1 \end{pmatrix}.$$

This system can be solved by forward substitution. Another direct methods for solving saddle-point systems can be found in Rozložník [2018], chapter 5, and in Benzi et al. [2005], pages 29-43.

1.4.2 Iterative methods

We are now going to look at the iterative methods for solving saddle-point system $\mathcal{A}\vec{x} = \vec{b}$. Let $\vec{x}_0$ be given initial approximation of the solution $\vec{x}$. The idea of iterative methods is to use $\vec{x_0}$ to generate a sequence of approximations $\vec{x}_1,\ \vec{x}_2,\ \vec{x}_3,\ ...$ that converge to the exact solution. For more details about iterative methods we refer to Saad [2003].

The iterative methods can be classified into 2 main classes:

1. Stationary iterative methods
2. Krylov subspace methods

Nowadays, stationary iterative methods are usually used as a smoother for multigrid methods or as preconditioners of Krylov subspace methods. As stationary methods are not subject of our interest, we exclude the theory of stationary iterative methods in this thesis. For more details about stationary iterative methods, we refer to Saad [2003], chapter 4, and Benzi et al. [2005], chapter 8.

Before Krylov subspace methods are discussed, let us introduce **projection process** as in Liesen and Strakoš [2016]:

Let us consider the saddle-point system $\mathcal{A}\vec{x} = \vec{b}$ with $\mathcal{A} \in \mathbb{R}^{N\times N}$ and $\vec{b} \in \mathbb{R}^N$. Assume that in the iterative process the n-th approximation $\vec{x}_n,\ n = 1, 2, ...$ is of the form

$$\vec{x}_n \in \vec{x}_0 + \mathcal{S}_n, \tag{1.41}$$

where $\mathcal{S}_n \subset \mathbb{R}^N$ is n-dimensional subspace and $\vec{x}_0 \in \mathbb{R}^N$ is initial approximation. The subspace $\mathcal{S}_n$ is called **search space**. Since $\mathcal{S}_n$ has dimension n, we have n degrees of freedom to construct $\vec{x}_n$. Thus for determining the approximation $\vec{x}_n$ we need n constraints in general. For this purpose, let us denote $\vec{r}_n = \vec{b} - \mathcal{A}\vec{x}_n \in \vec{r}_0 + \mathcal{A}\mathcal{S}_n$ the n-th residual corresponding to the n-th approximation $\vec{x}_n$. The following orthogonal condition will be required

$$\vec{r}_n \perp \mathcal{C}_n, \tag{1.42}$$

where $\mathcal{C}_n \subset \mathbb{R}^N$ is n-dimensional subspace called **constraint space**. The conditions (1.41)-(1.42) form projection process.

If the columns of the matrix $\mathbf{S}_n \in \mathbb{R}^{N\times n}$ form basis of the searching space $\mathcal{S}_n$, then the condition (1.41) can be interpreted as

$$\vec{x}_n = \vec{x}_0 + \mathbf{S}_n\vec{t}_n$$

for some $\vec{t}_n \in \mathbb{R}^n$. For determining $\vec{t}_n$ we use the condition (1.42). Assume that the columns of the matrix $\mathbf{C}_n \in \mathbb{R}^{N\times n}$ form basis of the constraint space $\mathcal{C}_n$. Then the condition (1.42) reads as

$$\vec{0} = \mathbf{C}_n^*\vec{r}_n = \mathbf{C}_n^*\vec{r}_0 - \mathbf{C}_n^*\mathcal{A}\mathbf{S}_n\vec{t}_n \iff \mathbf{C}_n^*\mathcal{A}\mathbf{S}_n\vec{t}_n = \mathbf{C}_n^*\vec{r}_0.$$

It means that for determining the n-th approximation $\vec{x}_n$ we need to solve so-called projected system

$$\mathbf{C}_n^*\mathcal{A}\mathbf{S}_n\vec{t}_n = \mathbf{C}_n^*\vec{r}_0,$$

with the matrix $\mathbf{C}_n^*\mathcal{A}\mathbf{S}_n$ of order n.

Definition 4 (Well-defined projection process)
*We say that the projection process is **well-defined** at step n, if the projected matrix $\boldsymbol{C}_n^*\mathcal{A}\boldsymbol{S}_n$ is nonsingular.*

If the projection process (1.41)-(1.42) is well defined, then the n-th approximation $\vec{x}_n$ is a solution of saddle-point system $\mathcal{A}\vec{x} = \vec{b}$ if and only if $\vec{r}_n = \vec{b} - \mathcal{A}\vec{x}_n = \vec{0}$. It means that the sufficient condition for termination of the projection process at step n is $\vec{r}_n = \vec{0}$.

Lemma 1 (Lemma 2.1.7, Liesen and Strakoš [2016], page 18)
Assume that the projection process (1.41)-(1.42) *is well-defined at step n. If $\vec{r}_0 \in \mathcal{S}_n$ and $\mathcal{A}\mathcal{S}_n = \mathcal{S}_n$, then $\vec{r}_n = 0$.*

Proof. See Liesen and Strakoš [2016], pages 18-19. □

This property is known as finite termination property.

1.4.3 Krylov subspace methods

Now we turn our attention to Krylov subspace methods. Consider the saddle-point system

$$\mathcal{A}\vec{x} = \vec{b}, \tag{1.43}$$

where $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ and $\vec{b} \in \mathbb{R}^{n+m}$. Let $\vec{x}_0 \in \mathbb{R}^{n+m}$ be an initial approximation of the solution $\vec{x} \in \mathbb{R}^{n+m}$ of (1.43) and let $\vec{r}_0 = \vec{b} - \mathcal{A}\vec{x}_0$ be initial residual. The idea of Krylov subspace methods is to build up a nested sequence

$$\mathcal{K}_1(\mathcal{A}, \vec{r}_0) \subset \mathcal{K}_2(\mathcal{A}, \vec{r}_0) \subset \mathcal{K}_3(\mathcal{A}, \vec{r}_0) \subset ..., \tag{1.44}$$

of Krylov subspaces

$$\mathcal{K}_k(\mathcal{A}, \vec{r}_0) = span\{\vec{r}_0, \mathcal{A}\vec{r}_0, \mathcal{A}^2\vec{r}_0, ..., \mathcal{A}^{k-1}\vec{r}_0\}, \;\; k=1,2,..., \tag{1.45}$$

where $\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$ denotes the k-th Krylov subspace of dimension $k \in \mathbb{N}$, such that the k-th approximation $\vec{x}_k$ satisfies

$$\vec{x}_k \in \vec{x}_0 + \mathcal{K}_k(\mathcal{A}, \vec{r}_0), \tag{1.46}$$

$$\vec{r}_k = \vec{b} - \mathcal{A}\vec{x}_k \perp \mathcal{C}_k. \tag{1.47}$$

Here, $\mathcal{C}_k$ denotes the constraint space of dimension k. Conditions (1.46)-(1.47) are called **Petrov-Galerkin conditions**.

Remark 6
If $\vec{y} \in \mathcal{K}_k(\mathcal{A}, \vec{r}_0)$, then

$$\vec{y} = \sum_{i=0}^{k-1} \alpha_i \mathcal{A}^i \vec{r}_0 \tag{1.48}$$

for some coefficients $\alpha_i,\ i = 1, .., k-1$. Equivalently we can write

$$\vec{y} = p_{k-1}(\mathcal{A})\vec{r}_0, \tag{1.49}$$

where $p_{k-1}(z) = \sum_{i=0}^{k-1} \alpha_i z^i$ is the real polynomial of degree $k-1$.

It is well known that there exits a uniquely defined $d \in \mathbb{N}$ such that the vectors $\vec{r}_0,\ \mathcal{A}\vec{r}_0,\ ...,\ \mathcal{A}^{d-1}\vec{r}_0$ are linearly independent and the vectors $\vec{r}_0,\ \mathcal{A}\vec{r}_0,\ ...,\ \mathcal{A}^{d}\vec{r}_0$ are linearly dependent. The number d is referred as the grade of the vector $\vec{r}_0$ with respect to $\mathcal{A}$. From this it follows that

$$\mathcal{A}\mathcal{K}_d(\mathcal{A}, \vec{r}_0) \subseteq \mathcal{K}_d(\mathcal{A}, \vec{r}_0). \tag{1.50}$$

Remark 7 (Lemma 2.2.2, Liesen and Strakoš [2016], page 21)
If $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ is nonsingular, then $\mathcal{A}\mathcal{K}_d(\mathcal{A}, \vec{r}_0) = \mathcal{K}_d(\mathcal{A}, \vec{r}_0)$.

Also note that it follows directly from the definition of $\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$ that

$$dim\ \mathcal{K}_k(\mathcal{A}, \vec{r}_0) = k, \tag{1.51}$$

for $k = 1, .., d$. Thus the nested sequence (1.44) grows only to dimension d and then it becomes invariant under $\mathcal{A}$, i.e.,

$$\mathcal{K}_1(\mathcal{A}, \vec{r}_0) \subset \mathcal{K}_2(\mathcal{A}, \vec{r}_0) \subset ... \subset \mathcal{K}_d(\mathcal{A}, \vec{r}_0) = \mathcal{K}_{d+i}(\mathcal{A}, \vec{r}_0),\ \textit{for all } i \geq 1. \tag{1.52}$$

This together with the Lemma (1) give the following result:

Lemma 2 (Theorem 2.2.3, Liesen and Strakoš [2016], page 21)
Let d be the grade of $\vec{r}_0$ with respect to $\mathcal{A}$. Assume that the projection process (1.41)-(1.42) *is well-defined at step d and let the search spaces be given by the Krylov subspaces $\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$. If $\mathcal{A}$ is nonsingular, then $\vec{r}_d = \vec{0}$.*

Proof. The Remark (7) implies $\mathcal{A}\mathcal{K}_d(\mathcal{A}, \vec{r}_0) = \mathcal{K}_d(\mathcal{A}, \vec{r}_0)$. Using Lemma (1) we obtain $\vec{r}_d = \vec{0}$.

□

Based on this lemma we know, that if the nested sequence of Krylov subspaces has become invariant under $\mathcal{A}$, then we obtain a zero residual and the process terminates.

In order to obtain uniquely defined approximations $x_k,\ k = 1, 2, ...$ satisfying Petrov-Galerkin conditions, we need to choose appropriate constraint spaces $\mathcal{C}_k$. Knowing some properties of the saddle-point matrix $\mathcal{A}$, we can use the following theorem:

Theorem 4 (Theorem 2.3.1, Liesen and Strakoš [2016], pages 23-24)
Assume that d is the grade of the initial residual vector $\vec{r}_0$ with respect to $\mathcal{A}$ and $\mathcal{S}_n = \mathcal{K}_k(\mathcal{A}, \vec{r}_0)$. Then the following hold:

1. *If $\mathcal{A}$ is symmetric positive definite and $\mathcal{C}_k = \mathcal{K}_k(\mathcal{A}, \vec{r}_0)$, $k = 1, 2, ...$, then the projection process* (1.41)-(1.42) *is well-defined at every step k until it terminates at step d. Hence there exist a uniquely defined approximations $\vec{x}_1, ..., \vec{x}_d$ satisfying Petrov-Galerkin conditions:*

$$\begin{aligned} \vec{x}_k &\in \vec{x}_0 + \mathcal{K}_k(\mathcal{A}, \vec{r}_0) \\ \vec{r}_k &= \vec{b} - \mathcal{A}\vec{x}_k \perp \mathcal{K}_k(\mathcal{A}, \vec{r}_0), \end{aligned} \tag{1.53}$$

   for $k = 1, .., d$. Equivalent optimality property is

$$\|\vec{x} - \vec{x}_k\|_{\mathcal{A}} = \min_{\vec{y} \in \vec{x}_0 + \mathcal{K}_k(\mathcal{A}, \vec{r}_0)} \|\vec{x} - \vec{y}\|_{\mathcal{A}} \,. \tag{1.54}$$

   Relation (1.54) *is mathematical characterization of the conjugate gradient (CG) method.*

2. *If $\mathcal{A}$ is nonsingular and $\mathcal{C}_k = \mathcal{A}\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$, $k = 1, 2, ...$, then the projection process* (1.41)-(1.42) *is well-defined at every step k until it terminates at step d. Hence there exists a uniquely defined approximations $\vec{x}_1, ..., \vec{x}_d$ satisfying Petrov-Galerkin conditions:*

$$\begin{aligned} \vec{x}_k &\in \vec{x}_0 + \mathcal{K}_k(\mathcal{A}, \vec{r}_0) \\ \vec{r}_k &= \vec{b} - \mathcal{A}\vec{x}_k \perp \mathcal{A}\mathcal{K}_k(\mathcal{A}, \vec{r}_0), \end{aligned} \tag{1.55}$$

   for $k = 1, .., d$. Equivalent optimality property is

$$\|\vec{r}_k\| = \min_{\vec{y} \in \vec{x}_0 + \mathcal{K}_k(\mathcal{A}, \vec{r}_0)} \left\|\vec{b} - \mathcal{A}\vec{y}\right\| \tag{1.56}$$

$$\|\vec{x} - \vec{x}_k\|_{\mathcal{A}^*\mathcal{A}} = \min_{\vec{y} \in \vec{x}_0 + \mathcal{K}_k(\mathcal{A}, \vec{r}_0)} \|\vec{x} - \vec{y}\|_{\mathcal{A}^*\mathcal{A}} \,. \tag{1.57}$$

   Relation (1.56) *is mathematical characterization of the MINRES method and* (1.57) *is mathematical characterization of the GMRES method.*

Proof. See Liesen and Strakoš [2016], page 24-25.

□

There exists some other choices of the constraint spaces $\mathcal{C}_k$. For example, one can choose $\mathcal{C}_k = \mathcal{K}_k(\mathcal{A}^\mathsf{T}, \vec{r}_0)$. If the matrix $\mathcal{A}$ is symmetric, then we can apply Theorem (4). The case when $\mathcal{A}$ is nonsymmetric leads to biorthogonalization methods like biconjugate gradient method (BiCG).

The choice of bases for $\mathcal{S}_k$ and $\mathcal{C}_k$ is very important because of numerical stability in finite precision. Therefore we will be interested in generating orthogonal bases for the Krylov subspaces. If matrix $\mathcal{A}$ is symmetric, then we can use **Lanczos algorithm** which generates an orthogonal basis of $\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$ using *three-term recurrences*. Krylov subspace methods like CG or MINRES use this effective algorithm for generating orthogonal basis. If matrix $\mathcal{A}$ is nonsymmetric, then we can use **Arnoldi algorithm** which generates an orthogonal basis of $\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$ using *long-term recurrences*. Krylov subspace methods like GMRES use this algorithm for generating orthogonal basis. For detailed discussion we refer to Saad [2003], section 6.3 and 6.6.

Remark 8
Both Lanczos and Arnoldi algorithms are based on Gram-Schmidt orthogonalisation process.

In the case the matrix $\mathcal{A}$ is nonsymmetric, nonorthogonal basis can sometimes lead to efficient methods. If we relax the orthogonality requirement, we can generate two sets of mutually biorthogonal basis vectors using the **Lanczos biorthogonalisation algorithm**. This algorithm uses three-term recurrences and it is used by Krylov subspace methods like BiCG or QMR. For details we refer to Saad [2003], sections 7.1-7.3. As we see, if one wants to generate the basis of the Krylov subspace $\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$, one has to choose between orthogonality and long-term recurrences.

Method	$\mathcal{C}_k$	$\mathcal{A}$	Basis	Algorithm
CG	$\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$	symm. pos. def.	orth.	Lanczos
FOM	$\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$	nonsymm.	orthogonal	Arnoldi
MINRES	$\mathcal{A}\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$	symm. indefinite	orth.	Lanczos
GMRES	$\mathcal{A}\mathcal{K}_k(\mathcal{A}, \vec{r}_0)$	nonsymm.	orth.	Arnoldi
BiCG	$\mathcal{K}_k(\mathcal{A}^\mathsf{T}, \vec{r}_0)$	nonsymm.	non-orth.	Lanczos biorth.

Table 1.1: Overview of selected Krylov subspace methods.

Remark 9
There are some other Krylov subspace methods like BiCGStab, CGNR, CGNE, TFQMR, SYMMLQ that differ by selecting a search space $\mathcal{S}_n$ and the procedure of computing new approximation $\vec{x}_n$. A detailed discussion of Krylov subspace method can be found in Saad [2003].

1.4.4 Solvers

Let us compare solvers for solving systems of linear equations. The classification of solvers depends on the methods of getting the solution of the linear system. We distinguish between direct and iterative solver. By direct solver is meant a solver using some direct method to solve the system of linear equations. Analogically, the iterative solver is a solver that uses an iterative method (Krylov subspace method) to solve a system of linear equations.

Direct solvers are very robust and effective for solving sufficiently small linear systems in practice. However, if we solve a large system (typically from some 3D models), there are a couple of problems:

1. if the matrix of order n is very large (in the order of a few millions), then n^3 operations are needed to solve the corresponding linear system using Gaussian elimination
2. memory complexity is too prohibitive (n^2 for **LU** factorization)
3. **LU** factorization may introduce fill-in.
4. sometimes we do not need to solve the system exactly (nonlinear systems)
5. direct solvers are difficult to parallelize

In contrast to the direct solvers, iterative solvers are suitable for solving large problems as they can often be parallelized more efficiently. The disadvantage of iterative solvers is slow rate of the convergence of iterative methods. It can be improved by preconditioning.

1.5 Preconditioning

In the previous section we have concluded that iterative solvers, which are based on iterative methods, are more suitable for solving large systems of linear equations. However, the rate of convergence of iterative methods such as Krylov subspace methods is slow. This cause can be remedied by preconditioning.

Definition 5 (Preconditioning)
Let us consider the system of linear equations

$$\mathcal{A}\vec{x} = \vec{b}. \tag{1.58}$$

By ***preconditioning*** *of the system* (1.58) *we mean the transformation of this system into another system with more favourable properties. We know three types of preconditioning:*

1. *left preconditioning*
$$\mathcal{P}^{-1}\mathcal{A}\vec{x} = \mathcal{P}^{-1}\vec{b}. \tag{1.59}$$
2. *right preconditioning*
$$\mathcal{A}\mathcal{P}^{-1}\mathcal{P}\vec{x} = \vec{b}. \tag{1.60}$$
3. *two-sided preconditioning*
$$\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1}\mathcal{P}_2\vec{x} = \mathcal{P}_1^{-1}\vec{b}. \tag{1.61}$$

The systems (1.59)- (1.61) *are called* ***preconditioned systems*** *and the matrices* $\mathcal{P},\ \mathcal{P}_1,\ \mathcal{P}_2$ *are called* ***preconditioners****.*

Remark 10
The system (1.60) *can be written as*

$$\mathcal{A}\mathcal{P}^{-1}\vec{y} = \vec{b},\ \ \vec{x} = \mathcal{P}^{-1}\vec{y}. \tag{1.62}$$

A similar relation holds for system (1.61)*.*

The idea of preconditioning is to construct preconditioner $\mathcal{P}$ such that the preconditioned system may be easier to solve. We focus on constructing the effective preconditioners for saddle-point systems. When the preconditioner is constructed, Krylov subspace method is applied for solving the preconditioned system. We require the following three conditions for effective preconditioner $\mathcal{P}$:

1. the convergence of Krylov subspace method applied to preconditioned system must be rapid
2. evaluating $\vec{c} = \mathcal{P}^{-1}\vec{b}$ must be inexpensive

3. $\mathcal{P}^{-1}$ must be easily computable

Which preconditioner is the best for given saddle-point system? The answer to this question is that it depends on the system itself. Consequently, we have not one universal preconditioner for all saddle-point systems and therefore we have to use different techniques to construct an appropriate preconditioner. We will show constructions of 2 effective preconditioners for saddle-point systems inspired by Elman [2014], sections 4.1, 4.2 and 9.2.

1.5.1 Block diagonal preconditioner

Our aim is to construct effective preconditioner for the saddle-point system

$$\mathcal{A}\vec{x} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \vec{\mathbf{b}} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix}, \tag{1.63}$$

where the block $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\mathbf{B} \in \mathbb{R}^{m\times n}$ has full row rank, $n, m \in \mathbb{N},\ n \geq m$.

Let

$$\mathcal{A} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{1.64}$$

be block $\mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}$ factorization of saddle-point matrix $\mathcal{A}$ as in (1.32). Since $\mathbf{A}$ is symmetric positive definite and $\mathbf{S}$ is symmetric negative definite, saddle-point matrix $\mathcal{A}$ is symmetric indefinite, i.e., $\mathcal{A}$ has both positive and negative eigenvalues.

Indefiniteness of saddle-point matrix $\mathcal{A}$ is the reason why we are not able to use CG method - CG method is not robust in this case. A good choice of robust Krylov subspace method for solving symmetric indefinite systems is MINRES method. Based on the theorem (4) we know that the MINRES method computes approximations $\vec{x}_k,\ k = 1, 2, ...$ of the exact solution $\vec{x}$ of the saddle-point system (1.63) satisfied Petrov-Galenrkin condition as in (1.55). Moreover, applying the remark (6) we can write

$$\begin{aligned} \vec{x} - \vec{x}_k &= \vec{x} - \vec{x}_0 - p_{k-1}(\mathcal{A})\vec{r}_0 = \vec{x} - \vec{x}_0 - p_{k-1}(\mathcal{A})(\mathcal{A}\vec{x} - \mathcal{A}\vec{x}_0) \\ &= (\mathcal{I} - \mathcal{A}p_{k-1}(\mathcal{A}))(\vec{x} - \vec{x}_0) = q_k(\mathcal{A})(\vec{x} - \vec{x}_0), \end{aligned} \tag{1.65}$$

where $q_k(z) = 1 - zp_{k-1}(z),\ q_k \in \Pi_k$, Π_k is the set of all polynomials of degree at most k, and therefore the optimality property (1.56) can be equivalently written as

$$\|\vec{r}_k\| = \min_{\vec{y}\in\vec{x}_0+\mathcal{K}_k(\mathcal{A},\vec{r}_0)} \left\|\vec{b} - \mathcal{A}\vec{y}\right\| \iff \|\vec{r}_k\| = \min_{q\in\Pi_k,\ q(0)=1} \|q(\mathcal{A})\vec{r}_0\|. \tag{1.66}$$

Let v_k be eigenvector of $\mathcal{A}$ corresponding to eigenvalue λ_k, i.e., $\mathcal{A}v_k = \lambda_k v_k$, $k = 1, 2, ...$ Suppose

$$\vec{r}_0 = \sum_i \alpha_i v_i$$

for some coefficients α_i. Since $\vec{r}_k = q_k(\mathcal{A})\vec{r}_0$ from (1.65), we can write

$$\vec{r}_k = \sum_i \alpha_i q_k(\lambda_i) v_i.$$

Hence (1.66) can be estimated as

$$\|\vec{r}_k\| = \min_{q\in\Pi_k,\ q(0)=1} (\sum_i \alpha_i^2 q(\lambda_i)^2(\vec{v}_i,\vec{v}_i))^{1/2} \le \min_{q_k\in\Pi_k,\ q_k(0)=1} \max_i |q(\lambda_i)|\,\|\vec{r}_0\|\,. \quad (1.67)$$

Thus we see that the rate of convergence of MINRES method depends on the distribution of the eigenvalues of the saddle-point matrix $\mathcal{A}$.

Effective preconditioner for the saddle-point system (1.63) has to cluster both the positive and negative eigenvalues so that (1.67) is sufficiently small after a few iterations. Next very important property of effective preconditioner is that it has to preserve the symmetry of our saddle-point system (1.63). If we were to destroy the symmetry, we would have to use Krylov subspace methods for nonsymmetric systems.

Consider block diagonal preconditioner

$$\mathcal{P}_d = \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y} \end{pmatrix}, \quad (1.68)$$

where $\mathbf{X} \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\mathbf{Y} \in \mathbb{R}^{m\times m}$ is symmetric positive definite. It follows that $\mathcal{P}_d$ is symmetric positive definite and therefore we can write

$$\mathcal{P}_d = \mathcal{P}_d^{\frac{1}{2}}\mathcal{P}_d^{\frac{1}{2}}. \quad (1.69)$$

Then two-sided preconditioning of the system (1.63) with $\mathcal{P}_d$ has a form

$$\mathcal{P}_d^{-\frac{1}{2}}\mathcal{A}\mathcal{P}_d^{-\frac{1}{2}}\vec{y} = \mathcal{P}_d^{-\frac{1}{2}}\vec{b},\ \ \vec{y} = \mathcal{P}_d^{\frac{1}{2}}\vec{x}. \quad (1.70)$$

As the matrix $\mathcal{P}_d^{-\frac{1}{2}}\mathcal{A}\mathcal{P}_d^{-\frac{1}{2}}$ is symmetric indefinite, we can use MINRES method for solving preconditioned system (1.70).

For a suitable choice of the matrices $\mathbf{X}$ and $\mathbf{Y}$, consider the following generalized eigenvalue problem:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y} \end{pmatrix}\begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix}$$
$$\iff \quad . \quad (1.71)$$
$$\begin{aligned} \mathbf{A}\vec{x}_1 + \mathbf{B}^\mathsf{T}\vec{x}_2 &= \lambda\mathbf{X}\vec{x}_1 \\ \mathbf{B}\vec{x}_1 &= \lambda\mathbf{Y}\vec{x}_2 \end{aligned}$$

It is clear that if $\mathbf{X} = \mathbf{A}$ and $\vec{x}_2 = \vec{0}$, then $\lambda = 1$ is an eigenvalue of multiplicity $n - m$ corresponding to any eigenvector $\begin{pmatrix} \vec{x}_1 \\ \vec{0} \end{pmatrix}$ with $\mathbf{B}\vec{x}_1 = \vec{0}$.

For $\mathbf{Y} = -\mathbf{S} = \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\mathsf{T}$ we have

$$\begin{aligned} \mathbf{B}^\mathsf{T}\vec{x}_2 &= (\lambda-1)\mathbf{A}\vec{x}_1 \\ \mathbf{B}\vec{x}_1 &= \lambda\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\vec{x}_2 \end{aligned} \iff (\lambda^2-\lambda-1)\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\vec{x}_2 = \vec{0}. \quad (1.72)$$

By solving $\lambda^2 - \lambda - 1 = 0$ we get the remaining eigenvalues $\lambda = \frac{1}{2}(1 \pm \sqrt{5})$, each of multiplicity m. It means that the matrix of preconditioned system (1.70) has

3 distinct eigenvalues and therefore MINRES method applied to this preconditioned system will terminate with the exact solution after 3 iterations. Considering advantages of MINRES method (three-term recurrences, optimality), we can conclude that

$$\mathcal{P}_d = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix}. \tag{1.73}$$

is effective preconditioner for the saddle-point system (1.63). For more details we refer to Elman [2014], chapter 4.

Also other approaches are possible. One can use symmetric indefinite block diagonal preconditioner or nonsymmetric block diagonal preconditioner, which lead to solving of nonsymmetric systems. For this purpose, see Benzi et al. [2005], pages 61-66.

1.5.2 Block triangular preconditioner

Let us consider the saddle-point system

$$\mathcal{A}\vec{x} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \vec{\mathbf{b}} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix}, \tag{1.74}$$

where the block $\mathbf{A} \in \mathbb{R}^{n\times n}$ is nonsymmetric and $\mathbf{B} \in \mathbb{R}^{m\times n}$ has full row rank, $n, m \in \mathbb{N}, n \geq m$. As $\mathbf{A}$ is nonsymmetric, saddle-point matrix $\mathcal{A}$ is nonsymmetric and therefore we can use Krylov subspace method like GMRES for solving the system (1.74). For accelerating the rate of convergence of GMRES method we need effective preconditioner.

We start with the block diagonal preconditioner $\mathcal{P}_d$ of the form as in (1.73), i.e.,

$$\mathcal{P}_d = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix}. \tag{1.75}$$

Unlike the preconditioner (1.73), this preconditioner is nonsymmetric because $\mathbf{A}$ is nonsymmetric. Similarly as in (1.71)-(1.72), we get that the matrix of preconditioned system has 3 distinct eigenvalues, which means that GMRES method applied to preconditioned system will terminate with the exact solution after 3 iterations. However, when we use GMRES method, we must give up either three-term recurrences or optimality. Let us show how to construct more effective preconditioner for saddle-point system (1.74).

Consider $\mathbf{LU}$ factorization of the saddle-point matrix $\mathcal{A}$:

$$\mathcal{A} = \mathcal{L}\mathcal{U} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \tag{1.76}$$

Taking $\mathcal{P}_t = \mathcal{U}$ we receive

$$\mathcal{A}\mathcal{P}_t^{-1} = \mathcal{L} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix},$$

which means that all eigenvalues of preconditioned system $\mathcal{A}\mathcal{P}_t^{-1}$ are equal to 1. Thus GMRES method will terminate with the exact solution after 2 iterations.

Hence effective preconditioner for saddle-point system (1.74) with nonsymmetric block $\mathbf{A}$ has a form

$$\mathcal{P}_t = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}. \tag{1.77}$$

For more details we refer to Elman [2014], section 9.2.

We derived effective diagonal

$$\mathcal{P}_d = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix} \iff \mathcal{P}_d^{-1} = \begin{pmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S}^{-1} \end{pmatrix} \tag{1.78}$$

and triangular

$$\mathcal{P}_t = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \iff \mathcal{P}_t^{-1} = \begin{pmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\mathbf{B}^\mathsf{T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{-1} \end{pmatrix} \tag{1.79}$$

block preconditioners for corresponding saddle-point systems. However, as you can see from (1.78) and (1.79), the inverses of $\mathbf{A}$ and $\mathbf{S}$ are required. It is too expensive for practical computations and therefore suitable approximations are needed. This is one of the subjects of the next chapter.

2. Incompressible flow

Let $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$, be a bounded domain occupied by the fluid during the time interval $I = (0,T) \subset \mathbb{R}$, $T > 0$. Let us denote $\Omega_T = \Omega \times (0,T)$. Using Eulerian description for fluid motion, the differential form of the law of conservation of mass is given by the *continuity equation*

$$\frac{\partial \rho}{\partial t} + \rho \operatorname{div}(\vec{v}) = 0 \ \textit{in}\ \Omega_T, \tag{2.1}$$

where $\rho = \rho(\vec{x},t)$ is the density of the fluid and $\vec{v} = \vec{v}(\vec{x},t)$ is the velocity of fluid particle that at time t occupies the point $\vec{x}$.

The law of conservation of linear momentum is given by the *equation of the motion*

$$\frac{\partial(\rho\vec{v})}{\partial t} + \operatorname{div}(\rho\vec{v} \otimes \vec{v}) = \rho\vec{f} + \operatorname{div}(\mathbb{T}) \ \textit{in}\ \Omega_T, \tag{2.2}$$

where $\mathbb{T}$ is the stress tensor, $\vec{f}$ is the density of the body force and $p = p(\vec{x},t)$ represents the pressure.

Remark 11
Notice that the velocity $\vec{v}$ is the vector function and the pressure p is the scalar function.

In this thesis we will be interested in incompressible, homogeneous, Newtonian fluids. This type of fluids has a constant density $\rho > 0$ and therefore the continuity equation (2.1) can be written as

$$\operatorname{div}(\vec{v}) = 0 \ \textit{in}\ \Omega_T. \tag{2.3}$$

The stress tensor $\mathbb{T}$ is given by the following constitutive equation

$$\mathbb{T} = (-p + \lambda \operatorname{div}(\vec{v}))\mathbb{I} + \mu(\nabla\vec{v} + (\nabla\vec{v})^{\mathsf{T}}), \tag{2.4}$$

where λ and μ are viscosity coefficients. As $\operatorname{div}(\vec{v}) = 0$, the coefficient λ is unimportant. Coefficient μ is called *dynamic viscosity* and it represents the resistance of the fluid to shearing. Based on (2.3) and (2.4), the equation of the motion (2.2) can be written as

$$\frac{\partial\vec{v}}{\partial t} + (\vec{v}\cdot\nabla)\vec{v} = \vec{f} - \frac{\nabla p}{\rho} + \frac{\mu}{\rho}\,\Delta\vec{v} \ \textit{in}\ \Omega_T \tag{2.5}$$

provided $\mu > 0$ is constant. The term $(\vec{v}\cdot\nabla)\vec{v}$ is called *convective term* and the quantity $\nu = \frac{\mu}{\rho} > 0$ is called *kinematic viscosity*. Using simplification $p := \frac{p}{\rho}$, we can rewrite (2.5) as

$$\frac{\partial\vec{v}}{\partial t} - \nu\,\Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p = \vec{f} \ \textit{in}\ \Omega_T \tag{2.6}$$

Thus the equations describing the flow of incompressible, homogeneous, Newtonian fluids are given by

$$\begin{aligned} \frac{\partial\vec{v}}{\partial t} - \nu\,\Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p &= \vec{f} \\ \operatorname{div}(\vec{v}) &= 0 \end{aligned} \tag{2.7}$$

considered in Ω_T, $T > 0$.

Remark 12
The system of equations (2.7) *is called* ***Navier-Stokes system****.*

Throughout this thesis we will distinguish between *steady* and *unsteady* flow. The *steady flow* is defined as a type of flow in which the fluid parameters are time-independent, i.e. $\frac{\partial \cdot}{\partial t} = 0$. The *unsteady flow* is a type of flow in which the fluid parameters are time-dependent.

In the case of steady flow, the Navier-Stokes system has a form

$$\begin{aligned} -\nu\ \Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p = \vec{f}\ \ in\ \Omega \\ \operatorname{div}(\vec{v}) = 0\ \ in\ \Omega \end{aligned} \tag{2.8}$$

Remark 13
Notice that the Navier-Stokes system is nonlinear in both steady and unsteady cases due to convective term $(\vec{v}\cdot\nabla)\vec{v}$*.*

For more details about the mechanics of continua and incompressible fluids we refer to Gurtin [2010].

2.1 Finite element method

In this section we remind the basic definitions and concepts of the finite element method (FEM) for solving an elliptic boundary value problems posed over a bounded domain $\Omega \subset \mathbb{R}^d$ with a Lipschitz continuous boundary $\partial\Omega = \Gamma$. For more details about FEM we refer to Ciarlet [2002].

2.1.1 Galerkin method

Let V be infinite dimensional Hilbert space and consider the abstract variational problem:

$$Find\ v \in V\colon\ a(v,u) = F(u)\ \forall u \in V, \tag{2.9}$$

where $a : V \times V \to \mathbb{R}$ is continuous V-elliptic bilinear form and $F : V^* \to \mathbb{R}$ is continuous linear functional.

Theorem 5 (Lax-Milgram)
Under the assumptions above, the variational problem (2.9) *has an unique solution* $v \in V$*.*

Proof. See Ciarlet [2002], pages 8-9.

□

Instead of seeking a solution $v \in V$, we will look for its approximation. A well-known approach for approximating the solution $v \in V$ is **Galerkin method**, which consists in defining discrete problems over finite dimensional subspaces $V^h \subset V$ of the space V. Let V^h be any finite dimensional subspace of V. Then the discrete problem reads:

$$Find\ v_h \in V^h\colon\ a(v_h,u_h) = F(u_h)\ \forall u_h \in V^h. \tag{2.10}$$

As V^h is Hilbert space, the problem (2.10) has a unique solution thanks Lax-Milgram theorem (5). To solve the discrete problem (2.10), we must construct a

suitable finite dimensional subspace V^h of V. This issue will be discussed below, but assume for now that $\{\varphi_1, ..., \varphi_{N_h}\}$ forms a basis of V^h with $\dim(V^h) = N_h$. Expressing the solution $v_h \in V^h$ of (2.10) in terms of the basis functions φ_i, we can write

$$v_h = \sum_{i=1}^{N_h} V_i \varphi_i, \tag{2.11}$$

where the coefficients V_i are to be determined. Substituting (2.11) into (2.10) and taking $u_h = \varphi_j$, we obtain the system of linear equations

$$\sum_{i=1}^{N_h} a(\varphi_i, \varphi_j) V_i = F(\varphi_j),\ 1 \leq j \leq N_h \tag{2.12}$$

for the coefficients V_i. This system is called **discrete system** or Galerkin system and it can be solved by using direct or iterative methods.

For constructing the finite dimensional subspace V^h of a Hilbert space V, we use FEM. Consider that the variational problem is posed over the domain Ω. Let us define a characteristic feature of the FEM - triangulation $\mathscr{T}^h$ of the domain Ω:

Definition 6 (Triangulation)
*The finite family $\mathscr{T}_h$ of the subsets T of $\overline{\Omega}$ is called **triangulation** of Ω, if*

1. $\overline{\Omega} = \bigcup_{T \in \mathscr{T}_h} T$
2. *any $T \in \mathscr{T}_h$ is closed and the interior int(T) of T is nonempty and connected*
3. *for each $T \in \mathscr{T}_h$ the boundary ∂T is Lipschitz continuous*
4. *the intersection of the interiors of any two distinct $T_1,\ T_2 \in \mathcal{T}_h$ is empty*

Let $\mathcal{T}_h = \{T_i\}$ be a triangulation of the domain Ω. The subsets T_i are often called **cells**. In practice, cells T_i are typically simple polygonal shapes like triangles, quadrilateral or tetrahedra. The parameter h is called **mesh size parameter** and it denotes the maximum diameter of the cells T_i of the triangulation $\mathcal{T}_h$. On each cell T_i we define a finite dimensional space of functions P_{T_i} and the basis Σ_{T_i} of the dual space $P^*_{T_i}$. This triple is called **finite element**. More formally, the classical Ciarlet's definition of a finite element (Ciarlet [2002], page 78) is the following:

Definition 7 (Finite element)
*A **finite element** is a triple (T, P_T, Σ_T) where*

1. *$T \subset \mathbb{R}^d$ is a bounded closed set with nonempty interior and the Lipschitz continuous boundary ∂T*
2. *P_T is a n-dimensional space of functions defined on T*
3. *The set of degrees of freedom $\Sigma_T = \{\sigma_1, ..., \sigma_n\}$ forms a basis of P^*_T, the dual space of P_T.*

The finite dimensional subspaces V^h of V are then constructed by patching together a set of finite elements $\{T, P_T, \Sigma_T\}_{T\in\mathcal{T}_h}$. FEM in its simplest form is a process of constructing such finite dimensional subspaces V^h, which shall be called **finite element spaces** (Ciarlet [2002], page 38).

Remark 14
We speak about conforming finite element method, if $V^h \subset V$. Otherwise we speak about nonconforming finite element method.

Since this thesis is not about the constructing the finite element spaces, we will not discuss all details and all possible constructions here. The specific choice of finite element spaces will be specified according to the problem.

2.1.2 Inf-sup condition

Consider the abstract variational problem:

$$\textit{Find } v \in V\textit{: } a(v,u) = F(u)\ \forall u \in V, \tag{2.13}$$

where V is infinite dimensional Hilbert space, $a : V \times V \to \mathbb{R}$ is continuous bilinear form and $F : V^* \to \mathbb{R}$ is continuous linear functional. Let us discuss the existence and uniqueness of the solution of (2.13).

Consider first the case when the bilinear form a is V-elliptic. Then the Lax-Milgram theorem (5) implies the existence and an uniqueness of the solution $v \in V$ of (2.13). Let us consider the case when bilinear form a is not V-elliptic. In this case a weaker condition is sufficient:

$$\exists \alpha > 0 : 0 < \ \alpha = \inf_{\substack{v\in V\\ v\neq 0}} \sup_{\substack{u\in V\\ u\neq 0}} \frac{a(v,u)}{\|v\|_V \|u\|_V}, \tag{2.14}$$

where $\|\cdot\|_V$ denotes the norm in V. Condition (2.14) is commonly referred to as **continuous inf-sup condition**. If bilinear form a satisfies continuous inf-sup condition (2.14), then the abstract variational problem (2.13) has an unique solution (Theorem 2.8, Babuška [1973], pages 184-185).

Let us discuss the stability of the discretization of (2.13). Let $V^h \subset V$ be finite dimensional subspace of V. Consider the discrete problem:

$$\textit{Find } v_h \in V^h\textit{: } a(v_h,u_h) = F(u_h)\ \forall u_h \in V^h. \tag{2.15}$$

In order to have an unique solution for the discrete problem (2.15), the discrete analogue of (2.14) must hold:

$$\exists \alpha_0 > 0 : 0 < \ \alpha_0 \leq \alpha_h = \inf_{\substack{v_h\in V^h\\ v_h\neq 0}} \sup_{\substack{u_h\in V^h\\ u_h\neq 0}} \frac{a(v_h,u_h)}{\|v_h\|_V \|u_h\|_V}. \tag{2.16}$$

To be uniformly stable, the constant α_h must be uniformly bounded below. Condition (2.16) is referred to as **inf-sup condition** or discrete inf-sup condition. If (2.16) does not hold, instabilities may occur. For more details we refer to Logg et al. [2012], pages 659-660

Remark 15
The condition (2.16) *does not necessarily hold for any finite element space V^h.*

2.2 Stokes problem

Assume that Ω is a bounded domain in $\mathbb{R}^d$, $d \in \{2, 3\}$, with a Lipschitz continuous boundary $\partial\Omega = \Gamma$ and let us consider the Navier-Stokes system for steady flow as in (2.8). In the case $\nu >> 1$, i.e., the fluid is very viscous, the term $\nu\Delta\vec{v}$ dominates over the convective term and thus we can ignore the convective term. Then the simplified system, called **Stokes system**, can be written as

$$\begin{aligned} -\nu\ \Delta\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega \\ \operatorname{div}(\vec{v}) &= 0\ \ in\ \Omega \end{aligned} \tag{2.17}$$

Let us introduce the boundary value problem. Assume that $\Gamma = \Gamma_D \cup \Gamma_N$, where the Dirichlet boundary Γ_D and the Neumann boundary Γ_N are distinct. Adding boundary conditions

$$\begin{aligned} \vec{v} &= \vec{v}_D\ \ on\ \Gamma_D \\ \nu\frac{\partial\vec{v}}{\partial n} - p\vec{n} &= \vec{v}_N\ \ \ on\ \Gamma_N \end{aligned} \tag{2.18}$$

we received the **Stokes problem**: Find the velocity $\vec{v} = \vec{v}(\vec{x})$ and the pressure $p = p(\vec{x})$ such that

$$\begin{aligned} -\nu\ \Delta\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega \\ \operatorname{div}(\vec{v}) &= 0\ \ in\ \Omega \\ \vec{v} &= \vec{v}_D\ \ on\ \Gamma_D \\ \nu\frac{\partial\vec{v}}{\partial n} - p\vec{n} &= \vec{v}_N\ \ \ on\ \Gamma_N, \end{aligned} \tag{2.19}$$

where $\frac{\partial\vec{v}}{\partial n}$ denotes the directional derivative in the normal direction, $\vec{n}$ is the unit outer normal vector, $\vec{v}_D$ is prescribed fluid velocity on the Dirichlet boundary and $\vec{v}_N$ is the surface force or stress on the Neumann boundary.

The aim of this section is to derive the discrete Stokes system for the Stokes problem (2.19) by using FEM and to discuss the effective preconditioning techniques for its solving. We start with the variational problem:

2.2.1 Variational problem

Let us introduce a mixed variational problem of (2.19). Suppose $\nu > 0$, $\vec{f} \in \mathbf{L}^2(\Omega)$, $\vec{v}_D \in \mathbf{H}^{\frac{1}{2}}(\Gamma_D)$, $\int_{\Gamma_D} \vec{v}_D \cdot \vec{n} = 0$, $\vec{v}_N \in \mathbf{L}^2(\Gamma_N)$ and let us denote $\mathbf{H}^1_D(\Omega) = \{\vec{v} \in \mathbf{H}^1(\Omega) :\ \vec{v} = \vec{v}_D\ on\ \Gamma_D\}$. Assume that $(\vec{v},\ p)$ is a classical solution of (2.19), i.e., $\vec{v} \in \mathbf{C}^2(\overline{\Omega})$, $p \in C^1(\overline{\Omega})$ and $(\vec{v},\ p)$ satisfy (2.19). Testing the first equation (the equation of the motion) in (2.19) with an arbitrary $\vec{\varphi} \in \mathbf{H}^1_0(\Omega)$ and using per-partes integration yield

$$\nu\int_\Omega \nabla\vec{v} : \nabla\vec{\varphi} - \int_\Omega p\operatorname{div}(\vec{\varphi}) = \int_\Omega \vec{f}\cdot\vec{\varphi} + \int_{\Gamma_N} \vec{v}_N\cdot\vec{\varphi}. \tag{2.20}$$

Testing the second equation (the continuity equation) in (2.19) with an arbitrary $\psi\ \in L^2(\Omega)$ yield

$$\int_\Omega \psi\operatorname{div}(\vec{v}) = 0. \tag{2.21}$$

Then the mixed variational problem of (2.19) reads: Find $(\vec{v},\ p) \in \mathbf{H}^1_D(\Omega) \times L^2(\Omega)$ such that

$$\begin{aligned} \nu \int_\Omega \nabla \vec{v} : \nabla \vec{\varphi} - \int_\Omega p \operatorname{div}(\vec{\varphi}) &= \int_\Omega \vec{f} \cdot \vec{\varphi} + \int_{\Gamma_N} \vec{v}_N \cdot \vec{\varphi} \quad \forall \vec{\varphi} \in \mathbf{H}^1_0(\Omega) \\ \int_\Omega \psi \operatorname{div}(\vec{v}) &= 0 \quad \forall \psi \in L^2(\Omega) \end{aligned} \tag{2.22}$$

or equivalently

$$\begin{aligned} a(\vec{v}, \vec{\varphi}) + b(\vec{\varphi}, p) &= F(\vec{\varphi}) \quad \forall \vec{\varphi} \in \mathbf{H}^1_0(\Omega) \\ b(\vec{v}, \psi) &= 0 \quad \forall \psi \in L^2(\Omega) \end{aligned} \tag{2.23}$$

with

$$\begin{aligned} a(\vec{v}, \vec{\varphi}) &= \nu \int_\Omega \nabla \vec{v} : \nabla \vec{\varphi} \\ b(\vec{v}, \psi) &= -\int_\Omega \psi \operatorname{div}(\vec{v}) \\ F(\vec{\varphi}) &= \int_\Omega \vec{f} \cdot \vec{\varphi} + \int_{\Gamma_N} \vec{v}_N \cdot \vec{\varphi}. \end{aligned} \tag{2.24}$$

If $\vec{v}_* \in \mathbf{H}^1(\Omega),\ \operatorname{div}(\vec{v}_*) = 0$ and $\vec{v}_*|_{\Gamma_D} = \vec{v}_D$ in the sense of traces, then we can reformulate the mixed variational problem as: Find $(\vec{v},\ p) \in \mathbf{H}^1(\Omega) \times L^2(\Omega)$ such that

$$\begin{aligned} \vec{v} - \vec{v}_* &\in \mathbf{H}^1_0(\Omega) \\ a(\vec{v}, \vec{\varphi}) + b(\vec{\varphi}, p) &= F(\vec{\varphi}) \quad \forall \vec{\varphi} \in \mathbf{H}^1_0(\Omega) \\ b(\vec{v}, \psi) &= 0 \quad \forall \psi \in L^2(\Omega) \end{aligned} \tag{2.25}$$

For more details we refer to Elman [2014].

2.2.2 Discrete Stokes problem

Let us assume that $\mathbf{V}^h_0 \subset \mathbf{H}^1_0(\Omega),\ \mathbf{V}^h \subset \mathbf{H}^1(\Omega)$ and $P^h \subset L^2(\Omega)$ are finite dimensional spaces. Then the discrete Stokes problem is as follows: Find $(\vec{v}_h,\ p_h) \in \mathbf{V}^h \times P^h$ such that

$$\begin{aligned} \vec{v}_h - \vec{v}_{*h} &\in \mathbf{V}^h_0(\Omega) \\ a(\vec{v}_h, \vec{\varphi}_h) + b(\vec{\varphi}_h, p_h) &= F(\vec{\varphi}_h) \quad \forall \vec{\varphi}_h \in \mathbf{V}^h_0 \\ b(\vec{v}_h, \psi_h) &= 0 \quad \forall \psi_h \in P^h, \end{aligned} \tag{2.26}$$

where $\vec{v}_{*h} \in \mathbf{V}^h$ approximate the function $\vec{v}_*$.

For the construction finite dimensional spaces V^h_0 and P^h, we use FEM. Without loss of generality consider that $\Omega \subset \mathbb{R}^d,\ d \in \{2, 3\}$, is a polygonal domain. If not, we can consider the polygonal approximation $\Omega_h \approx \Omega$. Let $\mathcal{T}_h$ be a triangulation of the domain Ω consisting of the d-simplicies $T \in \mathcal{T}_h$. To ensure the stability of the discretization, the finite element spaces $\mathbf{V}^h_0$ and P^h have to satisfy the inf-sup condition

$$\exists \alpha > 0 :\ \alpha \leq \inf_{\substack{\psi_h \in P^h \\ \psi_h \neq 0}} \sup_{\substack{\vec{\varphi}_h \in \mathbf{V}^h_0 \\ \vec{\varphi}_h \neq \vec{0}}} \frac{|b(\vec{\varphi}_h, \psi_h)|}{\|\vec{\varphi}_h\|_{\mathbf{H}^1_0} \|\psi_h\|_{L^2}}. \tag{2.27}$$

Well-known finite element spaces satisfying (2.27) are the following

$$\begin{aligned} \mathbf{V}_0^h &= \{\vec{\varphi}_h \in \mathbf{C}(\overline{\Omega_h}) : \ \vec{\varphi}_h \in [P^{k+1}(T)]^d, \vec{\varphi}_h = \vec{0} \ on \ \Gamma_D \ \forall T \in \mathscr{T}_h\}, \\ P^h &= \{\psi_h \in C(\overline{\Omega_h}) : \ \psi_h \in P^k(T) \ \forall T \in \mathscr{T}_h\}. \end{aligned} \tag{2.28}$$

In this case we speak about conforming **Taylor-Hood elements** P^{k+1}/P^k, $k \geq 1$. Another pair of finite element spaces satisfying (2.27) is the nonconforming **Crouzeix-Raviant elements**. For more examples of inf-sup stable finite element spaces we refer to Logg et al. [2012], chapter 20.

Assume that $\mathbf{V}_0^h$ and P^h satisfy the inf-sup condition (2.27). Let $\dim(\mathbf{V}_0^h) = n_v \in \mathbb{N}$ and $\dim(P^h) = n_p \in \mathbb{N}$. If $\{\vec{\varphi}_1, ..., \vec{\varphi}_{n_v}\}$ is a basis for $\mathbf{V}_0^h$ and $\{\psi_1, ..., \psi_{n_p}\}$ is a basis for P^h, then the solution $(\vec{v}_h, \ p_h) \in \mathbf{V}^h(\Omega) \times P^h(\Omega)$ of (2.26) can be expressed as

$$\begin{aligned} \vec{v}_h &= \vec{v}_{*h} + \sum_{j=1}^{n_v} V_j \vec{\varphi}_j \\ p_h &= \sum_{j=1}^{n_p} P_j \psi_j, \end{aligned} \tag{2.29}$$

where the real coefficients V_j and P_j are to be determined. Substituting (2.29) into (2.26) and using test functions $\vec{\varphi}_h = \vec{\varphi}_i$, $i = 1, ..., n_v$, and $\psi_h = \psi_i$, $i = 1, ..., n_p$, we obtain the **discrete Stokes system**

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{c} \\ \vec{d} \end{pmatrix}, \tag{2.30}$$

where

$$\begin{aligned} a_{ij} &= a(\vec{\varphi}_j, \vec{\varphi}_i) \\ b_{ij} &= b(\vec{\varphi}_j, \psi_i) \\ c_i &= F(\vec{\varphi}_i) - \nu \int_\Omega \nabla \vec{v_{*h}} : \nabla \vec{\varphi}_i \\ d_i &= -b(\vec{v}_{*h}, \psi_i). \end{aligned} \tag{2.31}$$

Remark 16
The matrix $\boldsymbol{A}$ *is called the* ***vector-Laplacian matrix*** *and the matrix* $\boldsymbol{B}$ *is called the* ***divergence matrix****.*

By definition, the matrix $\mathbf{A}$ is obviously symmetric. Moreover, it can be shown that it is positive definite (see Elman [2014], page 17). As the inf-sup condition (2.27) holds, the matrix $\mathbf{B}$ has a full row rank. At this point we are able to solve the discrete Stokes system (2.30) by using a direct or iterative methods. In the next part we will discuss how to solve this system by using Krylov subspace methods with effective preconditioning.

2.2.3 Preconditioning

We derived the discrete Stokes system

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^\mathsf{T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{c} \\ \vec{d} \end{pmatrix}, \tag{2.32}$$

where the matrix $\mathbf{A}$ is symmetric positive definite and the matrix $\mathbf{B}$ has a full row rank. As the Schur complement matrix $\mathbf{S}$ is symmetric and negative definite, the saddle-point matrix of (2.32) is indefinite. Robust Krylov subspace method for solving such systems is MINRES method. Now we turn our attention to finding effective preconditioner for (2.32). Here we can apply the theory described in subsection 1.5.1. In this subsection we showed that the effective preconditioner for systems like (2.32) is of the form

$$\mathcal{P}_E = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix} \iff \mathcal{P}_E^{-1} = \begin{pmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S}^{-1} \end{pmatrix}. \tag{2.33}$$

Remark 17
The subscript E in the preconditioner $\mathcal{P}_E$ means that the preconditioner is exact or ideal.

In general, computation with the exact preconditioners is very expensive and therefore inexact preconditioners are used. In our case the action of the inverses of the matrix $\mathbf{A}$ and of the dense Schur complement matrix $\mathbf{S}$ are required. It is impractical and therefore we will approximate our exact preconditioner $\mathcal{P}_E$ by inexact preconditioner $\mathcal{P}_I$ of the form

$$\mathcal{P}_I = \begin{pmatrix} \tilde{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{S}} \end{pmatrix} \iff \mathcal{P}_I^{-1} = \begin{pmatrix} \tilde{\mathbf{A}}^{-1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{S}}^{-1} \end{pmatrix}, \tag{2.34}$$

where $\tilde{\mathbf{A}} \approx \mathbf{A}$ and $\tilde{\mathbf{S}} \approx -\mathbf{S}$.

It is well-known that the **pressure mass matrix** $\mathbf{M}_P$ is spectrally equivalent to the matrix (**-S**) and therefore we set $\tilde{\mathbf{S}} = \mathbf{M}_P$. For the choice $\tilde{\mathbf{A}} = \mathbf{A}$ and $\tilde{\mathbf{S}} = \mathbf{M}_P$, the MINRES method applied to the preconditioned discrete Stokes system will not terminates after three iterations as in the case of using the exact preconditioner (2.33), but the convergence rate will be fast enough. Moreover, the number of iterations of MINRES method will be independent of the mesh size parameter. Here we refer to Elman [2014], section 4.2, and Olshanskii and Tyrtyshnikov [2014], section 5.2.10.

Remark 18
Let $\{\psi_i\}$ be basis for the discrete pressure space. Then the pressure mass matrix $\boldsymbol{M}_P$ defined as $m_{p_{ij}} = \int_\Omega \psi_j \psi_i$, is symmetric positive definite.

The choice $\tilde{\mathbf{A}} = \mathbf{A}$ requires one exact Poisson solve. Our aim is to select $\tilde{\mathbf{A}}$ to be any appropriate preconditioner for Laplacian operator. A suitable approximation of the matrix $\mathbf{A}^{-1}$ can be performed by application of one V-cycle of algebraic multigrid method (Elman [2014], page 198). Another option is to invert the matrix $\mathbf{A}$ by using CG method.

Thus the effective preconditioner for the discrete Stokes system (2.32) is based on the approximation of the matrix (-$\mathbf{S}$) by pressure mass matrix $\mathbf{M}_P$ and using one V-cycle of multigrid method for approximating the inverse of the matrix $\mathbf{A}$. For more details about multigrid method we refer to Saad [2003], chapter 13.

2.3 Navier-Stokes problem

Assume that $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, is a bounded domain with a Lipschitz continuous boundary $\Gamma = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$. Let us denote $I = (0,\ T)$, $T > 0$, the time interval and $\Omega_T = \Omega \times I$. The Navier-Stokes system for unsteady flow consists of a pair of equation of the motion and the continuity equation:

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} - \nu\ \Delta \vec{v} + (\vec{v} \cdot \nabla)\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega_T \\ \mathrm{div}(\vec{v}) &= 0\ \ in\ \Omega_T, \end{aligned} \tag{2.35}$$

where

$$(\vec{v} \cdot \nabla)\vec{v} = \sum_{i=1}^{d} v_i \frac{\partial \vec{v}}{\partial x_i} \tag{2.36}$$

is the convective term, $\vec{v} = \vec{v}(\vec{x}, t)$ denotes the velocity of the fluid and $p = p(\vec{x}, t)$ represent the pressure (or more precisely the ratio between the pressure and density of the fluid). This system was derived at the beginning of this chapter.

Now let us consider the boundary value problem called **Navier-Stokes problem**: Find the velocity $\vec{v} = \vec{v}(\vec{x}, t)$ and the pressure $p = p(\vec{x}, t)$ such that

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} - \nu\ \Delta \vec{v} + (\vec{v} \cdot \nabla)\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega_T \\ \mathrm{div}(\vec{v}) &= 0\ \ in\ \Omega_T \\ \vec{v} &= \vec{v}_D\ \ on\ \Gamma_D \times I \\ \nu \frac{\partial \vec{v}}{\partial n} - p\vec{n} &= \vec{0}\ \ on\ \Gamma_N \times I \\ \vec{v}(\vec{x}, 0) &= \vec{v}_0\ \ in\ \Omega, \end{aligned} \tag{2.37}$$

where the kinematic viscosity $\nu > 0$, the fluid velocity $\vec{v}_D$ on the Dirichlet boundary Γ_D, initial condition $\vec{v}_0 = \vec{v}_0(\vec{x})$ and the density of the body force $\vec{f} = \vec{f}(\vec{x}, t)$ are prescribed.

The aim of this section is to describe an effective preconditioning techniques for solving discrete Navier-Stokes problem while we will distinguish between steady and unsteady flow.

2.3.1 Steady flow

First we turn our attention towards steady flow. In this case the flow parameters are time-independent and therefore the problem (2.37) reduces to seeking the velocity $\vec{v} = \vec{v}(\vec{x})$ and the pressure $p = p(\vec{x})$ such that

$$\begin{aligned} -\nu\ \Delta \vec{v} + (\vec{v} \cdot \nabla)\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega \\ \mathrm{div}(\vec{v}) &= 0\ \ in\ \Omega \\ \vec{v} &= \vec{v}_D\ \ on\ \Gamma_D \\ \nu \frac{\partial \vec{v}}{\partial n} - p\vec{n} &= \vec{0}\ \ on\ \Gamma_N. \end{aligned} \tag{2.38}$$

In the similar manner as in the Stokes problem, we will derive the discrete Navier-Stokes system for steady flow and then we will discuss the effective preconditioning techniques.

Remark 19
The Navier-Stokes system (or problem) describing the steady flow is simply called a steady Navier-Stokes system (or problem).

Discrete problem

Assume that $\nu > 0,\ \vec{f} \in \mathbf{L}^2(\Omega),\ \vec{v}_D \in \mathbf{H}^{\frac{1}{2}}(\Gamma_D),\ \int_{\Gamma_D} \vec{v}_D \cdot \vec{n} = 0$ and $\vec{v}_* \in \mathbf{H}^1(\Omega)$ such that $\mathrm{div}(\vec{v}_*) = 0$ and $\vec{v}_*|_{\Gamma_D} = \vec{v}_D$ in the sense of traces. Similarly as in the Stokes problem, we first introduce the concept of mixed variational problem.

Testing the first equation in (2.38) with an arbitrary $\vec{\varphi} \in \mathbf{H}^1_0(\Omega)$ and the second equation with an arbitrary $\psi \in L^2(\Omega)$ yield the mixed variational problem: Find $(\vec{v},\ p) \in \mathbf{H}^1_D(\Omega) \times L^2(\Omega)$ such that

$$\begin{aligned} \nu \int_\Omega \nabla\vec{v} : \nabla\vec{\varphi} + \int_\Omega (\vec{v}\cdot\nabla)\vec{v}\cdot\vec{\varphi} - \int_\Omega p\,\mathrm{div}(\vec{\varphi}) &= \int_\Omega \vec{f}\cdot\vec{\varphi} \quad \forall\vec{\varphi} \in \mathbf{H}^1_0(\Omega) \\ \int_\Omega \psi\,\mathrm{div}(\vec{v}) &= 0 \quad \forall\psi \in L^2(\Omega). \end{aligned} \tag{2.39}$$

Using the notation as in (2.24) with $F(\vec{\varphi}) = \int_\Omega \vec{f}\cdot\vec{\varphi}$ (as $\vec{v}_N = \vec{0}$ on Γ_N), the problem (2.39) can be written as: Find $(\vec{v},\ p) \in \mathbf{H}^1(\Omega) \times L^2(\Omega)$ such that

$$\begin{aligned} \vec{v} - \vec{v}_* &\in \mathbf{H}^1_0(\Omega) \\ a(\vec{v},\vec{\varphi}) + e(\vec{v},\vec{v},\vec{\varphi}) + b(\vec{\varphi},p) &= F(\vec{\varphi}) \quad \forall\vec{\varphi} \in \mathbf{H}^1_0(\Omega) \\ b(\vec{v},\psi) &= 0 \quad \forall\psi \in L^2(\Omega) \end{aligned} \tag{2.40}$$

where

$$e(\vec{v},\vec{v},\vec{\varphi}) = \int_\Omega \sum_{i,j=1}^{d} v_j \frac{\partial v_i}{\partial x_j}\varphi_i. \tag{2.41}$$

Similarly as in the Stokes problem, let $\mathbf{V}^h_0 \subset \mathbf{H}^1_0(\Omega)$ and $P^h \subset L^2(\Omega)$ be finite element spaces satisfied the inf-sup condition (2.27). Moreover, let $\mathbf{V}^h \subset \mathbf{H}^1(\Omega)$ be finite dimensional subspace of the space $\mathbf{H}^1(\Omega)$. Then the discrete steady Navier-Stokes problem has a form: Find $(\vec{v}_h,\ p_h) \in \mathbf{V}^h \times P^h$ such that

$$\begin{aligned} \vec{v}_h - \vec{v}_{*h} &\in \mathbf{V}^h_0 \\ a(\vec{v}_h,\vec{\varphi}_h) + e(\vec{v}_h,\vec{v}_h,\vec{\varphi}_h) + b(\vec{\varphi}_h,p_h) &= F(\vec{\varphi}_h) \quad \forall\vec{\varphi}_h \in \mathbf{V}^h_0 \\ b(\vec{v}_h,\psi_h) &= 0 \quad \forall\psi_h \in P^h, \end{aligned} \tag{2.42}$$

where $\vec{v}_{*h} \approx \vec{v}_*$ as in the Stokes problem.

As the convective term is nonlinear, the discrete steady Navier-Stoke problem is nonlinear. Linearization can be performed by using the **Newton method** or the **Picard method**. Let us describe the using of the Newton method in short. Let $(\vec{v}^0_h, p^0_h) \in \mathbf{V}^h(\Omega) \times L^2(\Omega)$ be given initial iterate. We will construct the sequence $\{(\vec{v}^k_h, p^k_h)\} \subset \mathbf{V}^h(\Omega) \times L^2(\Omega)$ of iterates which should converge to the solution $(\vec{v}_h, p_h) \in \mathbf{V}^h(\Omega) \times L^2(\Omega)$ of the discrete problem (2.42). Using the Newton method, the discrete steady Navier-Stokes problem in the k-th iterate reads: Find corrections $\hat{v}^k_h \in \mathbf{V}^h_0$ and $\hat{p}^k_h \in P^h$ such that

$$\begin{aligned} a(\hat{v}^k_h,\vec{\varphi}_h) + b(\vec{\varphi}_h,\hat{p}^k_h) + e(\hat{v}^k_h,\vec{v}^k_h,\vec{\varphi}_h) + e(\vec{v}^k_h,\hat{v}^k_h,\vec{\varphi}_h) &= R^k_V(\vec{\varphi}_h) \quad \forall\vec{\varphi}_h \in \mathbf{V}^h_0 \\ b(\hat{v}^k_h,\psi_h) &= R^k_P(\psi_h) \quad \forall\psi_h \in P^h, \end{aligned} \tag{2.43}$$

where

$$\begin{aligned} R_V^k(\vec{\varphi}_h) &= F(\vec{\varphi}_h) - e(\vec{v}_h^k, \vec{v}_h^k, \vec{\varphi}_h) - a(\vec{v}_h^k, \vec{\varphi}_h) - b(\vec{\varphi}_h, p_h^k) \\ R_P^k(\psi_h) &= -b(\vec{v}_h^k, \psi_h). \end{aligned} \tag{2.44}$$

If $\{\psi_1, ..., \psi_{n_p}\}$ is a basis for P^h and $\{\vec{\varphi}_1, ..., \vec{\varphi}_{n_v}\}$ is a basis for $\mathbf{V}_0^h$, then any element $\vec{v}_h \in \mathbf{V}^h$, $\hat{v}_h \in \mathbf{V}_0^h$ and $p_h,\ \hat{p}_h \in P^h$ can be decomposed on these bases as follows:

$$\begin{aligned} \vec{v}_h &= \vec{v}_{*h} + \sum_{j=1}^{n_v} \alpha_j \vec{\varphi}_j, \qquad & \hat{v}_h &= \sum_{j=1}^{n_v} V_j \vec{\varphi}_j, \\ p_h &= \sum_{j=1}^{n_p} \beta_j \psi_j, & \hat{p}_h &= \sum_{j=1}^{n_p} P_j \psi_j. \end{aligned} \tag{2.45}$$

Substituting (2.45) into (2.43) and using test functions $\vec{\varphi}_h = \vec{\varphi}_i$, $i = 1, ..., n_v$, and $\psi_h = \psi_i$, $i = 1, ..., n_p$, we obtain the **discrete steady Navier-Stokes system**

$$\begin{pmatrix} \tilde{\mathbf{A}}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{c} \\ \vec{d} \end{pmatrix}, \tag{2.46}$$

where

$$\begin{aligned} \tilde{\mathbf{A}}^{(k)} &= \mathbf{A} + \mathbf{N}^{(k)} + \mathbf{W}^{(k)} \\ a_{ij} &= a(\vec{\varphi}_j, \vec{\varphi}_i) \\ n_{ij}^{(k)} &= e(\vec{v}_h^k, \vec{\varphi}_j, \vec{\varphi}_i) \\ w_{ij}^{(k)} &= e(\vec{\varphi}_j, \vec{v}_h^k, \vec{\varphi}_i) \\ b_{ij} &= b(\vec{\varphi}_j, \psi_i) \end{aligned} \tag{2.47}$$

and the right-hand-side vectors are the nonlinear residuals associated with the current iterate $(\vec{v}_h^k, p_h^k)$.

Remark 20
The matrix $\boldsymbol{N}^{(k)}$ *is called the* ***vector-convection matrix*** *and the matrix* $\boldsymbol{W}^{(k)}$ *is called the* ***Newton derivative matrix****.*

After solving the system of linear equations (2.46) for the current iterate k, the next iterate $(\vec{v}_h^{k+1}, p_h^{k+1}) \in \mathbf{V}^h(\Omega) \times L^2(\Omega)$ is defined via

$$\begin{aligned} \vec{v}_h^{k+1} &= \vec{v}_h^k + \hat{v}_h^k \\ p_h^{k+1} &= p_h^k + \hat{p}_h^k. \end{aligned} \tag{2.48}$$

It means that we need to solve the system of linear equations (2.46) in each iterate k of the Newton method. For more details about using Newton and Picard methods for the discrete steady Navier-Stokes problem we refer to Elman [2014], pages 344-346.

Let us discuss the properties of the saddle-point matrix of the discrete steady Navier-Stokes system (2.46). We already know that the vector-Laplacian matrix $\mathbf{A}$ is symmetric and positive definite. The divergence matrix $\mathbf{B}$ has a full row rank as the finite element spaces satisfied the inf-sup condition. By definition, the

Newton derivative matrix $\mathbf{W}^{(k)}$ is symmetric and the vector-convection matrix $\mathbf{N}^{(k)}$ is nonsymmetric. Thus the matrix $\tilde{\mathbf{A}}$ is nonsymmetric and therefore the saddle-point matrix of the discrete steady Navier-Stokes system is nonsymmetric. Krylov subspace methods like GMRES or direct methods can be used for solving such systems.

If we use Picard method instead of Newton method, then in each iterate we will solve the following saddle-point system

$$\begin{pmatrix} \mathbf{A} + \mathbf{N}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{c} \\ \vec{d} \end{pmatrix}. \tag{2.49}$$

As before, the matrix $\mathbf{A}+\mathbf{N}^{(k)}$ is nonsymmetric due to nonsymmetricity of the matrix $\mathbf{N}^{(k)}$ and therefore the saddle-point matrix (2.49) arises from Picard method is also nonsymmetric.

Preconditioning

The last step in determining the solution $(\vec{v}_h,\ p_h) \in \mathbf{V}^h \times P^h$ of the discrete steady Navier-Stokes problem (2.42) is to effectively solve the saddle-point system

$$\begin{pmatrix} \mathbf{F} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{c} \\ \vec{d} \end{pmatrix}, \tag{2.50}$$

which arises from the Newton or Picard method in each iteration. As mentioned before, the matrix $\mathbf{F}$ is nonsymmetric for both Newton and Picard method and therefore the saddle-point matrix of the system (2.50) is nonsymmetric. For solving such a system, we can use Krylov subspace method like GMRES or FGMRES.

In the section 1.5.2 we discussed the effective block triangular preconditioner for systems like (2.50). We concluded that the effective exact preconditioner for such systems is of the form

$$\mathcal{P}_E = \begin{pmatrix} \mathbf{F} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}. \tag{2.51}$$

As we mentioned in the Stokes problem, the computing with the exact preconditioner is impractical and very expensive. Let us now discuss the approximations of the matrices $\mathbf{F}$ and $\mathbf{S}$.

We start with the approximation of the matrix $\mathbf{F}$. A spectrally equivalent approximation of the matrix $\mathbf{F}$ can be obtained by using a few cycles of multigrid method. For more details we refer to Benzi et al. [2005], page 69. Another option, how to invert the matrix $\mathbf{F}$, is to use GMRES method or LU factorization.

The approximation of the Schur complement matrix $\mathbf{S}$ is a delicate problem. We show two approaches how to approximate the Schur complement matrix $\mathbf{S}$. Both these approaches are inspired by the section 9.2 in Elman [2014], pages 364-384. Assume that the saddle-point system (2.50) arises from the Picard method, i.e, $\mathbf{F} = \mathbf{A} + \mathbf{N}^{(k)}$. Then the entries of the matrix $\mathbf{F}$ in the k-the iterate have a form

$$f_{ij} = \nu \int_\Omega \nabla\vec{\varphi}_j \cdot \nabla\vec{\varphi}_i + \int_\Omega (\vec{v}_h^k \cdot \nabla\vec{\varphi}_j)\vec{\varphi}_i, \tag{2.52}$$

where $\{\vec{\varphi}_1, ..., \vec{\varphi}_{n_v}\}$ forms a basis of the space $\mathbf{V}_0^h$. Consider the convection-diffusion operator

$$L_V = -\nu\Delta + \vec{v}_h^k \cdot \nabla \tag{2.53}$$

defined on the velocity space. Note that the matrix $\mathbf{F}$ contains a discrete version of the operator L_V as a component. Let us denote

$$L_P = -\nu\Delta + \vec{v}_h^k \cdot \nabla \tag{2.54}$$

the convective-diffusion operator defined on the pressure space. Then we can introduce the commutator L of the convection-diffusion operators L_V and L_P with the divergence operator $\nabla\cdot$ as follows

$$L = \nabla \cdot L_V - L_P \nabla\cdot \tag{2.55}$$

The discrete version L_h of the commutator L can be written as

$$L_h = (\mathbf{M}_P^{-1}\mathbf{B})(\mathbf{M}_V^{-1}\mathbf{F}) - (\mathbf{M}_P^{-1}\mathbf{F}_P)(\mathbf{M}_P^{-1}\mathbf{B}), \tag{2.56}$$

where $\mathbf{M}_V$ denotes the velocity mass matrix, $\mathbf{M}_P$ denotes the pressure mass matrix and the matrix $\mathbf{F}_P$ is defined as

$$f_{P_{ij}} = \nu \int_\Omega \nabla\psi_j \cdot \nabla\psi_i + \int_\Omega (\vec{v}_h^k \cdot \nabla)\psi_j \cdot \psi_i. \tag{2.57}$$

Here $\{\psi_1, ..., \psi_{n_p}\}$ denotes the basis of the space P^h.

Suppose that L_h is small ($L_h \approx 0$). Multiplying (2.56) by $\mathbf{M}_P\mathbf{F}_P^{-1}\mathbf{M}_P$ from the left and by $\mathbf{F}^{-1}\mathbf{B}^\mathsf{T}$ from the right we obtain

$$-\mathbf{S} = \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^\mathsf{T} \approx \mathbf{M}_P\mathbf{F}_P^{-1}(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^\mathsf{T}). \tag{2.58}$$

Since the matrix $\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^\mathsf{T}$ is dense, we replace this matrix by spectrally equivalent the pressure Laplacian matrix $\mathbf{A}_P$ which is defined as

$$a_{P_{ij}} = \int_\Omega \nabla\psi_j \cdot \nabla\psi_i. \tag{2.59}$$

Using $\mathbf{A}_P$ instead of $\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^\mathsf{T}$, we obtain the following approximation of the Schur complement matrix $\mathbf{S}$:

$$\begin{gathered} \mathbf{S} \approx -\mathbf{M}_P\mathbf{F}_P^{-1}\mathbf{A}_P \\ \Longleftrightarrow \\ \mathbf{S}^{-1} \approx -\mathbf{A}_P^{-1}\mathbf{F}_P\mathbf{M}_P^{-1} \end{gathered} \tag{2.60}$$

This approximation of the Schur complement matrix $\mathbf{S}$ is commonly referred to as **pressure convection-diffusion preconditioning** (PCD).

Remark 21
Once the pressure Laplacian matrix $\boldsymbol{A}_P$ is defined, we can write

$$\boldsymbol{F}_P = \nu\boldsymbol{A}_P + \boldsymbol{K}_P, \tag{2.61}$$

*where $\boldsymbol{K}_P$ defined as $k_{P_{ij}} = \int_\Omega (\vec{v}_h^k \cdot \nabla)\psi_j \cdot \psi_i$, is **pressure convection matrix**.*

Applying the action of the approximation of the inverse of the Schur complement matrix $\mathbf{S}$ to a vector requires a solving a system with the Poisson matrix $\mathbf{A}_P$, matrix-vector multiplication by matrix $\mathbf{F}_p$ and a solving of a system with the pressure mass matrix $\mathbf{M}_P$. For prescribing the matrices $\mathbf{A}_P$ and $\mathbf{F}_P$, we need some boundary conditions. For Poisson solve, a homogeneous Neumann boundary condition is prescribed on Γ_D and a homogeneous Dirichlet boundary condition is prescribed on Γ_N. To define a matrix $\mathbf{F}_P$, we consider a homogeneous Robin condition as in Elman [2014], page 373. A poor choice of boundary conditions can seriously affect the performance of the PCD preconditioner.

The second approach of approximating the Schur complement matrix $\mathbf{S}$ is based on the constructing the matrix $\mathbf{F}_P$ by solving the least-squares problem

$$\min \left\| [\mathbf{M}_V^{-1}\mathbf{F}^{\mathsf{T}}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}}]_i - \mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{M}_P^{-1}[\mathbf{F}_P^{\mathsf{T}}]_i \right\|_{\mathbf{M}_V} \tag{2.62}$$

for the i-th column of the matrix $\mathbf{F}_P^{\mathsf{T}}$. Hence we need to solve the following normal equations

$$\mathbf{M}_P^{-1}\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{M}_P^{-1}[\mathbf{F}_P^{\mathsf{T}}]_i = [\mathbf{M}_P^{-1}\mathbf{B}\mathbf{M}_V^{-1}\mathbf{F}^{\mathsf{T}}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}}]_i. \tag{2.63}$$

After algebraic manipulations with using the symmetry of the mass matrices we obtain the following definition of the matrix $\mathbf{F}_P$

$$\mathbf{F}_P = (\mathbf{B}\mathbf{M}_V^{-1}\mathbf{F}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}})(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}})^{-1}\mathbf{M}_P. \tag{2.64}$$

Substituting (2.64) into (2.58) leads to the following approximation of the Schur complement matrix $\mathbf{S}$

$$\begin{aligned} \mathbf{S} &\approx -(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}})(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{F}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}})^{-1}(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}}) \\ &\Longleftrightarrow \\ \mathbf{S}^{-1} &\approx -(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}})^{-1}(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{F}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}})(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^{\mathsf{T}})^{-1} \end{aligned} \tag{2.65}$$

As the inverse of the velocity mass matrix $\mathbf{M}_V$ can be dense, we approximate $\mathbf{M}_V$ by the diagonal matrix $\mathbf{diag}(\mathbf{M}_V) = \mathbf{D}_V$. The resulting approximation to the inverse of the Schur complement matrix $\mathbf{S}$ is given by

$$\mathbf{S}^{-1} \approx -(\mathbf{B}\mathbf{D}_V^{-1}\mathbf{B}^{\mathsf{T}})^{-1}(\mathbf{B}\mathbf{D}_V^{-1}\mathbf{F}\mathbf{D}_V^{-1}\mathbf{B}^{\mathsf{T}})(\mathbf{B}\mathbf{D}_V^{-1}\mathbf{B}^{\mathsf{T}})^{-1} \tag{2.66}$$

This approach is called **least-squares commutator preconditioning** (LSC). Unlike the PCD preconditioning, the LSC preconditioning requires two discrete Poisson solves and matrix-vector multiplications with the matrices $\mathbf{B}$, $\mathbf{B}^{\mathsf{T}}$ and $\mathbf{D}_V^{-1}$. The advantage of LSC is that the explicit construction of the matrices $\mathbf{F}_P$ and $\mathbf{A}_P$ is not needed. Moreover, any other boundary conditions are not needed as the global boundary conditions are considered.

2.3.2 Unsteady flow

Let us remind the unsteady Navier-Stokes problem from the beginning of this section: Find the velocity $\vec{v} = \vec{v}(\vec{x}, t)$ and the pressure $p = p(\vec{x}, t)$ such that

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} - \nu\ \Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega_T \\ \operatorname{div}(\vec{v}) &= 0\ \ in\ \Omega_T \\ \vec{v} &= \vec{v}_D\ \ on\ \Gamma_D \times I \\ \nu\frac{\partial \vec{v}}{\partial n} - p\vec{n} &= \vec{0}\ \ on\ \Gamma_N \times I \\ \vec{v}(\vec{x}, 0) &= \vec{v}_0\ \ in\ \Omega, \end{aligned} \tag{2.67}$$

where the kinematic viscosity $\nu > 0$, the fluid velocity $\vec{v}_D$ on the Dirichlet boundary Γ_D, initial condition $\vec{v}_0 = \vec{v}_0(\vec{x})$ and the density of the body force $\vec{f} = \vec{f}(\vec{x}, t)$ are prescribed. Here $\Omega_T = \Omega \times I$ with $I = [0, T]$.

In a similar manner as steady case, we derive the discrete unsteady Navier-Stokes system. Since we work with the time dependent variables, unlike the steady flow case, the discretization in time is needed for numerical computations. For more details we refer to Elman [2014], chapter 10, pages 412-438.

Notation

We are going to consider time-dependent functions not as functions of $\vec{x}$ and t together, but rather as mappings of functions of t into the corresponding spaces of functions of $\vec{x}$. For example, we define

$$\begin{aligned} \vec{v} : I &\to \mathbf{H}^1_D(\Omega) \\ [\vec{v}(t)](\vec{x}) &:= \vec{v}(\vec{x}, t),\ t \in I. \end{aligned}$$

Discrete problem

Given $\nu > 0$, $\vec{f} \in \mathbf{L}^2(\Omega)$, $\vec{v}_D \in \mathbf{H}^{\frac{1}{2}}(\Gamma_D)$, $\int_{\Gamma_D} \vec{v}_D \cdot \vec{n} = 0$ and $\vec{v}_* \in \mathbf{H}^1(\Omega)$ such that $\operatorname{div}(\vec{v}_*) = 0$ and $\vec{v}_*|_{\Gamma_D} = \vec{v}_D$ in the sense of traces, the mixed variational problem for unsteady Navier-Stokes problem reads: For all time $t \in I$, find $(\vec{v},\ p) \in \mathbf{H}^1_D(\Omega) \times L^2(\Omega)$ such that

$$\begin{aligned} \int_\Omega \frac{d\vec{v}}{dt}\cdot\vec{\varphi} + \nu\int_\Omega \nabla\vec{v} : \nabla\vec{\varphi} + \int_\Omega (\vec{v}\cdot\nabla)\vec{v}\cdot\vec{\varphi} - \int_\Omega p\operatorname{div}(\vec{\varphi}) &= \int_\Omega \vec{f}\cdot\vec{\varphi} \\ \int_\Omega \psi\operatorname{div}(\vec{v}) &= 0 \end{aligned} \tag{2.68}$$

or equivalently

$$\begin{aligned} \left(\frac{d\vec{v}}{dt}, \vec{\varphi}\right)_{\mathbf{L}^2} + a(\vec{v}, \vec{\varphi}) + e(\vec{v}, \vec{v}, \vec{\varphi}) + b(\vec{\varphi}, p) &= F(\vec{\varphi}) \\ b(\vec{v}, \psi) &= 0 \end{aligned} \tag{2.69}$$

for all $(\vec{\varphi}, \psi) \in \mathbf{H}^1_0(\Omega) \times L^2(\Omega)$, with $\vec{v}(0) = \vec{v}_0(\vec{x})$.

Let $\mathcal{T}_h$ be the triangulation of the domain Ω. Over $\mathcal{T}_h$ consider inf-sup stable finite element spaces $\mathbf{V}^h_0 \subset \mathbf{H}^1_0(\Omega)$ and $P^h \subset L^2(\Omega)$ constructed by FEM. Moreover, let $V^h \subset \mathbf{H}^1(\Omega)$ be finite dimensional subspace of $\mathbf{H}^1(\Omega)$.

As the problem (2.69) is time-dependent, we need discretization in time. For this purpose, we use finite differences. Let $0 = t_0 < t_1 < ... < t_n = T$ be a partition of the time interval $I = [0, T]$. Let us denote

$$\tau_k = t_{k+1} - t_k, \quad \vec{v}_h(\vec{x}, t_k) = \vec{v}_h^k, \quad p_h(\vec{x}, t_k) = p_h^k, \quad \vec{f}(\vec{x}, t_k) = \vec{f}^k, \quad \vec{v}_{*h}(\vec{x}, t_k) = \vec{v}_{*h}^k \tag{2.70}$$

Let us now introduce the discrete unsteady Navier-Stokes problem with using Crank-Nicholson scheme. For simplicity we consider $\vec{v}_0 = \vec{0}$. Then the discrete unsteady Navier-Stokes problem reads: For each $k = 1, ..., n$ find $(\vec{v}_h^k,\ p_h^k) \in \mathbf{V}^h \times P^h$ such that for each $\vec{\varphi}_h \in \mathbf{V}_0^h$ and $\psi_h \in P^h$ the following hold

$$\begin{aligned} \vec{v}_h^k - \vec{v}_{*h}^k &\in \mathbf{V}_0^h \\ \left(\frac{\vec{v}_h^{k+1} - \vec{v}_h^k}{\tau_k}, \vec{\varphi}_h \right)_{\mathbf{L}^2} + a(\vec{v}_h^{k+\frac{1}{2}}, \vec{\varphi}_h) + e(\vec{v}_h^{k+\frac{1}{2}}, \vec{v}_h^{k+\frac{1}{2}}, \vec{\varphi}_h) + b(\vec{\varphi}_h, p_h^{k+\frac{1}{2}}) &= F^{k+\frac{1}{2}}(\vec{\varphi}_h) \\ b(\vec{v}_h^{k+1}, \psi_h) &= 0. \end{aligned} \tag{2.71}$$

with

$$\vec{v}_h^{k+\frac{1}{2}} = \frac{1}{2}\left(\vec{v}_h^{k+1} + \vec{v}_h^k\right), \quad p_h^{k+\frac{1}{2}} = \frac{1}{2}\left(p_h^{k+1} + p_h^k\right), \quad F^{k+\frac{1}{2}} = \frac{1}{2}\left(F^{k+1} + F^k\right). \tag{2.72}$$

Since the problem is nonlinear, the linearization at each time level is needed. For that purpose, we can use Newton or Picard method. If $\{\vec{\varphi}_1, ..., \vec{\varphi}_{n_v}\}$ is a basis for $\mathbf{V}_0^h$ and $\{\psi_1, ..., \psi_{n_p}\}$ is a basis for P^h, then using the similar strategy as in the steady case, we obtain the following system of linear equations

$$\begin{pmatrix} \hat{\mathbf{A}}^{(k+1)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V}^{k+1} \\ \vec{P}^{k+1} \end{pmatrix} = \begin{pmatrix} \vec{c}^k \\ \vec{d}^k \end{pmatrix}, \tag{2.73}$$

on each time level with the nonsymmetric matrix $\hat{\mathbf{A}}^{(k+1)}$. As the matrix $\hat{\mathbf{A}}^{(k+1)}$ is nonsymmetric, the saddle-point matrix of the system (2.73) is nonsymmetric.

Preconditioning

Let us discuss effective preconditioning techniques for solving (2.73) on each time level. As this system is nonsymmetric, Krylov subspace method like FGMRES or GMRES can be used. Now, we turn our attention to effective preconditioning techniques. As in the steady case, the effective preconditioner is block triangular preconditioner of the form

$$\mathcal{P}_E = \begin{pmatrix} \mathbf{F} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}. \tag{2.74}$$

The matrix F can be inverted by using Krylov subspace method like GMRES, or it can be replaced by one V-cycle of multigrid method. The problem here is again in the approximation of the Schur complement matrix $\mathbf{S}$. In the steady case, two approaches were presented - PCD preconditioning (2.60) and LSC preconditioning (2.66). For unsteady case, LSC preconditioning can be used without

any modifications, whereas some changes appear in the PCD preconditioning. Namely, the matrix $\mathbf{F}_P$ is defined as follows

$$\mathbf{F}_P = \frac{1}{\tau}\mathbf{M}_P + \nu\mathbf{A}_P + \mathbf{K}_P, \tag{2.75}$$

where τ stands for time step. It leads to the following approximations of the inverse of the Schur complement matrix $\mathbf{S}$:

$$\mathbf{S}^{-1} \approx -\mathbf{A}_P^{-1}\mathbf{F}_P\mathbf{M}_P^{-1} = (-\nu\mathbf{I} - \mathbf{A}_P^{-1}(\frac{1}{\tau}\mathbf{M}_P + \mathbf{K}_P))\mathbf{M}_P^{-1}. \tag{2.76}$$

Approximation (2.76) is usually referred as pressure-convection-reaction preconditioning (PCDR). The boundary conditions for constructing the matrices $\mathbf{A}_P$ and $\mathbf{F}_P$ are considered in the same manner as in the steady case. For detailed discussion about PCDR preconditioning we refer to Řehoř [2018], section 5.4, and to Blechta [2019], chapter 3.

3. Numerical results

In the previous chapter we studied the fundamental governing equations (the continuity equation and the equation of the motion) of fluid dynamics describing the flow of incompressible, homogeneous, Newtonian fluids. By using numerical methods, we derived the discrete Stokes and Navier-Stokes problems leading to saddle-point systems. We also discussed effective preconditioning techniques for accelerating the rate of the convergence of Krylov subspace methods for solving such systems. Let us now compare the performance of the different solvers for solving the saddle-point systems arising from fluid dynamics.

Remark 22
A discipline which connects mathematical modeling of fluid flows, numerical methods and computational software, is called ***computational fluid dynamics*** *(CFD).*

This chapter is devoted to present the results of numerical simulations of fluid flow problems. Two benchmark problems have been selected in this thesis for validation purposes:

1. lid-driven cavity
2. flow around the cylinder

All numerical simulations were performed using the software **Firedrake**. The attention is paid to comparing the performance of the solvers used in the computations. All codes presented in this chapter can be found on GitHub Mitro [2020].

3.1 Lid-driven cavity

The first benchmark, on which the numerical simulations were performed, is lid-driven cavity problem - see Ghia et al. [1982], Guj and Stella [1993] and Botella and Peyret [1998]. Is can be imagined as fluid confined to a closed cuboid cavity with flow driven by a prescribed tangential velocity on the top face.

3.1.1 Geometry and boundary conditions

Let us introduce the geometry of the lid-driven cavity problem in 2D and 3D together with boundary conditions. We start with the geometry of the problem in 2D. Let $\Omega = [0,1]^2$ represents a square cavity filled with fluid. Let us denote the walls of Ω as

1. $\Gamma_T = [0,1] \times 1$ the top wall
2. $\Gamma_B = [0,1] \times 0$ the bottom wall
3. $\Gamma_L = 0 \times [0,1]$ the left wall
4. $\Gamma_R = 1 \times [0,1]$ the right wall

To make the square domain Ω lid-driven, we assume that the upper wall Γ_U is moving from left to right with a velocity V. The geometry of the lid-driven cavity in 2D can be displayed as in the figure (3.1).

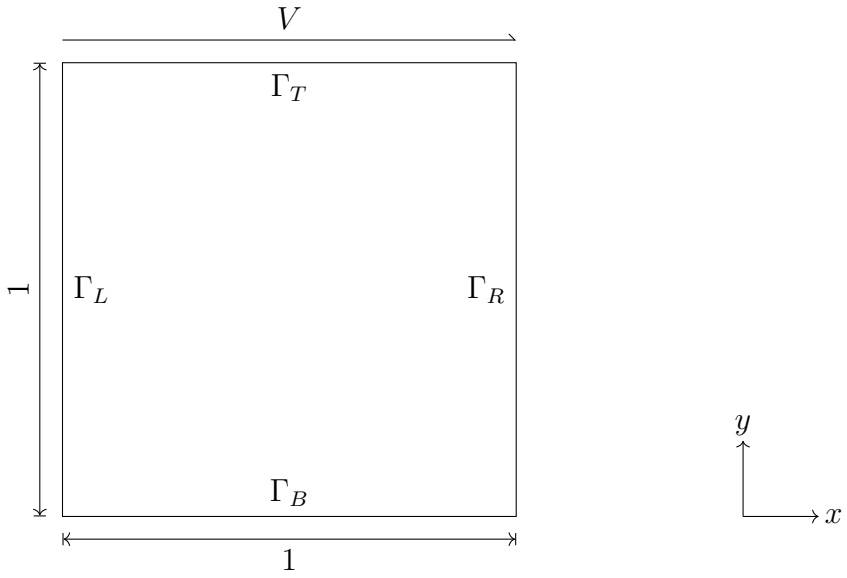


Figure 3.1: Lid-driven cavity problem: the geometry in 2D.

The boundary conditions for lid-driven problem are relatively simple. On the upper wall Γ_U a tangential velocity V is prescribed

$$\vec{v}|_{\Gamma_U} = (V, 0). \tag{3.1}$$

On the remaining three walls $\Gamma_L,\ \Gamma_B,\ \Gamma_R$ we consider no-slip boundary conditions

$$\vec{v}|_{\Gamma_L} = \vec{v}|_{\Gamma_B} = \vec{v}|_{\Gamma_R} = \vec{0}. \tag{3.2}$$

The computation domain in 3D is a cube $\Omega = [0, 1]^3$. The walls of Ω are denoted as follows

1. $\Gamma_T = [0, 1] \times 1 \times [0, 1]$ the top wall
2. $\Gamma_B = [0, 1] \times 0 \times [0, 1]$ the bottom wall
3. $\Gamma_L = [0, 1] \times [0, 1] \times 0$ the left wall
4. $\Gamma_R = [0, 1] \times [0, 1] \times 1$ the right wall
5. $\Gamma_F = 0 \times [0, 1] \times [0, 1]$ the front wall
6. $\Gamma_{Ba} = 1 \times [0, 1] \times [0, 1]$ the back wall

The boundary conditions are prescribed in the same manner as for 2D case:

$$\begin{aligned} &\vec{v}|_{\Gamma_V} = (V, 0, 0) \\ &\vec{v}|_{\Gamma_L} = \vec{v}|_{\Gamma_B} = \vec{v}_{\Gamma_R} = \vec{v}|_{\Gamma_{Ba}} = \vec{v}_{\Gamma_F} = \vec{0}. \end{aligned} \tag{3.3}$$

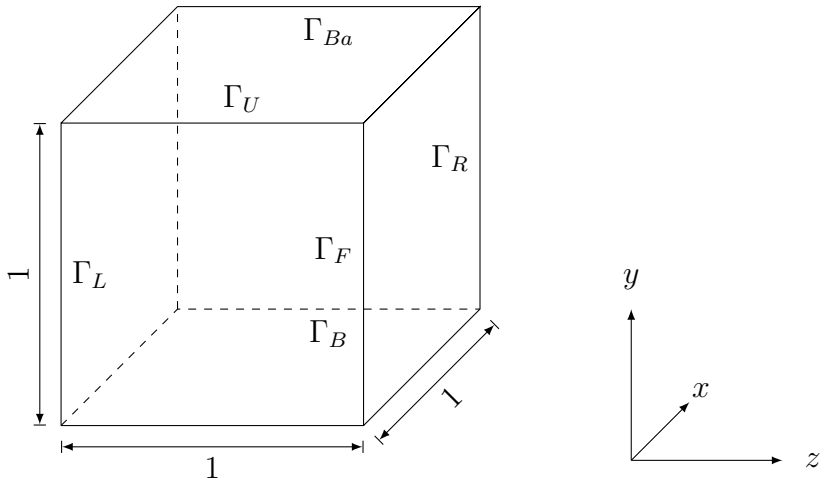


Figure 3.2: Lid-driven cavity problem: the geometry in 3D.

3.1.2 Steady Stokes problem in 2D

Let us first describe the behaviour of fluid in a lid-driven cavity in 2D by steady Stokes equations (2.17). Adding the boundary conditions (3.1), (3.2), the steady Stokes problem reads: Find the velocity $\vec{v} = (v_1(\vec{x}),\ v_2(\vec{x}))$ and the pressure $p = p(\vec{x})$ such that

$$\begin{aligned} -\nu\ \Delta\vec{v} + \nabla p &= \vec{f}\ in\ \Omega \\ \mathrm{div}(\vec{v}) &= 0\ in\ \Omega \\ \vec{v}|_{\Gamma_L} = \vec{v}|_{\Gamma_R} &= \vec{v}|_{\Gamma_B} = \vec{0} \\ \vec{v}|_{\Gamma_U} &= (V, 0). \end{aligned} \tag{3.4}$$

In our numerical simulation, we consider the kinematic viscosity $\nu = 0.01$, the density of the body force $\vec{f} = \vec{0}$ and the tangential velocity $V = 1$.

Let $\mathcal{T}_h$ be triangulation of the domain Ω consisting of the triangles. For the discretization of (3.4) we use inf-sup stable P^2/P^1 Taylor-Hood elements.

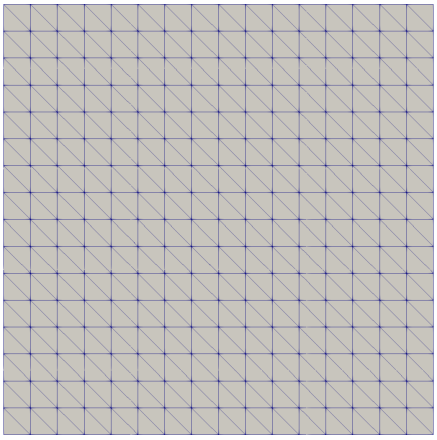


Figure 3.3: Triangulation $\mathcal{T}_h$ of the domain Ω for lid-driven cavity problem in 2D consisting of 512 triangles.

As we showed in section 2.2.2, the conforming finite element discretization of the steady Stokes problem with using inf-sup stable elements (in our case Taylor-Hood elements) leads to the saddle-point system

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{c} \\ \vec{d} \end{pmatrix} \iff \mathcal{A}\vec{x} = \vec{b}, \tag{3.5}$$

where the saddle-point matrix $\mathcal{A}$ is symmetric and indefinite. In section 2.2.3 we concluded that the effective inexact preconditioner $\mathcal{P}_I$ for such saddle-point system is block diagonal preconditioner of the form

$$\mathcal{P}_I = \begin{pmatrix} \tilde{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{S}} \end{pmatrix} \iff \mathcal{P}_I^{-1} = \begin{pmatrix} \tilde{\mathbf{A}}^{-1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{S}}^{-1} \end{pmatrix}, \tag{3.6}$$

where the matrix $\tilde{\mathbf{A}}$ approximates the matrix $\mathbf{A}$ and the matrix $\tilde{\mathbf{S}}$ approximates the matrix (-$\mathbf{S}$). Let us show how to solve (3.5) associated with the lid-driven cavity problem effectively by using computer.

Solver setting

In this part, the solver parameters for solving (3.5) in **Firedrake/PETSc** will be discussed. Three sets of parameters will be presented.

[MUMPS] The first set of parameters is based on using **MUMPS** (multifrontal massively parallel sparse direct solver) for solving systems of linear equations directly by using some factorization, for example LU factorization. For solving the saddle-point system (3.5), we use LU algorithm from MUMPS. In **Firedrake/PETSc**, it can be implemented as follows

```
MUMPS = {
       "mat_type":"aij",
       "ksp_type":"preonly",
       "pc_type":"lu",
       "pc_factor_mat_solver_type":"mumps",
       }
```

Figure 3.4: Lid-driven cavity problem in 2D - steady Stokes: set of parameters for MUMPS

[MINRES_PC] The second set of parameters is based on iterative solver with an effective block diagonal preconditioning. A suitable iterative method for solving (3.5) is MINRES, as we concluded in section 2.2.3. For accelerating the convergence of MINRES, we use inexact block diagonal preconditioner $\mathcal{P}_I$ defined above. As we discussed in section 2.2.3, the inverse of the matrix $\mathbf{A}$ can be approximated by using a single V-cycle of algebraic multigrid method and the matrix (-$\mathbf{S}$) can be approximated by spectrally equivalent pressure mass matrix $\mathbf{M}_P$, i.e, $\tilde{\mathbf{S}} = \mathbf{M}_P$. For inverting of the matrix $\mathbf{M}_P$, we use incomplete LU factorization. In **Firedrake/PETSc**, it can be implemented as follows

```
MINRES_PC = {
       "ksp_type": "minres",
       "pc_type": "fieldsplit",
       "pc_fieldsplit_type":"schur",
       "pc_fieldsplit_schur_fact_type":"diag",
       "fieldsplit_0":{
           "ksp_type":"preonly",
           "pc_type":"hypre",
           },
       "fieldsplit_1":{
           "ksp_type":"preonly",
           "pc_type":"python",
           "pc_python_type":"__main__.MassMatrix",
           "mass_pc_type":"bjacobi",
           "mass_sub_pc_type":"ilu",
           }
       }
```

Figure 3.5: Lid-driven cavity problem in 2D - steady Stokes: set of parameters for MINRES solver with block diagonal preconditioning

Remark 23
Instead of using the term iterative solver based on Krylov subspace method ,for example MINRES, we will use the term MINRES solver.

[MINRES] Finally, the third solver is MINRES solver without preconditioning. In **Firedrake/PETSc**, it can be setting as follows

```
MINRES = {
      "ksp_type":"minres",
      "pc_type":"none",
      }
```

Figure 3.6: Lid-driven cavity problem in 2D - steady Stokes: set of parameters for MINRES solver without preconditioning

Notice that MINRES solver with preconditioning needs much more parameters as MUMPS or MINRES solver without preconditioning. The reason is that we use $fieldsplit$ which presents an algebraic interface to building block preconditioners. Using $pc\_fieldsplit\_schur\_type$: $diag$ we tell the PETSc that we have block preconditioner and by $fieldsplit\_0$ and $fieldsplit\_1$ we specify how to apply the inverse of each diagonal block.

The figures (3.7), (3.8) represent the velocity and pressure distribution for the steady lid-driven cavity problem in 2D.

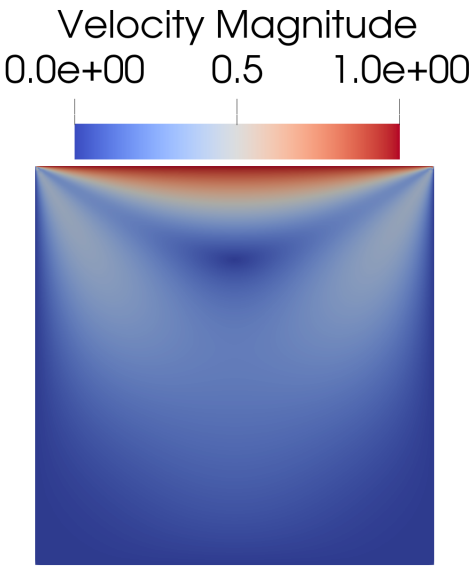


Figure 3.7: Lid-driven cavity problem in 2D - steady Stokes: Velocity of the fluid in lid-driven cavity in 2D.

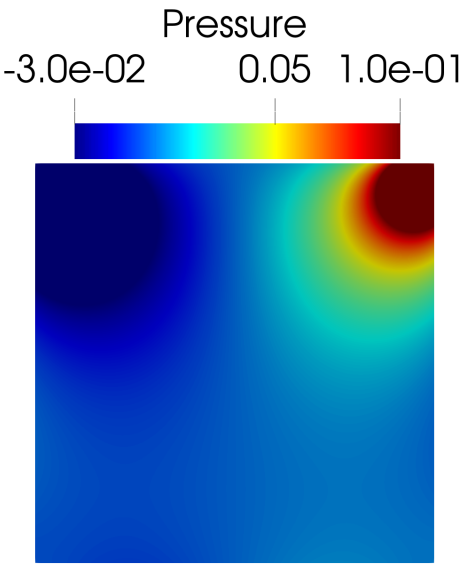


Figure 3.8: Lid-driven cavity problem in 2D - steady Stokes: Pressure of the fluid in lid-driven cavity in 2D.

Scalability

In this part, the main emphasis will be on comparing the scalability of solvers. First, let us recall the concept of scalability. We say that the program is scalable, if higher efficiency comes with larger amount of resources (processors). There are two different types of scalability:

- strong scalability
- weak scalability

Definition 8 (Strong scalability)
*The program is **strongly scalable**, if the achieved speedup is approximately equal to number of processors.*

It means that in strong scaling we concern the speedup for a fixed problem size with respect to the number of processors.

Definition 9 (Weak scalability)
*The program is **weakly scalable**, if the run time stays constant when problem size is increased proportionally to the number of processors.*

In weak scaling we concern the speedup for a scaled problem size with respect to the number of processors. Weak scalability is often easier to be achieved, unlike the strong scalability.

Definition 10 (Efficiency)
***Efficiency** E of the parallel program is defined as*

$$E = \frac{T_1}{pT_p}, \tag{3.7}$$

where p is number of processors, T_1 is time for running the program on one processor and T_p is time for running the program on p processors. The quantity T_1/T_p is called relative speedup of a parallel program.

Remark 24
The efficiency measures an acceleration with to the provided processors.

Let us discuss the scalability of all three solvers. As you can see in the strong scaling graph (3.9), respectively in the table (3.1), all three solvers are poor strongly scalable. The peak of performance of MUMPS solver was reached by using 16 processors and the peak of MINRES solver with preconditioning was reached by using 32 processors (see (3.1)). The MINRES solver with preconditioning was the fastest when using more than 8 processors, whereas the MINRES solver without preconditioning was the worst.

As you can see in the figure (3.10) of effectiveness, the MINRES solver without preconditioning is the most effective. The effectiveness of MINRES solver with preconditioning and MUMPS solver are almost the same.

The weak scaling graph (3.11) shows that both MUMPS solver and MINRES solver without preconditioning are not good weakly scalable. The MINRES solver with preconditioning is nearly weakly scalable. It means that the performance of the MINRES solver does not change dramatically as problem size increases proportionally to the number of processors.

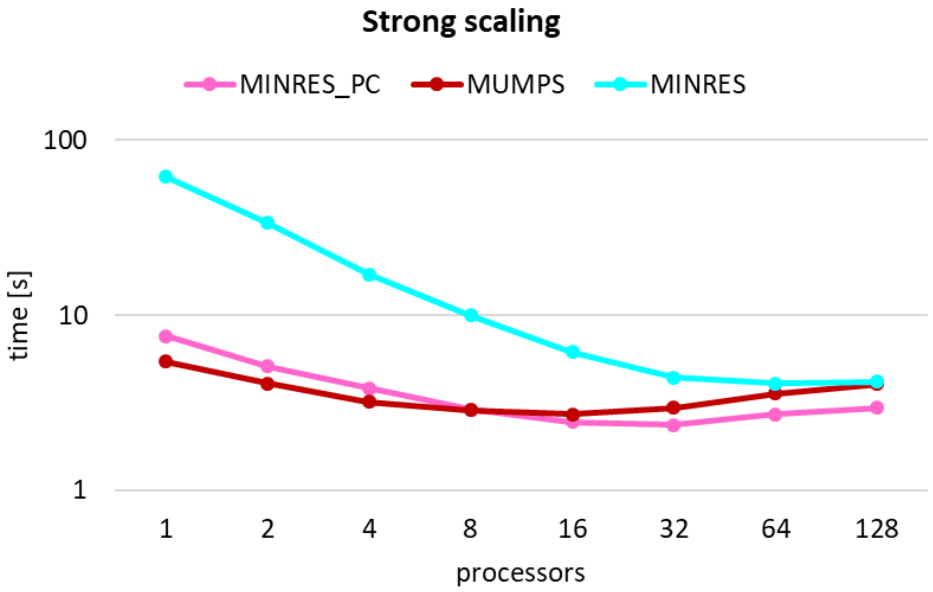


Figure 3.9: Lid-driven cavity problem in 2D - steady Stokes: Strong scaling for 148 739 unknowns on 32 768 cells, logarithmic scale.

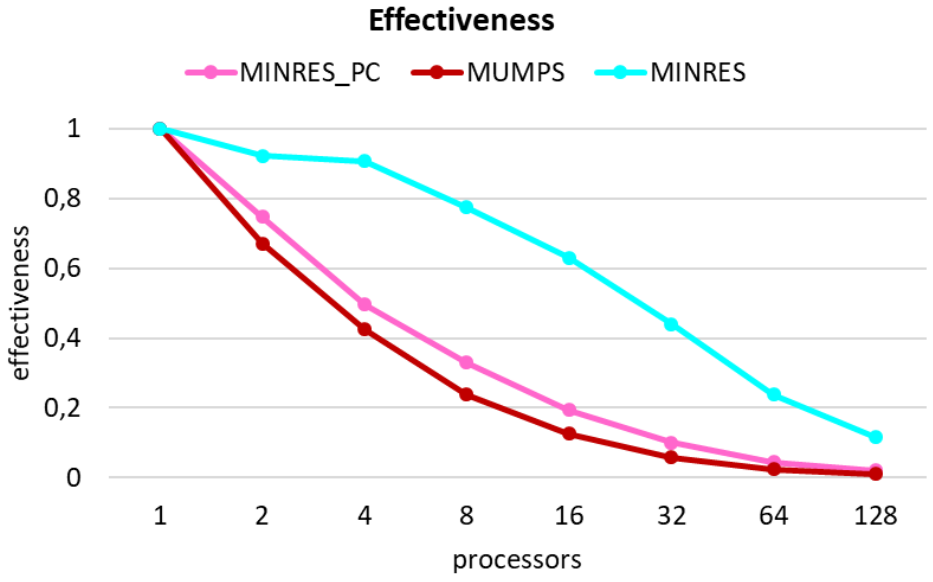


Figure 3.10: Lid-driven cavity problem in 2D - steady Stokes: Graph of effectiveness.

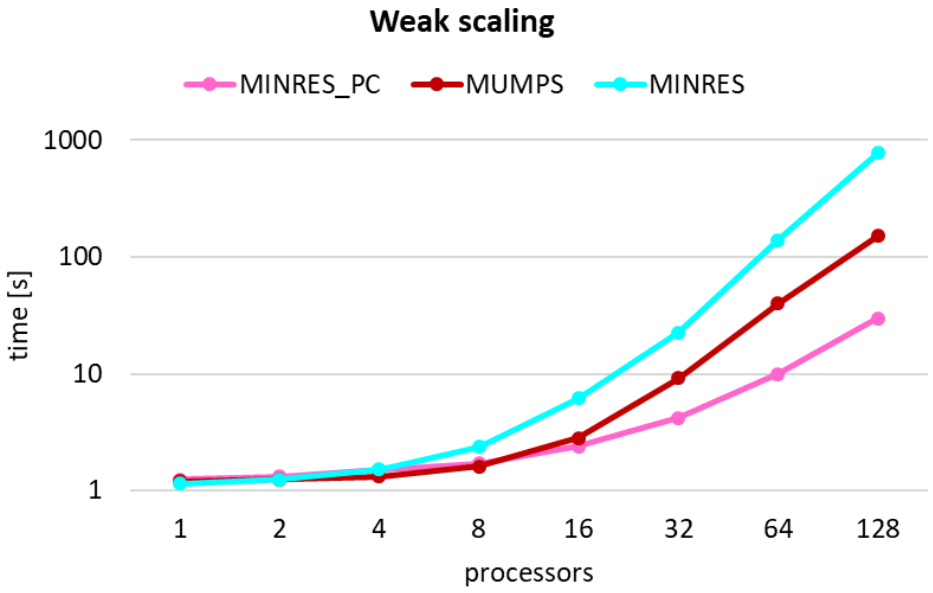


Figure 3.11: Lid-driven cavity problem in 2D - steady Stokes: Weak scaling graph, logarithmic scale.

processors	parameter	MUMPS	MINRES_PC	MINRES
1	time [s]	5.464	7.635	62.01
	iter	1	38	4 885
2	time [s]	4.068 s	5.106	33.6
	iter	1	42	4 885
4	time [s]	3.21 s	3.85	17.07
	iter	1	45	4 885
8	time [s]	2.877 s	2.893	9.994
	iter	1	46	4 885
16	time [s]	2.719 s	2.463	6.149
	iter	1	47	4 885
32	time [s]	2.969 s	2.358	4.409
	iter	1	47	4 884
64	time [s]	3.583	2.723	4.073
	iter	1	49	4 884
128	time [s]	4.051	2.968	4.193
	iter	1	50	4 884

Table 3.1: Lid-driven cavity problem in 2D - steady Stokes: Table of strong scaling data.

proc	cells	unknowns	parameter	MUMPS	MINRES_PC	MINRES
1	128	659	time [s]	1.199	1.24	1.146
			iter	1	41	201
2	512	2 467	time [s]	1.242	1.324	1.25
			iter	1	45	393
4	2 048	9 539	time [s]	1.315	1.493	1.523
			iter	1	48	917
8	8 192	37 507	time [s]	1.603	1.695	2.363
			iter	1	48	2 139
16	32 768	148 739	time [s]	2.817	2.387	6.137
			iter	1	47	4 885
32	131 072	592 387	time [s]	9.143	4.187	22.58
			iter	1	47	10 321
64	524 288	2 364 419	time [s]	40.09	9.961	140.3
			iter	1	48	21 557
128	2 097 152	9 447 427	time [s]	152.2	29.93	775.4
			iter	1	48	42 708

Table 3.2: Lid-driven cavity problem in 2D - steady Stokes: Table of weak scaling data.

3.1.3 Unsteady Navier-Stokes problem in 2D

Let us describe the flow of fluid in a lid-driven cavity by Navier-Stokes equations. First, we will discuss the case of unsteady flow in 2D. The unsteady Navier-Stokes problem reads: Find the velocity $\vec{v} = (v_1(\vec{x},\ t),\ \vec{v}_2(\vec{x},\ t))$ and the pressure $p = p(\vec{x},\ t)$ such that

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} - \nu\ \Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega \\ \mathrm{div}(\vec{v}) &= 0\ \ in\ \Omega \\ \vec{v}|_{\Gamma_L} = \vec{v}|_{\Gamma_R} &= \vec{v}|_{\Gamma_B} = \vec{0} \\ \vec{v}|_{\Gamma_T} &= (V, 0). \end{aligned} \tag{3.8}$$

The parameters in numerical simulation are taken as follows:

- the kinematic viscosity $\nu = 0.001$
- the body force $\vec{f} = \vec{0}$
- the time step $dt = 0.01$
- the initial time $T_0 = 0.0$
- the end time $T_{end} = 3$
- the tangential velocity $V = 1$

For the discretization in the space we use P^2/P^1 Taylor-Hood elements and for the discretization in the time the Crank-Nicholson scheme is used. The Newton method is considered for the linearization of the problem at each time level.

As we showed in section 2.3.2, the following saddle-point system is solved on each time level

$$\begin{pmatrix} \hat{\mathbf{A}}^{(k+1)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V}^{k+1} \\ \vec{P}^{k+1} \end{pmatrix} = \begin{pmatrix} \vec{c}^{k} \\ \vec{d}^{k} \end{pmatrix} \iff \mathcal{A}^{(k+1)}\vec{x} = \vec{b}, \tag{3.9}$$

where the saddle-point matrix $\mathcal{A}^{(k+1)}$ is nonsymmetric. Moreover, we concluded that the effective exact preconditioner for such system has a form

$$\mathcal{P}_E = \begin{pmatrix} \hat{\mathbf{A}}^{(k+1)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}. \tag{3.10}$$

Solver setting

Let us present three solvers for solving (3.9) in **Firedrake/PETSc**.

[MUMPS] The first solver is MUMPS. The set of parameters for MUMPS is the same as in (3.4).

The other two solvers are iterative solvers. As a outer method for solving the saddle-point system (3.9), both solvers use FGMRES method with the restart after 600 iterations. For inverting the matrix $\hat{\mathbf{A}}^{(k+1)}$, LU factorization is used. Now, we distinguish the solvers based on the approximation of the inverse of the Schur complement matrix $\mathbf{S}$.

[MyPCD] The second solver is FGMRES solver with our implementation of PCD preconditioning (2.60). Let us show the difference between our and firedrake's implementation of PCD preconditioning for unsteady flow. Our implementation of PCD preconditioning is based on (2.76), i.e., we use PCDR preconditioning. The firedrake's implementation does not support PCDR preconditioning and therefore PCD preconditioning defined by (2.60) is used for both steady and unsteady flow cases. It means that firedrake's PCD is not robust for unsteady flow problems. Let us recall that for construction the matrices $\mathbf{A}_P$ and $\mathbf{F}_P$, boundary conditions are needed. Here homogeneous Neumann boundary conditions are considered. Both systems associated with $\mathbf{A}_P$ and $\mathbf{F}_P$ are solved using LU factorization. The set of parameters for our implementation of PCD preconditioner in **Firedrake/PETSc** is as follows

```
MyPCD = {
  "ksp_type":"fgmres",
  "ksp_rtol": 1e-3,
  "ksp_gmres_restart":600,
  "ksp_gmres_modifiedgramschmidt":True,
  "mat_type":"matfree",
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"preonly",
    "pc_type":"python",
    "pc_python_type":"firedrake.AssembledPC",
    "assembled_pc_type":"lu",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"python",
    "pc_python_type":"FPCD.PCD",

    "pcd_Mp_ksp_type":"preonly",
    "pcd_Mp_pc_type":"lu",

    "pcd_Kp_ksp_type":"preonly",
    "pcd_Kp_pc_type":"lu",

    "pcd_Fp_mat_type":"matfree"
    }
  }
```

Figure 3.12: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: set of parameters for FGMRES solver with PCD preconditioning. Our implementation of PCD preconditioner is used.

[LSC] Finally, the third solvers is FGMRES solver with LSC preconditioning (2.66). Here, PETSc's implementation of LSC preconditioning is used. For solving the subsystems in LSC, we use LU factorization. The result set of parameters is as follows

```
LSC = {
  "ksp_type":"fgmres",
  "ksp_gmres_restart":600,
  "ksp_gmres_modifiedgramschmidt":True,
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"preonly",
    "pc_type":"lu",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"lsc",
    "lsc_pc_type":"lu",
    }
  }
```

Figure 3.13: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: set of parameters for FGMRES solver with LSC preconditioning. PETSc's implementation of LSC preconditioner is used.

Remark 25
Notice that we use FGMRES instead of GMRES. The reason is that FGMRES allows to use variable preconditioners while building up the searching subspace.

The figures (3.14), (3.15) represent the result velocity and the pressure distribution for unsteady lid-driven cavity problem in 2D.

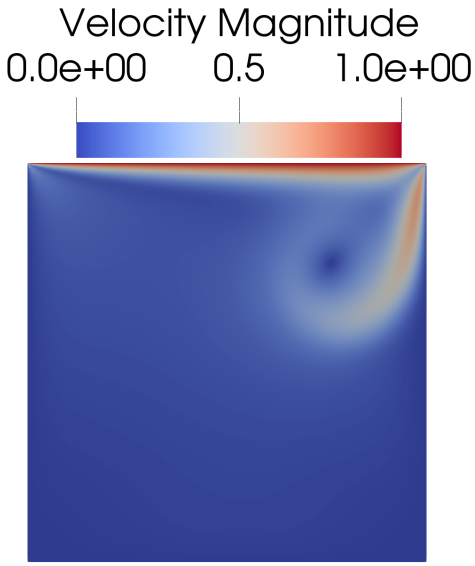


Figure 3.14: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: Velocity of the fluid in lid-driven cavity in 2D at the time $t = 3$.

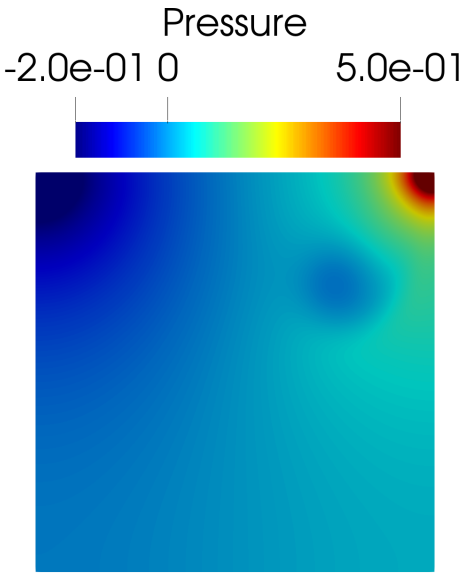


Figure 3.15: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: Pressure of the fluid in lid-driven cavity in 2D at the time $t = 3$.

Scalability

As you can see in the strong scaling graph (3.18) or in the table (3.3), all three solvers are bad strongly scalable. The peak of MUMPS and FGMRES solver with LSC preconditioning was reached by using 32 processors. The peak of FGMRES solver with PCD preconditioning was reached by using 64 processors. The graph of effectiveness (3.18) shows that both iterative solvers are almost equally effective.

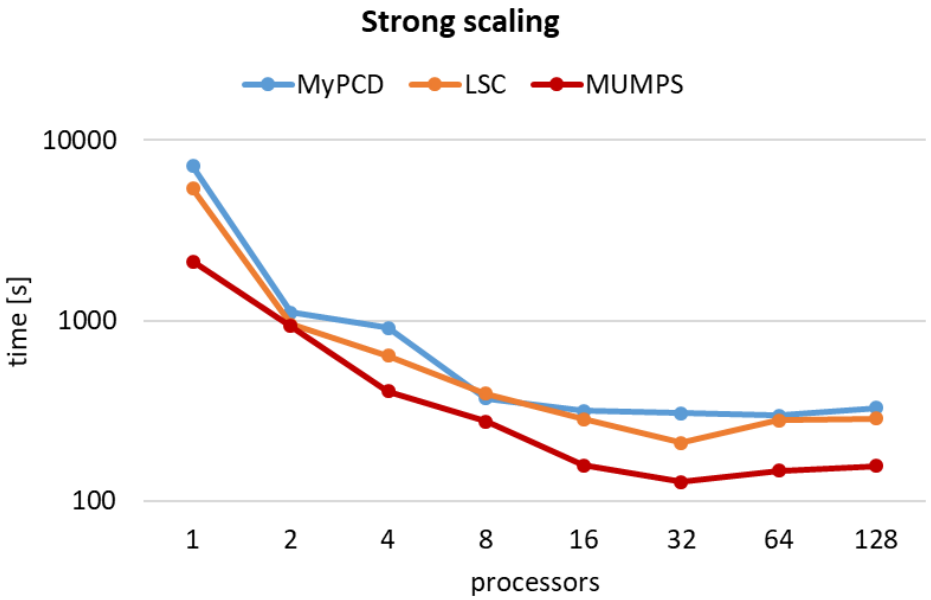


Figure 3.16: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: Strong scaling for 148 739 unknowns on 32 768 cells, logarithmic scale.

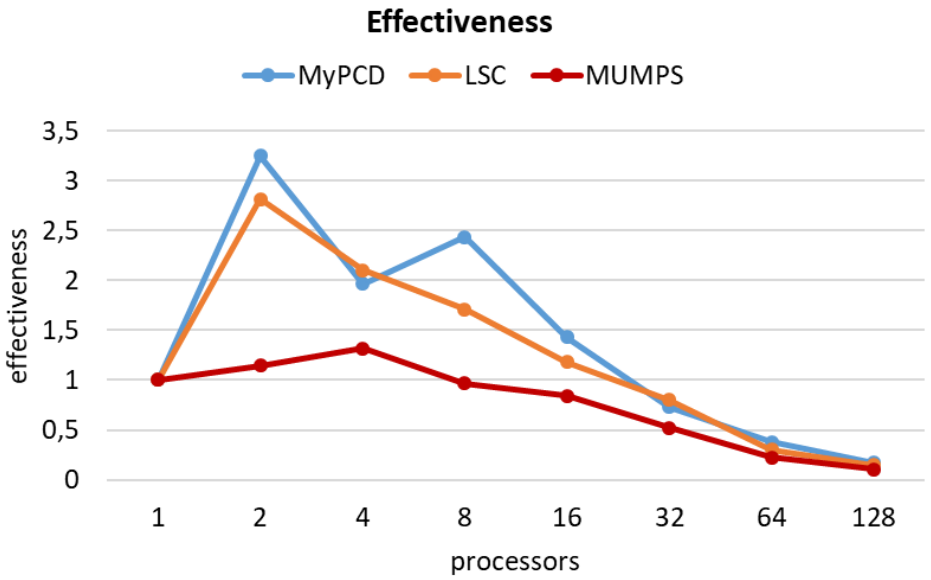


Figure 3.17: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: Graph of effectiveness.

As you can see in the figure (3.18) or in the table (3.4), all three solvers are very bad weakly scalable.

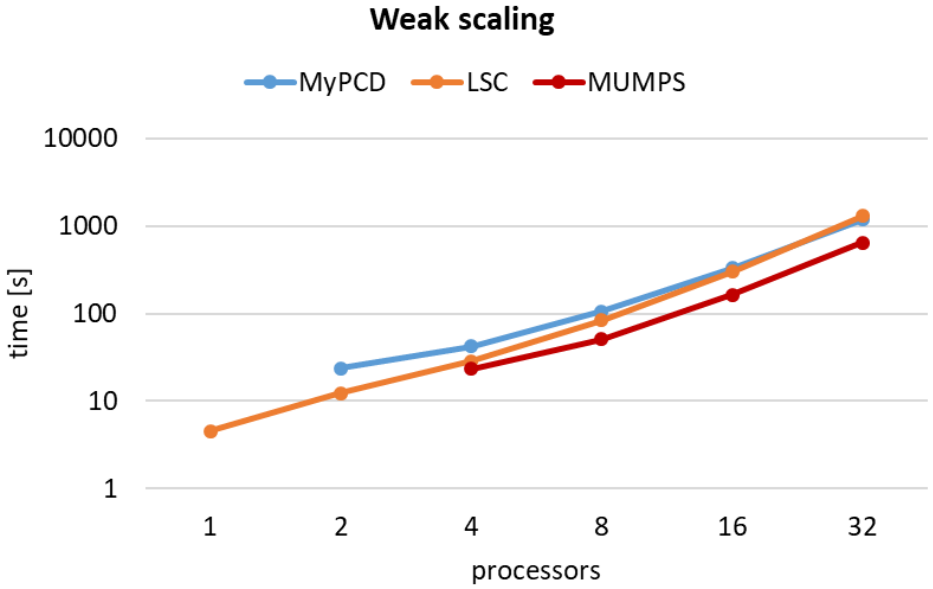


Figure 3.18: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: Weak scaling graph, logarithmic scale.

processors	parameter	LSC	MyPCD	MUMPS
1	time [s]	5 386	7 200	2 128
2	time [s]	955.8	1 108	931
4	time [s]	641.1	914.7	404.4
8	time [s]	394	369.4	275.7
16	time [s]	285.2	315.1	157.6
32	time [s]	210.7	307.6	127.2
64	time [s]	280.2	298.3	147.2
128	time [s]	287	329.3	156
	nonlinear it	900	900	900
	linear it	5345	8130	301

Table 3.3: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: Table of strong scaling data.

proc	cells	unknowns	parameter	LSC	MyPCD	MUMPS
1	8	659	time [s]	4.572	NOF	NOF
			linear it	3 009	NOF	NOF
2	16	2 467	time [s]	12.29	23.63	NOF
			linear it	2 502	5 401	NOF
4	32	9 539	time [s]	28.44	41.85	23.47
			linear it	3 261	5 998	900
8	64	37 507	time [s]	83.48	105.8	51.2
			linear it	4 142	6 934	900
16	128	148 739	time [s]	299.4	331.4	162.7
			linear it	5 345	8130	900
32	256	592 387	time [s]	1 308	1 189	641.7
			linear it	7 772	9031	900

Table 3.4: Lid-driven cavity problem in 2D - unsteady Navier-Stokes: Table of weak scaling data. NOF stand for reaching the maximum number of iterations.

3.1.4 Steady Navier-Stokes problem in 3D

The Navier-Stokes problem for steady flow in lid-driven cavity in 3D reads: Find the velocity $\vec{v}(\vec{x}) = (v_1(\vec{x}), v_2(\vec{x}), v_3(\vec{x}))$ and the pressure $p = p(\vec{x})$ such that

$$\begin{aligned} -\nu\ \Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega \\ \mathrm{div}(\vec{v}) &= 0\ \ in\ \Omega \\ \vec{v}|_{\Gamma_L} = \vec{v}|_{\Gamma_R} = \vec{v}|_{\Gamma_B} &= \vec{v}|_{\Gamma_{Ba}} = \vec{v}|_{\Gamma_F} = \vec{0} \\ \vec{v}|_{\Gamma_T} &= (V, 0, 0). \end{aligned} \tag{3.11}$$

The parameters in numerical simulation are taken as follows:

- the kinematic viscosity $\nu = 0.01$
- the body force $\vec{f} = \vec{0}$
- the tangential velocity $V = 1$

The triangulation of the domain in 3D case consists of tetrahedra. For discretization we use Taylor-Hood P^2/P^1 elements. Newton method is used for linearization purpose.

In section 2.3.1 we derived the discrete Navier-Stokes system for steady flow by using Newton method (or Picard method) as follows

$$\begin{pmatrix} \tilde{\mathbf{A}}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V}^k \\ \vec{P}^k \end{pmatrix} = \begin{pmatrix} \vec{c}^k \\ \vec{d}^k \end{pmatrix}, \tag{3.12}$$

where the matrix $\tilde{\mathbf{A}}^{(k)}$ is nonsymmetric. We also discussed about the effective preconditioning techniques for such systems. Effective exact preconditioner is of the form

$$\mathcal{P}_E = \begin{pmatrix} \tilde{\mathbf{A}}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}, \tag{3.13}$$

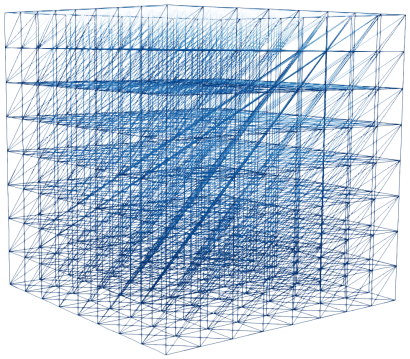


Figure 3.19: Triangulation $\mathcal{T}_h$ of the domain Ω for lid-driven cavity benchmark in 3D consisting of 1024 tetrahedra.

where the matrix $\tilde{\mathbf{A}}^{(k)}$ is defined as in (2.47). Two approximations of Schur complement matrix leading to effective preconditioning were derived - PCD and LSC preconditioning:

1. PCD: $\mathbf{S}^{-1} \approx -\mathbf{A}_P^{-1}\mathbf{F}_P\mathbf{M}_P^{-1}$
2. LSC: $\mathbf{S}^{-1} \approx -(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^\mathsf{T})^{-1}(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{F}\mathbf{M}_V^{-1}\mathbf{B}^\mathsf{T})(\mathbf{B}\mathbf{M}_V^{-1}\mathbf{B}^\mathsf{T})^{-1}$

Let us show how to solve the steady lid-driven cavity problem in 3D by using **Firedrake**.

Solver setting

Four solver settings will be showed for solving (3.8) in **Firedrake/PETSc**.

[MUMPS] The first solver is MUMPS. We use exactly the same set of parameters solver setting as in (3.4).

[PCD] The second solver is FGMRES solver with firedrake's implementation of PCD preconditioning. For inverting the matrix $\tilde{\mathbf{A}}^{(k)}$ in each Newton iteration, one V-cycle of algebraic multigrid method is used. In PCD preconditioning, the RICHRDSON method preconditioned by SOR method is used for solving the system with the pressure mass matrix and CG method preconditioned by block JACOBI (BJACOBI) method is used for solving the system with the pressure Laplacian matrix. The set of parameters in **Firedrak/PETSc** is as follows

```
PCD = {
  "ksp_type":"fgmres",
  "ksp_gmres_restart":600,
  "ksp_gmres_modifiedgramschmidt":True,
  "mat_type":"matfree",
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"preonly",
    "pc_type":"python",
    "pc_python_type":"firedrake.AssembledPC",
    "assembled_pc_type":"hypre",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"python",
    "pc_python_type":"firedrake.PCDPC",

    "pcd_Mp_ksp_type":"richardson",
    "pcd_Mp_ksp_max_it":5,
    "pcd_Mp_pc_type":"sor",

    "pcd_Kp_ksp_type":"cg",
    "pcd_Kp_ksp_type":5,
    "pcd_Kp_pc_type":"bjacobi",

    "pcd_Fp_mat_type":"matfree"
    }
  }
```

Figure 3.20: Lid-driven cavity problem in 3D - steady Navier-Stokes: set of parameters for FGMRES solver with PCD preconditioning. Firedrake's implementation of PCD preconditioner is used.

[MyPCD] The third solver is FGMRES solver with our implementation of PCD. The set of parameters is the same as in (3.20).

[LSC] The fourth solver is FGMRES solver with LSC preconditioning. Here, the matrix $\tilde{\mathbf{A}}^{(k)}$ is inverted by one V-cycle of algebraic multigrid method and the subsystems in LSC preconditioning are solved by LU factorization.

```
LSC = {
  "ksp_type":"fgmres",
  "ksp_gmres_restart":600,
  "ksp_gmres_modifiedgramschmidt":True,
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"preonly",
    "pc_type":"hypre",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"lsc",
    "lsc_pc_type":"lu",
    }
  }
```

Figure 3.21: Lid-driven cavity problem in 3D - steady Navier-Stokes: set of parameters for FGMRES solver with LSC preconditioning. PETSc's implementation of LSC preconditioner is used.

The following figures (3.22), (3.23) present the velocity and the pressure distribution for steady lid-driven cavity problem in 3D.

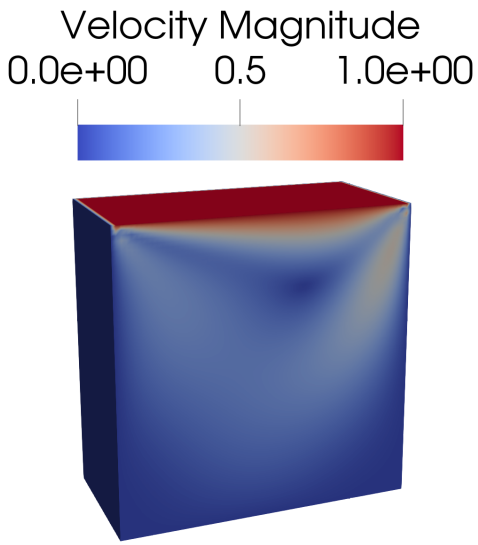


Figure 3.22: Lid-driven cavity problem in 3D - steady Navier-Stokes: Velocity of the fluid in lid-driven cavity in 3D.

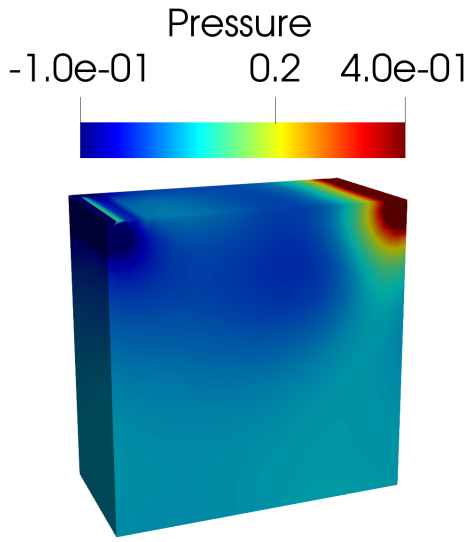


Figure 3.23: Lid-driven cavity problem in 3D - steady Navier-Stokes: Pressure of the fluid in lid-driven cavity in 3D.

Scalability

All four solvers are reasonably strongly scalable as you can see in the figure (3.24) or in the table (3.5). The FGMRES solver with PCD preconditioning is faster as FGMRES solver with LSC preconditioning, whereas the MUMPS solver is the worst. On the other hand, the MUMPS solver is the most effective as you can see in the figure (3.25). All four solvers are poor weakly scalable - see figure (3.26) or the table (3.6).

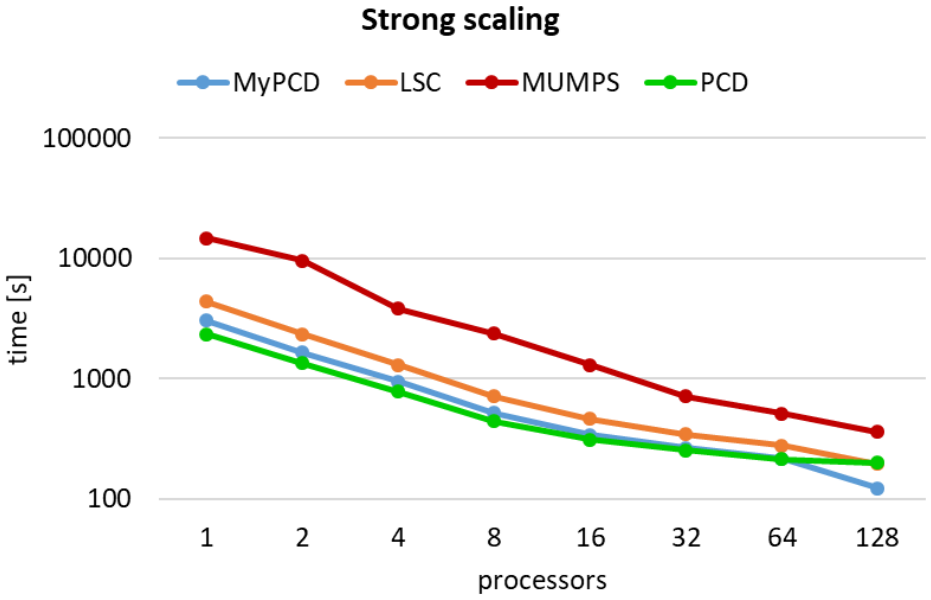


Figure 3.24: Lid-driven cavity problem in 3D - steady Navier-Stokes: Strong scaling for 859 812 unknowns on 196 608 cells, logarithmic scale.

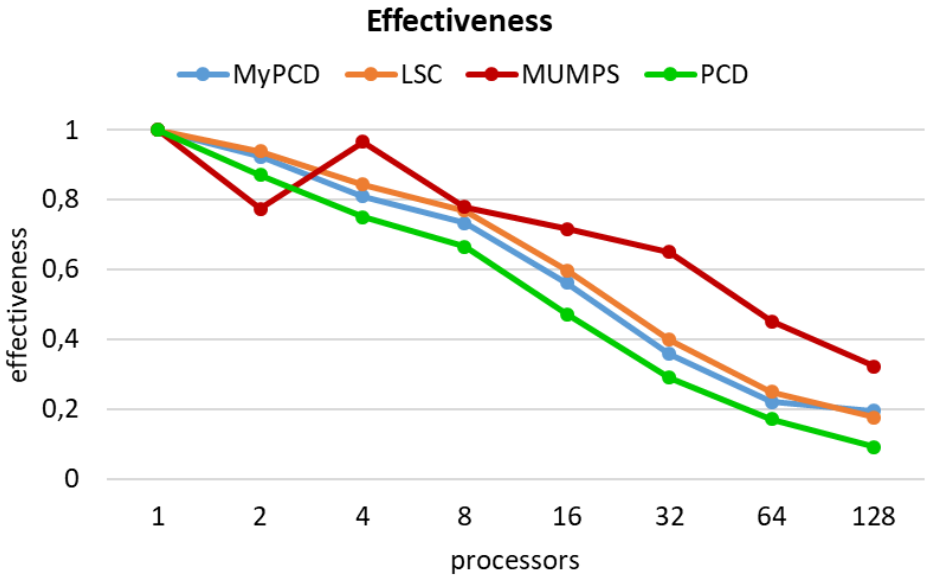


Figure 3.25: Lid-driven cavity problem in 3D - steady Navier-Stokes: Graph of effectiveness.

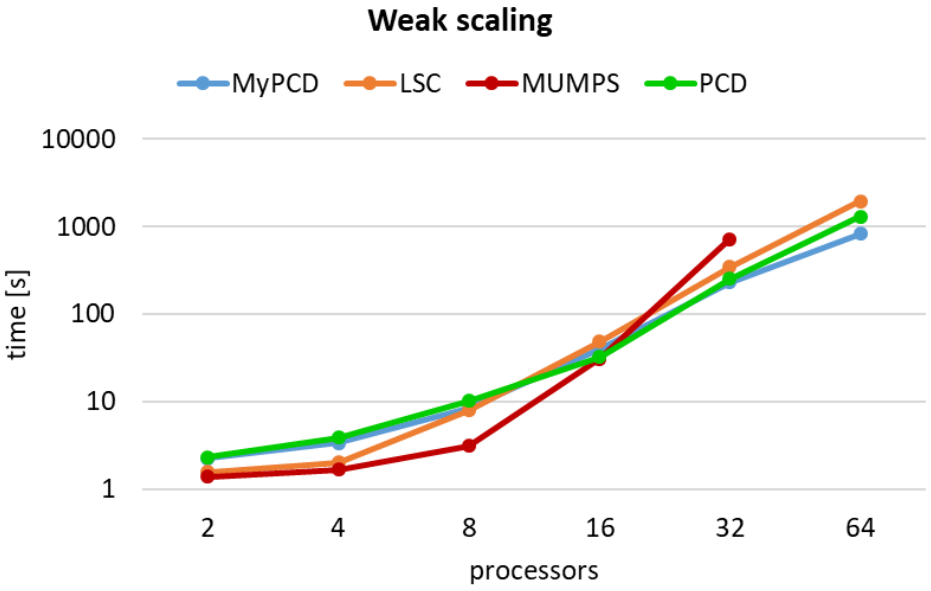


Figure 3.26: Lid-driven cavity problem in 3D - steady Navier-Stokes: Weak scaling graph, logarithmic scale.

processors	parameter	PCD	LSC	MyPCD	MUMPS
1	time [s]	2 346	4 407	3 056	14 830
2	time [s]	1 347	2 346	1 655	9 572
4	time [s]	782	1 305	943.7	3 835
8	time [s]	440.7	717.2	521.2	2 377
16	time [s]	311.2	461.4	340.2	1 295
32	time [s]	253.4	346.1	267	713.9
64	time [s]	214.6	277.2	216.4	513.5
128	time [s]	201	196.4	123.1	359.8
	nonlinear it	5	5	5	5
	linear it	358	626	455	5

Table 3.5: Lid-driven cavity problem in 3D - steady Navier-Stokes: Table of strong scaling data.

proc	cells	unknowns	parameter	PCD	LSC	MyPCD	MUMPS
2	48	402	time [s]	2.341	1.574	2.259	1.385
			linear it	165	140	220	5
4	348	2 312	time [s]	3.784	2.016	3.395 s	1.668
			linear it	305	177	235 it	5
8	3 072	15 468	time [s]	10.18	7.935	8.497 s	3.142
			linear it	446	238	250	5
16	24 576	112 724	time [s]	32.45	48.73	39.76	30.64
			linear it	518	389	256	5
32	196 608	859 812	time [s]	253.4	346.1	229.3	713.9
			linear it	783	644	250	5
64	1 572 864	6 714 692	time [s]	1 297	1 954	829	MEM
			linear it	1 016	820	180	MEM

Table 3.6: Lid-driven cavity problem in 3D - steady Navier-Stokes: Table of weak scaling data. MEM stands for out of memory range error.

3.1.5 Unsteady Navier-Stokes problem in 3D

Let us describe the flow in lid-driven cavity in 3D by unsteady Navier-Stokes equations. The problem reads: Find the velocity $\vec{v} = (v_1(\vec{x},\ t),\ \vec{v}_2(\vec{x},\ t),\ v_3(\vec{x},\ t))$ and the pressure $p = p(\vec{x},\ t)$ such that

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} - \nu\ \Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p &= \vec{f}\ in\ \Omega \\ \mathrm{div}(\vec{v}) &= 0\ in\ \Omega \\ \vec{v}|_{\Gamma_L} = \vec{v}|_{\Gamma_R} = \vec{v}|_{\Gamma_B} = \vec{v}|_{\Gamma_T} &= \vec{v}|_{\Gamma_F} = \vec{v}|_{\Gamma_{Ba}} = \vec{0} \\ \vec{v}|_{\Gamma_T} &= (V, 0, 0). \end{aligned} \tag{3.14}$$

The parameters in numerical simulation are taken as follows:

- the kinematic viscosity $\nu = 0.001$
- the body force $\vec{f} = \vec{0}$

- the time step $dt = 0.01$
- the initial time $T_0 = 0.0$
- the end time $T_{end} = 3$
- the tangential velocity $V = 1$

Triangulation of the domain Ω consists of tetrahedra as in the 3D steady case - see (3.19). Taylor-Hood elements P^2/P^1 are used for the discretization in space and the Crank-Nicholson scheme is used for the discretization in time. For linearization, we use Newton method. As we discussed in section 2.3.2, the following saddle-point system with nonsymmetric block $\tilde{\mathbf{A}}^{(k)}$ is solved on each time level

$$\begin{pmatrix} \tilde{\mathbf{A}}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V}^k \\ \vec{P}^k \end{pmatrix} = \begin{pmatrix} \vec{c}^k \\ \vec{d}^k \end{pmatrix}. \tag{3.15}$$

The effective exact preconditioner for such system has a form

$$\mathcal{P}_E = \begin{pmatrix} \tilde{\mathbf{A}}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}, \tag{3.16}$$

as was discussed in section 1.5.2.

Solver parameters

Two iterative solvers will be presented for solving (3.15). Here, MUMPS solver is not as effective as iterative solvers and therefore it is not present.

[MyPCD] The first solver is FGMRES solver with our implementation of PCD preconditioning. For inverting the matrix $\tilde{\mathbf{A}}^{(k)}$, we use GMRES method preconditioned by block JACOBI, with the maximum number of iteration set to 10. For inverting the Schur complement matrix $\mathbf{S}$, we use GMRES method with PCD preconditioning. As we use PCD preconditioning, we need to solve system with the pressure mass matrix $\mathbf{M}_P$ and the system with the pressure Laplacian matrix $\mathbf{A}_P$. The system with the pressure mass matrix can be solved by using RICHARDSON method preconditioned by SOR. For solving the system with the pressure Laplacian matrix, we use one V-cycle of algebraic multigrid method.

```
MyPCD = {
  "ksp_type":"fgmres",
  "ksp_rtol": 1e-3,
  "ksp_gmres_restart":600,
  "ksp_gmres_modifiedgramschmidt":True,
  "mat_type":"matfree",
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"gmres",
    "ksp_gmres_restart": 100,
    "ksp_max_it":10,
    "pc_type":"python",
    "pc_python_type":"firedrake.AssembledPC",
    "assembled_pc_type":"bjacobi",
    },

  "fieldsplit_1":{
    "ksp_type":"gmres",
    "ksp_rtol": 1e-1,
    "ksp_max_it": 10,
    "pc_type":"python",
    "pc_python_type":"FPCD.PCDPC",

    "pcd_Mp_ksp_type":"richardson",
    "pcd_Mp_ksp_max_it":5,
    "pcd_Mp_pc_type":"sor",

    "pcd_Kp_ksp_type":"preonly",
    "pcd_Kp_pc_type":"hypre",
    "pcd_Kp_pc_hypre_type":"boomeramg",
    "pcd_Kp_pc_hypre_boomeramg_strong_threshold": 0.2,
    "pcd_Kp_pc_hypre_boomeramg_P_max":2,

    "pcd_Fp_mat_type":"matfree"
    }
  }
```

Figure 3.27: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: set of parameters for FGMRES solver with PCD preconditioning. Our implementation of PCD preconditioner is used.

[LSC] The second solver is FGMRES solver with LSC preconditioning. Here we use GMRES method preconditioned by BJACOBI for inverting the matrix $\tilde{\mathbf{A}}^{(k)}$. The Schur complement matrix $\mathbf{S}$ is replace by LSC preconditioning. For solving the subsystems connected with LSC preconditioning, GMRES method preconditioned by one V-cycle of algebraic multigrid method is used.

```
LSC = {
  "ksp_type":"fgmres",
  "ksp_rtol":1e-3,
  "ksp_gmres_restart":400,
  "ksp_max_it":400,
  "ksp_gmres_modifiedgramschmidt":True,
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"gmres",
    "ksp_gmres_restart":50,
    "pc_type":"bjacobi",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"lsc",
    "lsc_pc_type":"ksp",
    "lsc_pc_ksp_type":"gmres",
    "lsc_pc_ksp_rtol":1e-2,
    "lsc_pc_ksp_max_it":50,
    "lsc_pc_ksp_gmres_restart":50,
    "lsc_pc_pc_type":"hypre",
    "lsc_pc_pc_hypre_type":"boomeramg",
    "lsc_pc_pc_hypre_boomeramg_strong_threshold": 0.2,
    "lsc_pc_pc_hypre_boomeramg_P_max":2,
    }
  }
```

Figure 3.28: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: set of parameters for FGMRES solver with LSC preconditioning. PETSc's implementation of LSC preconditioner is used.

The following figures (3.29), (3.30) present the velocity and the pressure distribution for unsteady lid-driven cavity problem in 3D.

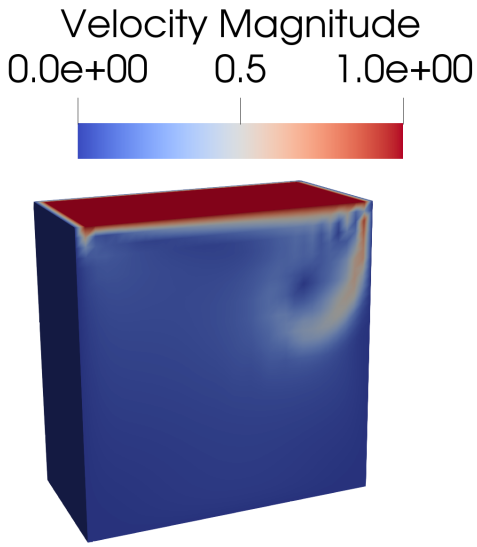


Figure 3.29: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: Velocity of the fluid in lid-driven cavity in 3D at time $t = 3$.

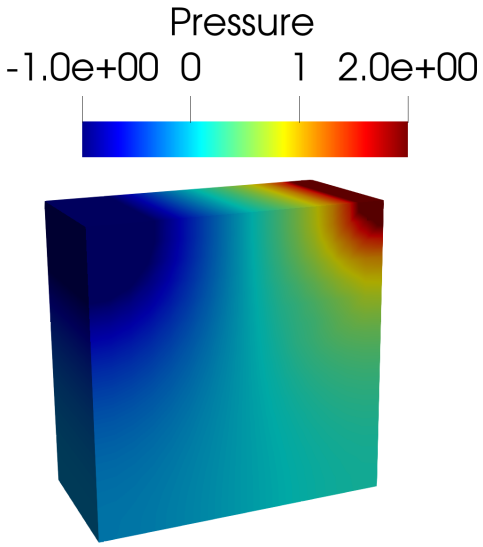


Figure 3.30: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: Pressure of the fluid in lid-driven cavity in 3D at time $t = 3$.

Scalability

Let us discuss the scalability of mentioned iterative solvers. As you can see in the figure (3.31) or in the table (3.7), both solvers are reasonably strongly scalable. Notice that the FGMRES solver with LSC preconditioning is faster as FGMRES solver with PCD preconditioning. Both solvers are almost equally effective as you can see in the figure (3.32). As can be seen in the figure (3.33) or in the table (3.8), both solvers are very bad weakly scalable.

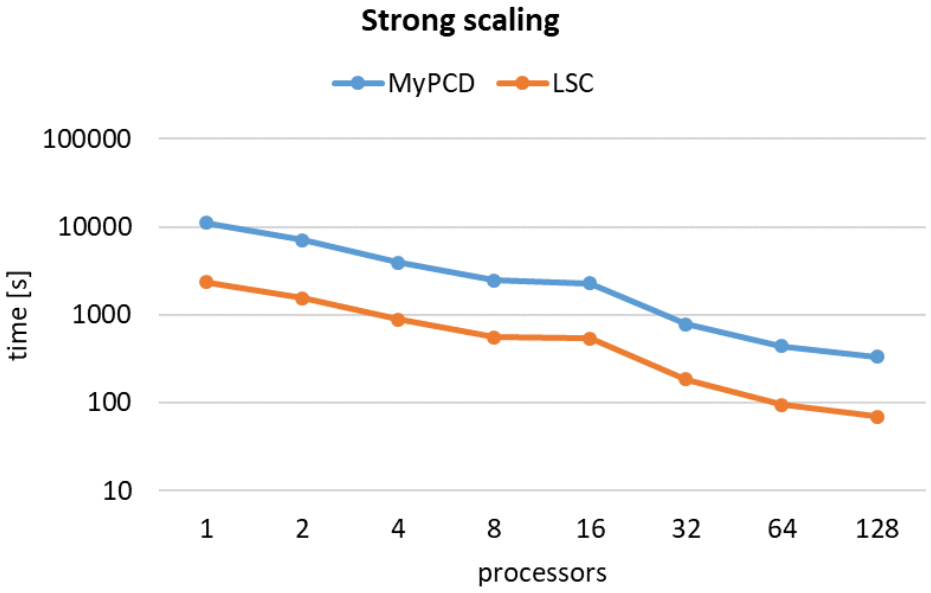


Figure 3.31: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: Strong scaling for 112 724 unknowns on 24 576 cells, logarithmic scale.

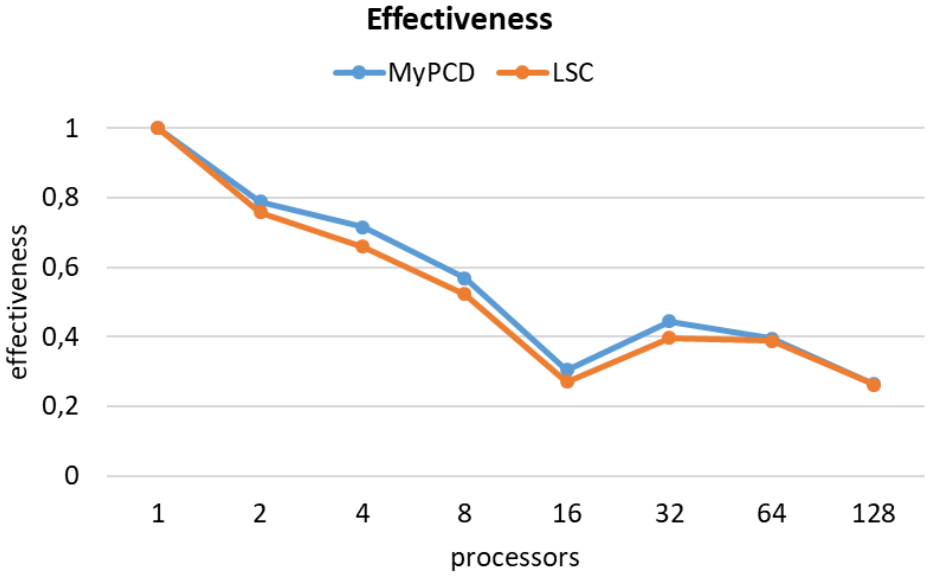


Figure 3.32: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: Graph of effectiveness.

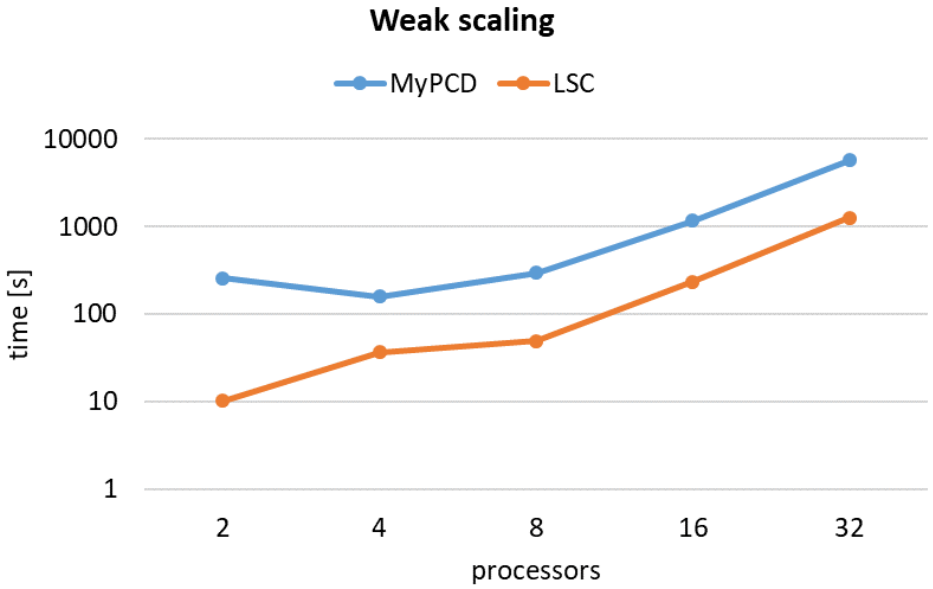


Figure 3.33: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: Weak scaling graph, logarithmic scale.

processors	parameter	LSC	MyPCD	MUMPS
1	time [s]	2 345	11 210	
2	time [s]	1 547	7 101	
4	time [s]	888.3	3 916	
8	time [s]	560.1	2 465	
16	time [s]	541.1	2 303	6 137
32	time [s]	184.8	789.3	
64	time [s]	94.29	444.2	
128	time [s]	69.93	331.3	
	nonlinear it	301	301	
	linear it	3 310	3 273	

Table 3.7: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: Table of strong scaling data. The time for MUMPS on 16 processors is computed for comparing

proc	cells	unknowns	parameter	LSC	MyPCD
2	48	402	time [s]	10.22	256.4
			linear it	2 710	2 893
4	348	2 312	time [s]	36.4	157.7
			linear it	3 310	3 037
8	3 072	15 468	time [s]	49.35	296 s
			linear it	3315	3010
16	24 576	112 724	time [s]	232.7	1 164
			linear in	3 313	3 288
32	196 608	859 812	time [s]	1 268	5 709
			linear it	3 310	3 265

Table 3.8: Lid-driven cavity problem in 3D - unsteady Navier-Stokes: Table of weak scaling data.

3.2 Flow around a cylinder

Let us introduce the second benchmark on which the numerical simulations were performed. As the benchmark we consider the flow around a fixed cylinder which was defined by Schäfer and Turek in Schäfer et al. [1996]. This benchmark simulates the flow of a fluid in a channel with an circular obstacle. The flow will be described by unsteady Navier-Stokes equations. We distinguish the simulations in 2D and 3D geometry.

3.2.1 Geometry and the boundary conditions

Let us first turn our attention to the geometry in 2D. The channel in 2D is rectangular with length 2.5 and height 0.41. The center of the circular obstacle (circle) is situated at the point $(0.5,\ 0.2)$ with the radius $R = 0.05$. The domain of the channel without a circular obstacle is given by $\Omega = [0,\ 2.5] \times [0,\ 0.41] \setminus B_R(0.5,\ 0.2)$. Let us denote the walls of the channel as follows

- $\Gamma_I = 0 \times [0,\ 0.41]$ the inflow wall
- $\Gamma_O = 2.5 \times [0,\ 0.41]$ the outflow wall
- Γ_R - the remaining walls of the channel
- $\Gamma_C = \partial B_R(0.5,\ 0.2)$ the surface of the circle

The geometry of the channel with circular obstacle can be displayed as follows

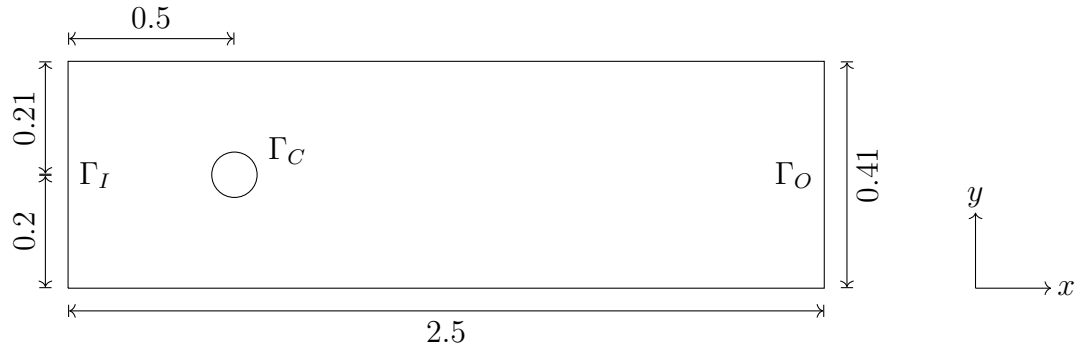


Figure 3.34: Flow around cylinder problem: the geometry in 2D

Let us discuss the boundary conditions. On the inflow wall Γ_I we prescribe a parabolic inflow profile

$$\vec{v}|_{\Gamma_I} = \left(\frac{4Vy(0.41 - y)}{0.41^2},\ 0 \right) \tag{3.17}$$

with the maximum velocity $V = 1.5$. The boundary condition on the outflow wall Γ_O is of type do-nothing

$$\nu \frac{\partial \vec{v}}{\partial n} - p\vec{n} = \vec{0}, \tag{3.18}$$

where $\vec{n}$ is the unit outer vector and the constant $\nu > 0$ denotes the kinematic viscosity. On the remaining walls Γ_R and on the surface of the circle Γ_C no-slip boundary conditions

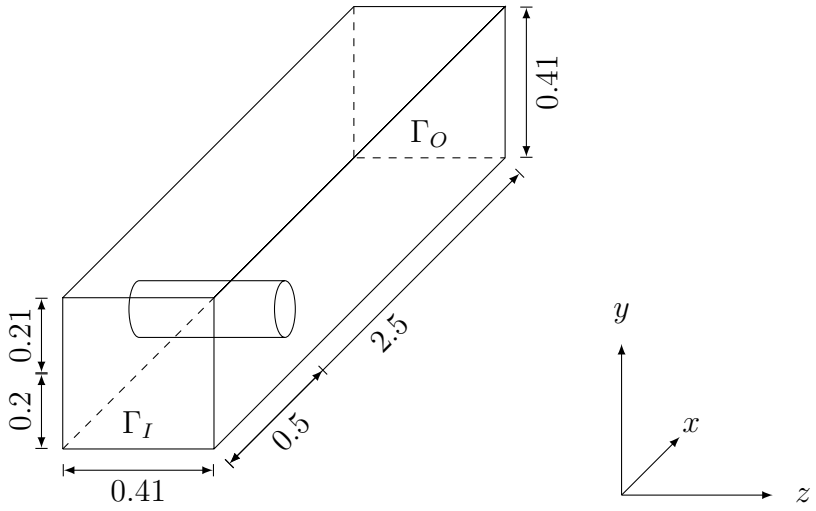
$$\vec{v}|_{\Gamma_R} = \vec{v}|_{\Gamma_C} = \vec{0} \tag{3.19}$$

are prescribed.

In 3D case, the flow is considered in the domain $\Omega = [0,\ 2.5] \times [0,\ 0.41] \times [0,\ 0.41] \backslash (B_R(0.5,\ 0.2, 0) \times 0.41)$ which represents the channel without the circular obstacle (cylinder). Similarly as in the 2D case we denote

- $\Gamma_I = 0 \times [0,\ 0.41] \times [0,\ 0.41]$ the inflow wall
- $\Gamma_O = 2.5 \times [0,\ 0.41] \times [0,\ 0.41]$ the outflow wall
- Γ_R - the remaining walls of the channel
- $\Gamma_C = \partial B_R(0.5,\ 0.2, 0)$ the surface of the cylinder

The visualization of the geometry in 3D:



Let us introduce the boundary conditions for our problem in 3D. On the inflow wall Γ_I we prescribe a parabolic inflow profile

$$\vec{v}|_{\Gamma_I} = \left(\frac{16Vyz(0.41 - y)(0.41 - z)}{0.41^4},\ 0,\ 0 \right) \tag{3.20}$$

with the maximum velocity $V = 2.25$. On the outflow wall Γ_O do-nothing boundary condition is prescribed

$$\nu \frac{\partial \vec{v}}{\partial n} - p\vec{n} = \vec{0}, \tag{3.21}$$

where $\vec{n}$ is the unit outer vector. On the remaining walls Γ_R and on the surface of the cylinder Γ_C we consider no-slip boundary conditions

$$\vec{v}|_{\Gamma_R} = \vec{v}|_{\Gamma_C} = \vec{0} \tag{3.22}$$

3.2.2 Unsteady Navier-Stokes problem in 2D

Let us introduce the unsteady Navier-Stokes problem describing the flow of the fluid around the circular obstacle in the channel in 2D: Find the velocity $\vec{v}(\vec{x}) = (v_1(\vec{x},\ t),\ v_2(\vec{x},\ t))$ and the pressure $p = p(\vec{x},\ t)$ such that

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} - \nu\ \Delta \vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega \\ \mathrm{div}(\vec{v}) &= 0\ \ in\ \Omega \\ \vec{v}|_{\Gamma_R} &= \vec{v}|_{\Gamma_C} = \vec{0} \\ \vec{v} &= \left(\frac{4Vy(0.41-y)}{0.41^2}, 0\right)\ \ on\ \Gamma_I \\ \nu\frac{\partial \vec{v}}{\partial n} - p\vec{n} &= \vec{0}\ \ on\ \Gamma_O. \end{aligned} \tag{3.23}$$

The parameters in numerical simulation are taken as follows:

- the kinematic viscosity $\nu = 0.001$
- the body force $\vec{f} = \vec{0}$
- the time step $dt = 0.01$
- the initial time $T_0 = 0.0$
- the end time $T_{end} = 5$
- the maximum velocity $V = 1.5$

For the discretization of (3.23) in the space, we use inf-sup stable P^2/P^1 Taylor-Hood elements. Crank-Nicholson scheme is considered for the discretization in the time. For the linearization, we use Newton method.

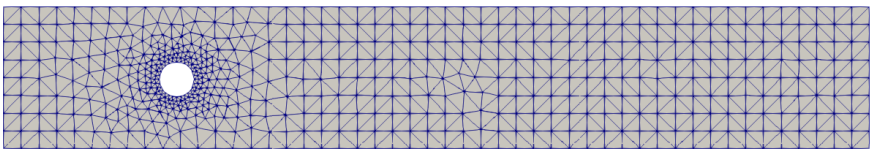


Figure 3.35: Triangulation $\mathcal{T}_h$ of the domain Ω for flow in the channel problem in 2D.

As in the section 3.1.3, on each time level we need to solve the saddle-point system

$$\begin{pmatrix} \hat{\mathbf{A}}^{(k+1)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V}^{k+1} \\ \vec{P}^{k+1} \end{pmatrix} = \begin{pmatrix} \vec{c}^{k} \\ \vec{d}^{k} \end{pmatrix} \iff \mathcal{A}^{(k+1)}\vec{x} = \vec{b}, \tag{3.24}$$

where the block $\hat{\mathbf{A}}^{(k+1)}$ is nonsymmetric. The effective exact preconditioner for (3.24) has a form

$$\mathcal{P}_E = \begin{pmatrix} \hat{\mathbf{A}}^{(k+1)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}. \tag{3.25}$$

Solver setting

Let us discuss the setting of the solver parameters for solving (3.24) in **Firedrake/PETSc**. Three solvers will be presented.

[MUMPS] The first solver is MUMPS. Set of parameters for MUMPS in **Firedrake/PETSc** is the same as in (3.4).

[MyPCD] The second solver is FGMRES solver with our implementation of PCD preconditioning. In this benchmark we use different boundary conditions for construction the pressure Laplacian matrix $\mathbf{A}_P$. Here we consider homogeneous Dirichlet boundary condition on the inlet, i.e., on the inflow wall Γ_I. LU factorization is used for inverting the matrix $\hat{\mathbf{A}}^{(k+1)}$ and also for solving the systems connected with the PCD preconditioning. The set of parameters is as follows

```
MyPCD = {
  "ksp_type":"fgmres",
  "ksp_gmres_restart":800,
  "ksp_gmres_modifiedgramschmidt":True,
  "mat_type":"matfree",
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"preonly",
    "pc_type":"python",
    "pc_python_type":"firedrake.AssembledPC",
    "assembled_pc_type":"lu",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"python",
    "pc_python_type":"firedrake.PCDPC",

    "pcd_Mp_ksp_type":"preonly",
    "pcd_Mp_pc_type":"lu",

    "pcd_Kp_ksp_type":"preonly",
    "pcd_Kp_pc_type":"lu",

    "pcd_Fp_mat_type":"matfree"
    }
  }
```

Figure 3.36: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: set of parameters for FGMRES solver with PCD preconditioning. Our implementation of PCD preconditioner is used.

[LSC] The third solver is FGMRES solver with LSC preconditioning. Here we use PETSc's implementation of LSC preconditioning. Again, LU factorization is used for inverting the matrix $\hat{\mathbf{A}}^{(k+1)}$ and for solving the systems connected with LSC preconditioning.

```
LSC = {
  "ksp_type":"fgmres",
  "ksp_gmres_restart":150,
  "ksp_gmres_modifiedgramschmidt":True,
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"preonly",
    "pc_type":"lu",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"lsc",
    "lsc_pc_type":"lu",
    }
  }
```

Figure 3.37: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: set of parameters for FGMRES solver with LSC preconditioning. PETSc's implementation of LSC preconditioner is used.

Figures (3.38) and (3.39) present the velocity and the pressure distribution for unsteady flow around the cylinder problem in 2D.

Scalability

As can be seen in the figure (3.40) or in the table (3.9) below, strong scalability is not achieved by any of the three solvers. The peak of performance of both iterative solvers (MyPCD and LSC) was reached by using 4 processors. The MUMPS solver was the fastest, whereas the FGMRES solver with our implementation of PCD was the slowest. The effectiveness of all three solvers is almost the same - see (3.42). FGMRES solver with our PCD preconditioning is reasonably weakly scalable as you can see in the figure (3.42) or in the table (3.10).

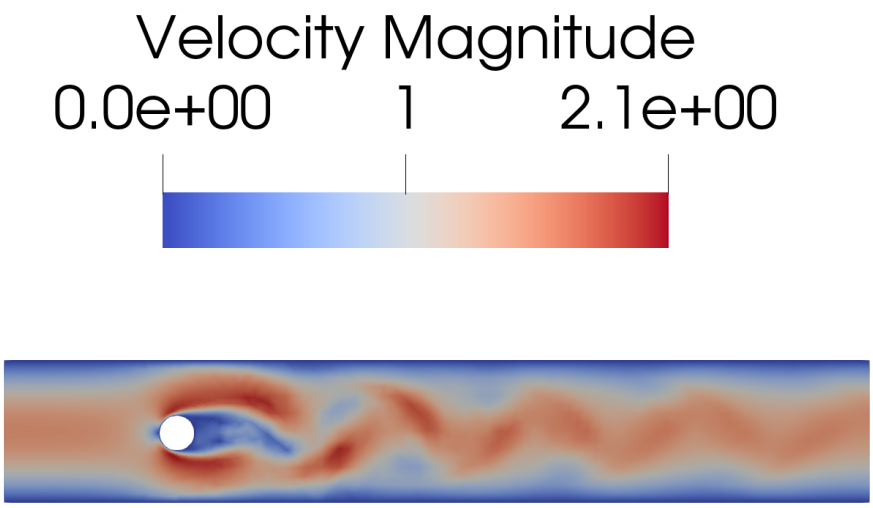


Figure 3.38: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: Velocity of the fluid in channel in 2D at the time $t = 5$.

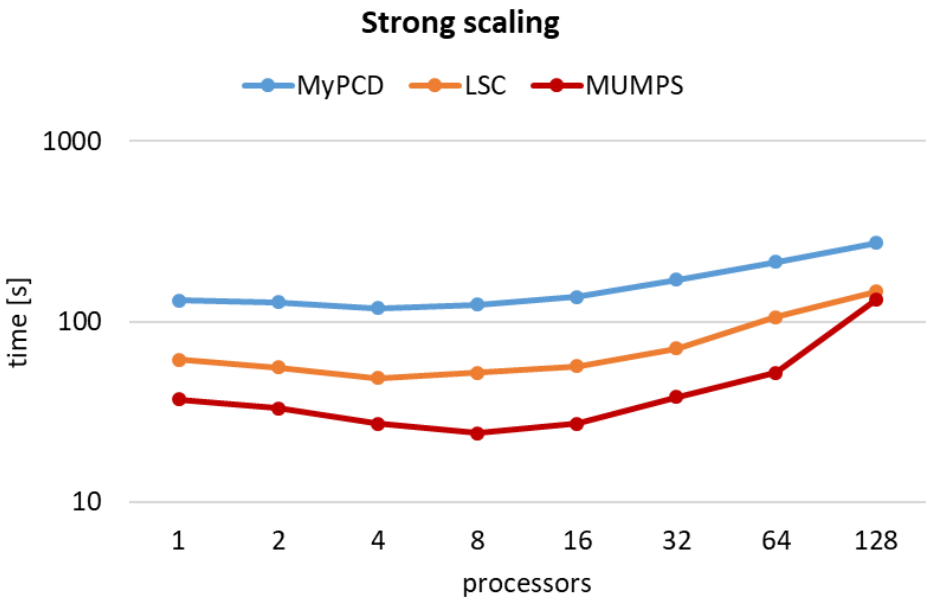


Figure 3.40: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: Strong scaling for 5 359 unknowns on 565 cells, logarithmic scale.

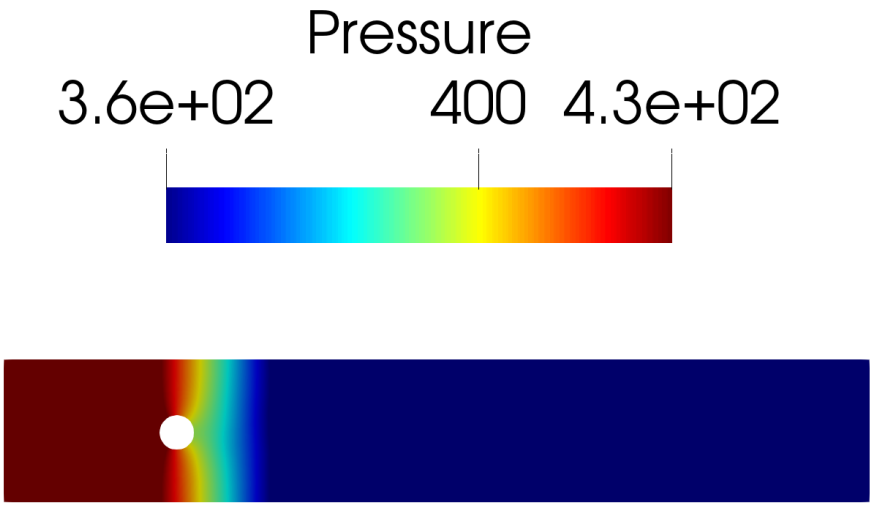


Figure 3.39: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: Pressure of the fluid in channel in 2D.

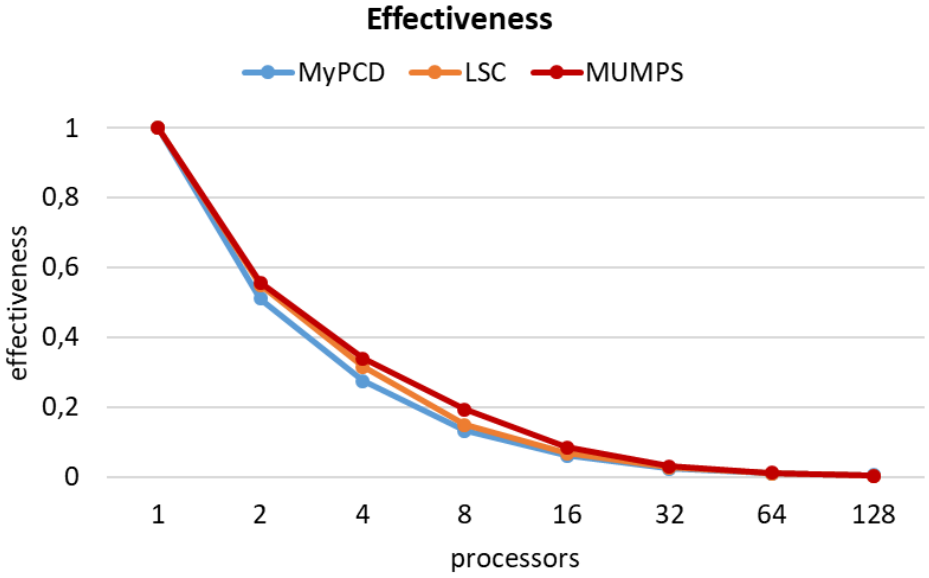


Figure 3.41: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: Strong scaling.: Graph of effectiveness.

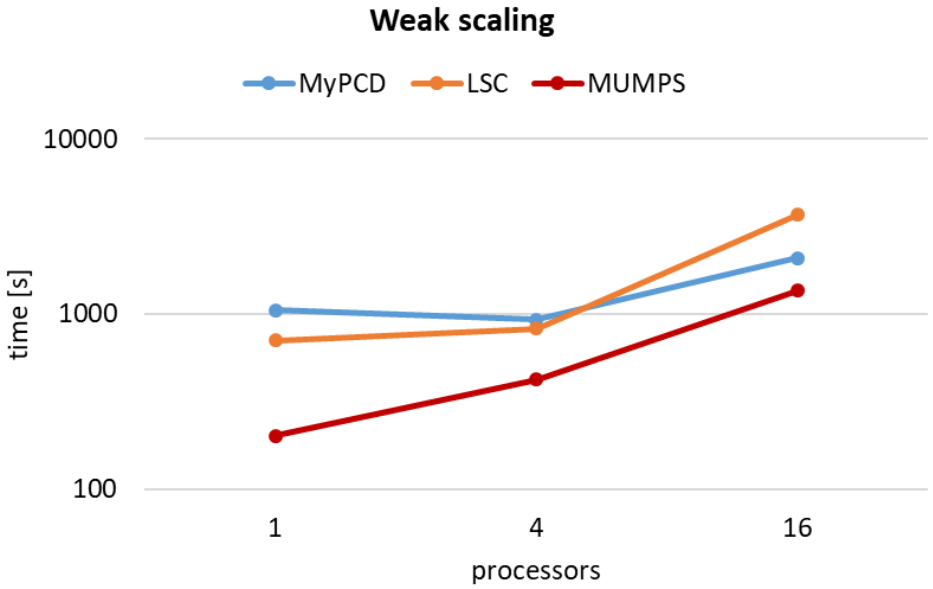


Figure 3.42: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: Weak scaling graph, logarithmic scale.

processors	parameter	MyPCD	LSC	MMUMPS
1	time [s]	131	61.44	36.97
2	time [s]	128.3	55.89	33.19
4	time [s]	119.2	48.73	27.2
8	time [s]	124.5	51.9	24
16	time [s]	136.7	56.63	27.18
32	time [s]	171.2	71.13	38.19
64	time [s]	213.9	105.8	52.05
128	time [s]	272.6	146.4	132.5
	nonlinear it	501	501	501
	linear it	27 440	11 600	501

Table 3.9: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: Table of strong scaling data.

proc	cells	unknowns	parameter	MyPCD	LSC	MUMPS
1	4 448	20 726	time [s]	1 053	704.7	201.3
			linear it	25 656	18 990	501
4	17 792	81 474	time [s]	926 s	828.6	421.6
			linear it	32 150	33 124	501
16	71 168	323 096	time [s]	2 088	3 700	1 363
			linear it	39 073	72 755	501

Table 3.10: Flow around the cylinder problem in 2D - unsteady Navier-Stokes: Table of weak scaling data.

3.2.3 Unsteady Navier-Stokes problem in 3D

Let us describe the flow around the cylinder in 3D by unsteady Navier-Stokes equations. The problem reads: Find the velocity $\vec{v} = (v_1(\vec{x},\ t),\ \vec{v}_2(\vec{x},\ t),\ v_3(\vec{x},\ t))$ and the pressure $p = p(\vec{x},\ t)$ such that

$$\begin{aligned} \frac{\partial \vec{v}}{\partial t} - \nu\ \Delta\vec{v} + (\vec{v}\cdot\nabla)\vec{v} + \nabla p &= \vec{f}\ \ in\ \Omega \\ \mathrm{div}(\vec{v}) &= 0\ \ in\ \Omega \\ \vec{v}|_{\Gamma_R} = \vec{v}|_{\Gamma_C} &= \vec{0} \\ \vec{v}|_{\Gamma_I} &= \left(\frac{16Vyz(0.41-y)(0.41-z)}{0.41^4},\ 0,\ 0\right). \end{aligned} \tag{3.26}$$

The parameters in numerical simulation are taken as follows:

- the kinematic viscosity $\nu = 0.001$
- the body force $\vec{f} = \vec{0}$
- the time step $dt = 0.01$
- the initial time $T_0 = 0.0$
- the end time $T_{end} = 5$
- the maximal velocity $V = 2.25$

Triangulation of the domain Ω consists of tetrahedra. Taylor-Hood elements P^2/P^1 are used for the discretization in space and the Crank-Nicholson scheme is used for the discretization in time. The problem is linearized by Newton method. As we discussed in section 2.3.2, the following saddle-point system with nonsymmetric block $\tilde{\mathbf{A}}^{(k)}$ is solved on each time level

$$\begin{pmatrix} \tilde{\mathbf{A}}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{V}^k \\ \vec{P}^k \end{pmatrix} = \begin{pmatrix} \vec{c}^k \\ \vec{d}^k \end{pmatrix}. \tag{3.27}$$

The effective exact preconditioner for such system has a form

$$\mathcal{P}_E = \begin{pmatrix} \tilde{\mathbf{A}}^{(k)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}. \tag{3.28}$$

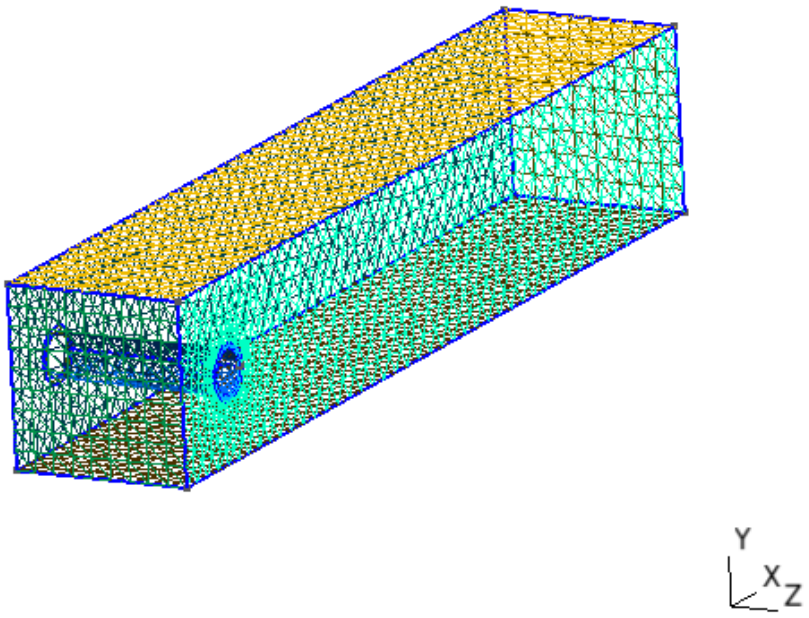


Figure 3.43: Triangulation $\mathcal{T}_h$ of the domain Ω for flow around cylinder problem in 3D.

Solver parameters

Let us introduce two sets of parameters for solving (3.27) by iterative solvers.

[MyPCD] The first solver is FGMRES solver with our implementation of PCD preconditioning. The matrix $\tilde{\mathbf{A}}^{(k)}$ is inverted by GMRES method preconditioned by block JACOBI. For inverting the Schur complement matrix $\mathbf{S}$, we use GMRES method with PCD preconditioning. In PCD preconditioning, the system with the pressure mass matrix is solved by using RICHARDSON method preconditioned by SOR. For solving the system with the pressure Laplacian matrix, we use one V-cycle of algebraic multigrid method. Let us recall that for constructing the pressure Laplacian matrix, homogeneous Dirichlet boundary condition was used. The set of parameters is as follows

```
MyPCD = {
  "ksp_type":"fgmres",
  "ksp_rtol": 1e-3,
  "ksp_gmres_restart":600,
  "ksp_gmres_modifiedgramschmidt":True,
  "mat_type":"matfree",
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"gmres",
    "ksp_gmres_restart": 100,
    "ksp_max_it":10,
    "ksp_rtol":1e-1,
    "pc_type":"python",
    "pc_python_type":"firedrake.AssembledPC",
    "assembled_pc_type":"bjacobi",
    },

  "fieldsplit_1":{
    "ksp_type":"gmres",
    "ksp_rtol": 1e-1,
    "ksp_max_it": 10,
    "pc_type":"python",
    "pc_python_type":"FPCD.PCDPC",

    "pcd_Mp_ksp_type":"richardson",
    "pcd_Mp_ksp_max_it":5,
    "pcd_Mp_pc_type":"sor",

    "pcd_Kp_ksp_type":"preonly",
    "pcd_Kp_pc_type":"hypre",
    "pcd_Kp_pc_hypre_type":"boomeramg",
    "pcd_Kp_pc_hypre_boomeramg_strong_threshold": 0.2,
    "pcd_Kp_pc_hypre_boomeramg_P_max":2,

    "pcd_Fp_mat_type":"matfree"
    }
  }
```

Figure 3.44: Flow around the cylinder problem in 3D - unsteady Navier-Stokes: set of parameters for FGMRES solver with PCD preconditioning. Our implementation of PCD preconditioner is used.

[LSC] The second solver is FGMRES solver with LSC preconditioning. The matrix $\tilde{\mathbf{A}}^{(k)}$ is inverted by GMRES method preconditioned by BJACOBI. For solving the subsystems connected with LSC preconditioning, RICHARDSON method preconditioned by one V-cycle of algebraic multigrid method is used.

```
LSC = {
  "ksp_type":"fgmres",
  "ksp_gmres_restart":150,
  "ksp_gmres_modifiedgramschmidt":True,
  "pc_type": "fieldsplit",
  "pc_fieldsplit_type":"schur",
  "pc_fieldsplit_schur_fact_type":"upper",

  "fieldsplit_0":{
    "ksp_type":"gmres",
    "ksp_gmres_restart":20,
    "pc_type":"bjacobi",
    },

  "fieldsplit_1":{
    "ksp_type":"preonly",
    "pc_type":"lsc",
    "lsc_pc_type":"ksp",
    "lsc_pc_ksp_type":"richardson",
    "lsc_pc_ksp_max_it":2,
    "lsc_pc_pc_type":"hypre",
    "lsc_pc_pc_hypre_type":"boomeramg",
    "lsc_pc_pc_hypre_boomeramg_strong_threshold": 0.2,
    "lsc_pc_pc_hypre_boomeramg_P_max":2,
    }
  }
```

Figure 3.45: Flow around the cylinder problem in 3D - unsteady Navier-Stokes: set of parameters for FGMRES solver with LSC preconditioning. PETSc's implementation of LSC preconditioner is used.

The following figures (3.46), (3.47) present the velocity and the pressure distribution for unsteady flow around the cylinder in 3D.

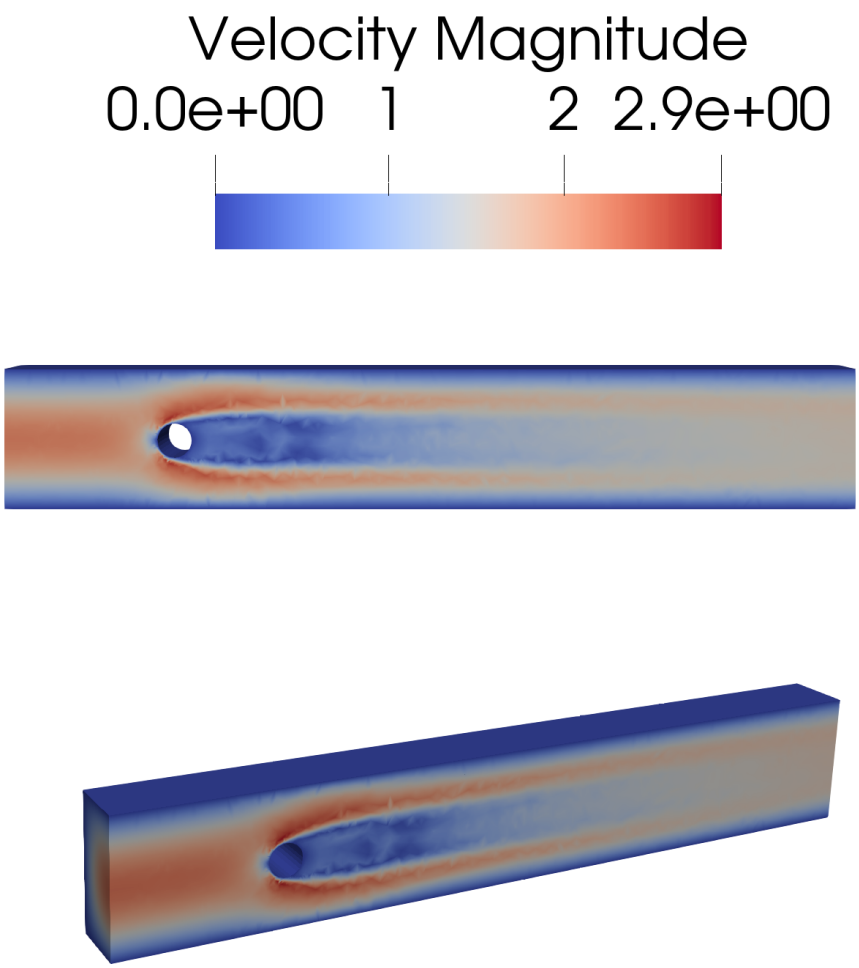


Figure 3.46: Flow around the cylinder problem in 3D - unsteady Navier-Stokes: Velocity of the fluid in lid-driven cavity in 3D at time $t = 5$.

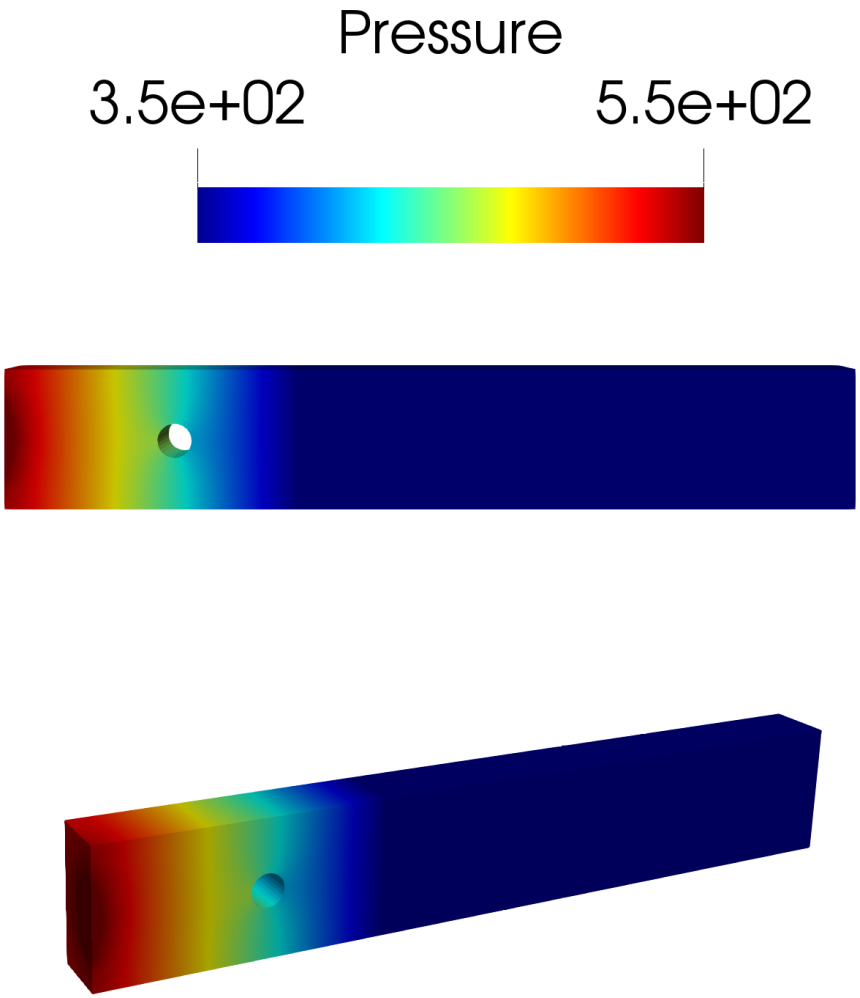


Figure 3.47: Flow around the cylinder problem in 3D - unsteady Navier-Stokes: Pressure of the fluid in lid-driven cavity in 3D at time $t = 5$.

Scalability

Let us now focus on the strong scalability of both iterative solvers. As you can see in the figure (3.48) or in the table (3.11), both solvers are reasonably strongly scalable. Notice that the FGMRES solver with LSC preconditioning is faster as FGMRES solver with PCD preconditioning. On the other hand, FGMRES solver with PCD preconditioning is more effective as you can see in the figure (3.32).

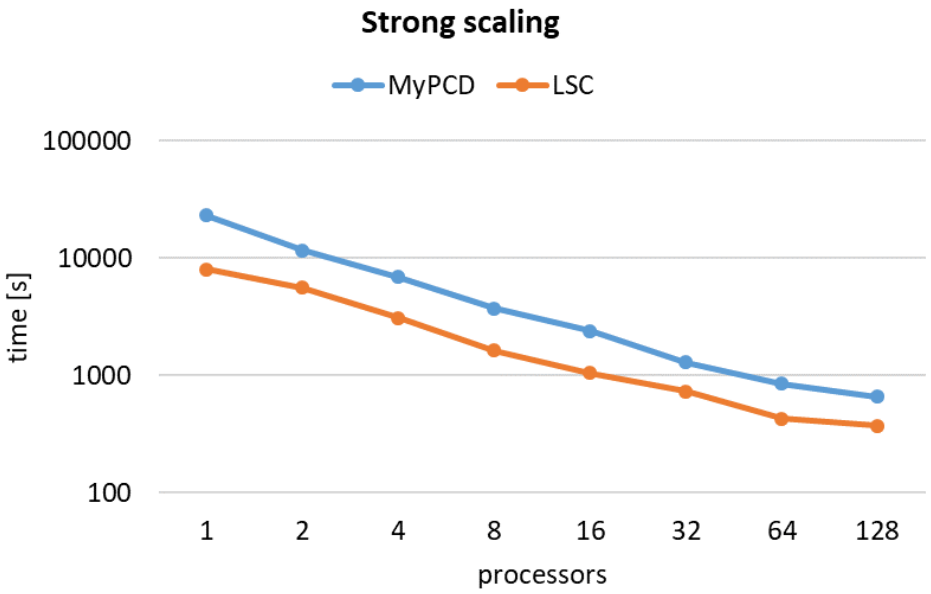


Figure 3.48: Flow around the cylinder problem in 3D - unsteady Navier-Stokes: Strong scaling for 121 889 unknowns on 24 019 cells, logarithmic scale.

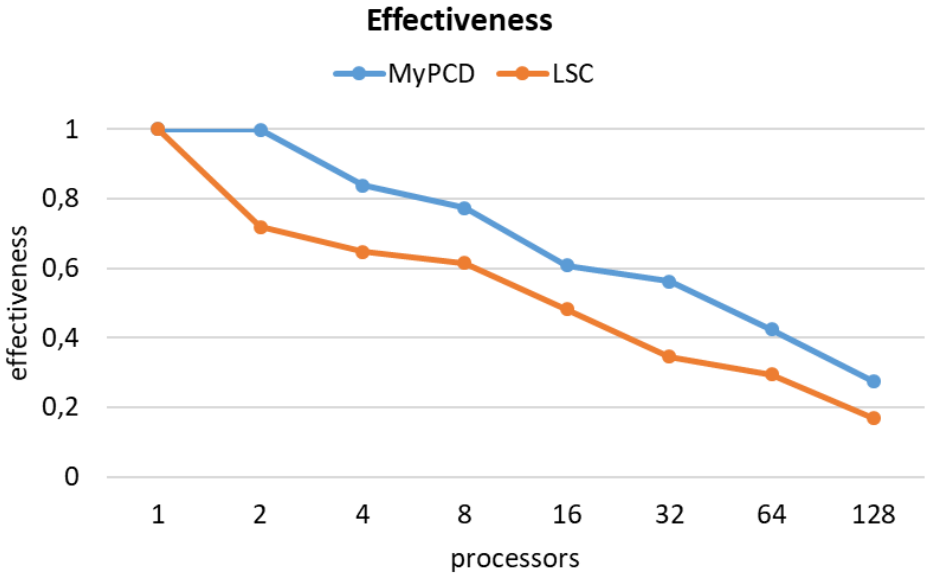


Figure 3.49: Flow around the cylinder problem in 3D - unsteady Navier-Stokes: Graph of effectiveness.

processors	parameter	MyPCD	LSC	MUMPS
1	time [s]	23 210	8 075	
2	time [s]	11 620	5 608	
4	time [s]	6 925	3 113	
8	time [s]	3 750	1 641	
16	time [s]	2 387	1 048	3 511
32	time [s]	1 290	730.4	
64	time [s]	855.2	428.6	
128	time [s]	662.9	373.9	
	nonlinear it	501	501	
	linear it	12 980	13 032	

Table 3.11: Flow around the cylinder problem in 3D - unsteady Navier-Stokes: Table of strong scaling data. The time for MUMPS on 16 processors is computed for comparing.

Conclusion

The motivation of this thesis was the need to solve large flow problems in 3D geometry, where any acceleration of computation is welcome. Since the core computational problem for incompressible flow solver is solution of a saddle-point system, we focused on 2 promising methods based on preconditioned Krylov subspace iterative methods.

In the first chapter, the mathematical foundations for saddle-point systems were presented. Two types of effective preconditioners for saddle-point systems were discussed: block diagonal and block triangular preconditioner. For solving the preconditioned saddle-point systems, the selected Krylov subspace methods were presented.

The cornerstone of the second chapter was the fundamental governing equations for the flow of incompressible, homogeneous, Newtonian fluids. Using specific mathematical procedures (FEM), we derived the saddle-point systems associated with the Stokes and Navier-Stokes problems. For the discretization in space, inf-sup stable P^2/P^1 Taylor-Hood elements were used. In the case of unsteady flow, the discretization in time was realized by Crank-Nicholson scheme. The main emphasis was on approximation of the Schur complement matrix. Two strategies was derived: pressure-convection-diffusion (PCD) preconditioning and least-square commutator (LSC) preconditioning.

The performance of these solvers was studied in the third chapter. For validation purposes, two benchmarks were selected: lid-driven cavity and flow around a cylinder. All numerical simulations were performed using **Firedrake/PETSc** library. We distinguished the steady and unsteady cases in two- and three- dimensional space. The main focus was on comparing the performance of our implementation of PCD preconditioning. Firedrake itself offers its own version of PCD preconditioning but this implementation is limited to steady case with Dirichlet boundary conditions only.

Iterative solvers using our implementation of PCD preconditioning worked well for selected benchmarks. In 2D case, the MUMPS was the most effective for both benchmarks. In 3D case, when steady flow was considered, the performance of iterative solvers using our implementation of PCD preconditioning was more or less the same as the performance of the remaining used iterative solvers. For unsteady flow in 3D case, iterative solvers using LSC preconditioning were the most effective, whereas the use of MUMPS solver becomes less efficient and very limited by its high memory requirements. The strong and weak scaling of used solvers was discussed in the third chapter. In the case of weak scaling, we did not achieve the ideal scalability. Improving the scalability is quite difficult task that could be performed by further careful tuning of the inner solvers used inside the respective preconditioners.

The aim of this thesis was to expand the PCD preconditioning in the Firedrake into more general tasks. This aim was achieved and comparison of performance with other solvers was presented. Our implementation of PCD preconditioning works well but there is a space for improvement - for example, a better way of application of boundary conditions.

Bibliography

P. R. Amestoy, I. S. Duff, J. Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

P. R. Amestoy, A. Guermouche, J. Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

P. R. Amestoy, A. Buttari, J. Y. L'Excellent, and T. Mary. Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures. *ACM Transactions on Mathematical Software*, 45:2:1–2:26, 2019.

I. Babuška. The finite element method with lagrangian multipliers. *Numerische Mathematik*, 20(3):179–192, June 1973. doi: 10.1007/bf01436561. URL `https://doi.org/10.1007/bf01436561`.

S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.11, Argonne National Laboratory, 2019.

M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, April 2005. doi: 10.1017/s0962492904000212. pg. 5-31, 40-96.

D. Bertaccini. *Iterative methods and preconditioning for large and sparse linear systems with applications*. CRC Press, Boca Raton, Florida, 2018. ISBN 978-1-4987-6416-2. pg. 231-274.

J. Blechta. *Towards efficient numerical computation offlows of non-Newtonian fluids*. Doctoral thesis, Charles University, Prague, Czech Republic, 2019. URL `https://dspace.cuni.cz/handle/20.500.11956/108384`.

O. Botella and R. Peyret. Benchmark spectral results on the lid-driven cavity flow. *Computers & Fluids*, 27(4):421–433, May 1998. doi: 10.1016/s0045-7930(98)00002-4. URL `https://doi.org/10.1016/s0045-7930(98)00002-4`.

P. Ciarlet. *The finite element method for elliptic problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002. ISBN 0-89871-514-8. pg. 36-55, 78-103.

L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124–1139, 2011. doi: http:

//dx.doi.org/10.1016/j.advwatres.2011.04.013. New Computational Methods and Software Tools.

H. Elman. *Finite elements and fast iterative solvers : with applications in incompressible fluid dynamics*. Oxford University Press, Oxford, 2014. ISBN 978-0-19-967880-8.

U. Ghia, K. N. Ghia, and C. T. Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, December 1982. doi: 10.1016/0021-9991(82)90058-4. URL https://doi.org/10.1016/0021-9991(82)90058-4.

G. Guj and F. Stella. A vorticity-velocity method for the numerical solution of 3d incompressible flows. *Journal of Computational Physics*, 106(2):286–298, June 1993. doi: 10.1016/s0021-9991(83)71108-3. URL https://doi.org/10.1016/s0021-9991(83)71108-3.

M. Gurtin. *The mechanics and thermodynamics of continua*. Cambridge University Press, New York, 2010. ISBN 978-0-521-40598-0. pg. 127-145, 259-270.

G. Karypis and V. Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *Journal of Parallel and Distributed Computing*, 48:71–85, 1998.

J. Liesen and Z. Strakoš. *Krylov subspace methods : principles and analysis*. Oxford University Press, Oxford, 2016. ISBN 978-0-19-965541-0. pg. 12-33.

A. Logg, K. A. Mardal, and G. Wells, editors. *Automated Solution of Differential Equations by the Finite Element Method*. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-23099-8. URL https://doi.org/10.1007/978-3-642-23099-8.

E. Mitro. CFD-Firedrake. https://github.com/ErikM23/CFD-Firedrake, 2020.

M. Olshanskii and E. Tyrtyshnikov. *Iterative methods for linear systems : theory and applications*. Society for Industrial and Applied Mathematics, Philadelphia, 2014. ISBN 978-1-611-973-45-7. pg. 1-66.

F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. McRae, G. T. Bercea, G. R. Markall, and P. H. J. Kelly. Firedrake: automating the finite element method by composing abstractions. *ACM Trans. Math. Softw.*, 43(3):24:1–24:27, 2016. ISSN 0098-3500. doi: 10.1145/2998441. URL http://arxiv.org/abs/1501.01809.

M. Rozložník. *Saddle-point problems and their iterative solution*. Birkhäuser, Cham, Switzerland, 2018. ISBN 978-3-030-01430-8. pg. 1-7, 21-31, 41-74.

Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2003. ISBN 978-0-89871-534-7. pg. 105-368.

M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher. Benchmark computations of laminar flow around a cylinder, 1996. URL https://doi.org/10.1007/978-3-322-89849-4_39.

M. Řehoř. *Diffuse interface models in theory of interacting continua*. PhD dissertation, Charles University, Prague, Czech Republic, 2018. URL https://dspace.cuni.cz/handle/20.500.11956/103641.

List of Figures

List of Tables