Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Předpodmínění lineárních soustav získaných diskretizací Navierových–Stokesových rovnic pomocí isogeometrické analýzy

Ing. Hana Horníková

disertační práce
k získání akademického titulu doktor (Ph.D.)
v oboru Aplikovaná matematika

Školitel: doc. Ing. Marek Brandner, Ph.D.
Konzultant specialista: doc. Ing. Bohumír Bastl, Ph.D.
Katedra matematiky

Plzeň 2022

University of West Bohemia
Faculty of Applied Sciences

Preconditioning for linear systems arising from discretization of the Navier–Stokes equations using isogeometric analysis

Ing. Hana Horníková

doctoral thesis
submitted for the degree Doctor of Philosophy (Ph.D.)
in the field of Applied Mathematics

Supervisor: doc. Ing. Marek Brandner, Ph.D.
Supervisor specialist: doc. Ing. Bohumír Bastl, Ph.D.
Department of Mathematics

Pilsen 2022

Declaration

I hereby declare that, to the best of my knowledge, this doctoral thesis is an original report of my own research and that any ideas, techniques or other materials from the work of others are fully acknowledged with the standard referencing practices.

In Pilsen, ...
Hana Horníková

Abstrakt

Tato práce se zabývá iteračním řešením sedlobodových soustav lineárních algebraických rovnic získaných diskretizací Navierových–Stokesových rovnic pro nestlačitelné proudění pomocí isogemetrické analýzy (IgA). Konkrétně se zaměřuje na předpodmiňovače pro krylovovské metody. Jedním z cílů práce je prozkoumat efektivitu moderních blokových předpodmiňovačů pro různé isogeometrické diskretizace, tj. pro B-spline bázové funkce různého stupně a spojitosti, a poskytnout přehled o jejich chování v závislosti na různých parametrech úlohy. Hlavním cílem je na základě této studie navrhnout vhodný přístup k řešení těchto soustav s případnými úpravami, které by zlepšily vlastnosti dané metody pro soustavy získané isogeometrickou analýzou.

Práce má dvě části. V první části jsou představeny úlohy pro nestlačitelné vazké proudění a metoda diskretizace pomocí isogeometrické analýzy. Dále uvádíme podrobný přehled metod řešení sedlobodových soustav lineárních rovnic, ve kterém se zaměřujeme především na blokové předpodmiňovače.

Druhá část je věnována numerickým experimentům. Provádíme srovnání vybraných předpodmiňovačů pro několik stacionárních a nestacionarních úloh ve dvou a třech dimenzích. Zvláštní pozornost je věnována aproximaci matice hmotnosti, jejíž volba se ukazuje být v kontextu IgA důležitá, a okrajovým podmínkám pro PCD předpodmiňovač. Navrhujeme vhodnou kombinaci varianty PCD, okrajových podmínek a jejich škálování, abychom získali efektivní předpodmiňovač, který je robustní vzhledem k stupni a spojitosti diskretizace. V mnoha případech se tato volba ukazuje jako nejefektivnější z uvažovaných metod.

Klíčová slova: Navierovy–Stokesovy rovnice, nestlačitelné vazké proudění, isogeometrická analýza (IgA), sedlobodová soustava lineárních algebraických rovnic, krylovovské metody, předpodmínění, blokové předpodmiňovače, PCD předpodmiňovač, LSC předpodmiňovač, předpodmiňovače typu SIMPLE, AL předpodmiňovač

Abstract

This doctoral thesis deals with iterative solution of the saddle-point linear systems obtained from discretization of the incompressible Navier–Stokes equations using the isogeometric analysis (IgA) approach. Specifically, it is focused on preconditioners for Krylov subspace methods. One of the goals of the thesis is to investigate the performance of the state-of-the-art block preconditioners for various IgA discretizations, i.e., for B-spline discretization bases of varying polynomial degree and interelement continuity, and provide an overview of their behavior depending on different problem parameters. The main goal is, based on the this study, to propose suitable solution approach to the considered linear systems with possible modifications that would improve the performance for IgA discretizations in particular.

The thesis is basically divided into two parts. In the first part we introduce the mathematical model of incompressible viscous flow and the isogeometric analysis discretization method. Then we provide a detailed overview of the solution techniques for saddle-point linear systems, especially aimed at the family of block preconditioners.

The second part is devoted to numerical experiments. We present a comparison of the selected preconditioners for several steady-state and time-dependent test problems in two and three dimensions. A particular attention is devoted to mass matrix approximation within the preconditioners, which appears to be important in the context of IgA, and to the boundary conditions for the pressure convection–diffusion (PCD) preconditioner. A suitable combination of PCD variant, boundary conditions and their appropriate scaling is proposed, leading to an effective preconditioner which is robust with respect to the discretization degree and continuity. In many cases, this choice of preconditioner proves to be the most efficient among all considered methods.

Keywords: Navier–Stokes equations, incompressible viscous flow, isogeometric analysis (IgA), saddle-point linear system, Krylov subspace method, preconditioning, block preconditioner, pressure convection–diffusion (PCD) preconditioner, least-squares commutator (LSC) preconditioner, SIMPLE-type preconditioner, augmented Lagrangian (AL) preconditioner

Acknowledgements

I would like to express my gratitude to my supervisor doc. Ing. Marek Brandner, Ph.D., for his support and guidance during my study, his patience and motivation.

I would also like to thank doc. Ing. Bohumír Bastl, Ph.D., for helping me develop my background regarding isogeometric analysis and prof. Kees Vuik from the Delft University of Technology for his time and valuable suggestions.

Last but not least, many thanks also belong to my family, friends and colleagues for their support and understanding.

I acknowledge the financial support of European Union's Horizon 2020 research and innovation programme under grant No. 678727, Czech Science Foundation GA ČR (Grantová agentura České Republiky) grant No. 19-04006S and Technology Agency of the Czech Republic TA ČR (Technologická agentura České Republiky) grant No. TK04020250.

Contents

Chapter 1

Introduction

The incompressible Navier–Stokes equations are a set of partial differential equations used to model the motion of an incompressible viscous Newtonian fluid in d-dimensional space. They consist of d nonlinear momentum equations representing the momentum conservation law and the incompressibility condition representing the mass conservation law. The solution of the equations is a d-dimensional flow velocity field and a scalar pressure field.

Numerical simulation of fluid flow is an indispensable part of many research and application areas, such as meteorology, biological engineering, aerodynamics, mechanical engineering and many others. The Navier–Stokes equations form a basis of many computational fluid dynamics (CFD) problems. With the continuous development of computer technologies and increasing requirements on the size and complexity of simulations, numerical methods for solution of the Navier–Stokes equations are still an evolving and challenging area of research interest.

The solution techniques are typically based on linearization of the governing equations and their spatial discretization. Classical discretization methods involve finite difference, finite volume or finite element method. The finite difference method (FDM) is very simple to derive and easy to implement, but it is not very useful for real world problems because its use for general complex domains is rather complicated. The finite volume methods (FVM) is a common approach in CFD, because it is well suited for solving problems based on conservation laws, even in complex domains with unstructured meshes. Many CFD softwares commonly used in the industrial practice are based on FVM. The finite element method (FEM) is also applicable for complex problems, but is not that popular in the context of CFD. To obtain a stable finite element solution of the incompressible flow problems, the finite element spaces for velocity and pressure have to fulfill the discrete inf-sup (or LBB) condition or some stabilization has to be employed. But even if an inf-sup stable pair of finite elements is used, spurious oscillations can still occur, especially for convection dominated flows, and stabilization is often necessary. Some of the most widely used stabilization methods are based on the Petrov–Galerkin approach, where the shape functions and test functions are different, e.g., SUPG/PSPG or GLS stabilization [49]. An alternative approach, quite popular in CFD nowadays, is the discontinuous Galerkin FEM (DGFEM), which combines some ideas from FEM and FVM, [19, 18]. The spatial discretization of the steady-state incompressible Navier–Stokes equations using any of the mentioned methods results in nonsymmetric sparse linear systems of saddle-point type. This is true also for the time-dependent equations, if implicit time-stepping method is used.

Our work is motivated by the problem of automatic shape optimization of runner blades in water turbines, which we have dealt with in several research projects. The goal is to improve the turbine efficiency in a wide range of operating conditions by changing the runner blade shape automatically, such that the turbine development costs and time are reduced. The optimization problem is stated as minimization of a given objective function subject to constraints given by the Navier–Stokes equations (or RANS equations in case of turbulent flow).

For several reasons, we have chosen the isogeometric analysis (IgA) approach for the spatial discretization of the Navier–Stokes equations. IgA is a relatively new discretization approach proposed by Hughes et al. in [63], which is based on the Galerkin method and has a lot in common with FEM. The main idea is to combine the computer aided design (CAD) tools, usually used for representing the geometries in industrial practice, with the analysis tools, i.e., to be able to compute the quantities of interest directly for the designed geometry, for example the velocity and pressure in the case of fluid flow. B-spline and NURBS objects are used as standard in CAD, therefore IgA uses the B-spline or NURBS basis as the basis for representing the approximate solutions. The IgA approach is suitable for the purpose of automatic shape optimization, because it allows an exact representation of the geometry regardless of the mesh coarseness and does not require new mesh generation every time the domain shape changes, since the computational mesh is already built in the geometry representation.

Similarly to FEM, the IgA discretization of the linearized Navier–Stokes equations (and also RANS equations) leads to a sequence of sparse saddle-point linear systems. One of the main differences between IgA and FEM is that the FEM solution is always C^0 across the element boundaries, while the higher-degree IgA solution is usually of higher-order interelement continuity. This leads to denser matrices, which makes the linear systems more expensive to solve compared to the linear systems of the same size arising from FEM discretizations [20, 21]. However, Hughes et al. show for elliptic problems, that the IgA solution of degree k has the same order of convergence as the classical FEM solution with basis functions of the same degree, independently of the order of continuity [23]. The order of convergence describes, how the error of the approximate solution changes under mesh refinement. Assuming that something similar holds also for other than elliptic problems, we expect IgA to be more efficient than the classical FEM, since the number of degrees of freedom grows much slower with mesh refinement for high-continuity IgA than for standard FEM. In other words, we expect to get an equally "good" solution for much less degrees of freedom using IgA.

During the automatic shape optimization process, the Navier–Stokes/RANS simulation has to be run many times and the linear systems arising from the IgA discretization have to be solved many times during one simulation. Their solution represents the main bottleneck of the whole process, therefore, an efficient linear solver is a key component.

A lot of attention has been devoted to the solution of the large saddle-point linear systems in recent decades. Researchers from various fields are interested in the solution methods for this problem, since it does not arise only in the context of the incompressible fluid flow, but also in many other applications. Some of them are, e.g., linear elasticity, constrained optimization, image reconstruction or mixed FEM approximations of elliptic PDEs. A survey on solving the saddle-point systems has been written by Benzi et al. [5].

There are generally two approaches to the solution of saddle-point problems – coupled and segregated. Coupled methods deal with the resulting linear system as a whole, computing both unknown vectors (velocity and pressure in the case of the incompressible

flow) simultaneously. Segregated methods compute the velocity and the pressure field separately. An example of such method is a Schur complement reduction [5] based on transforming the system to a block triangular one using a block LU factorization of the system matrix. First, the pressure is computed as a solution of a reduced system with the Schur complement matrix and the velocity is obtained afterwards. This approach is not very practical for the linear systems obtained as a discretization of the Navier–Stokes equations, because the Schur complement is dense. As another example of segregated approaches, we name the class of pressure-correction (or projection) methods (see, for example, [93]), where the momentum equations are solved first using the pressure from the previous step to obtain an intermediate velocity field. This velocity field does not fulfill the incompressibility condition. In the next step, it is projected onto the space of divergence-free vector fields using a pressure correction which is computed from an equation of Poisson type. For example, the well known SIMPLE algorithm (Semi-Implicit Method for Pressure Linked Equations) [82] is also a pressure-correction scheme. The advantage of the segregated approach is that we deal with smaller linear systems that can be easier to solve. On the other hand, the convergence to a steady state is often slow.

This work deals with the solution of the coupled linear system. The solution methods for systems of linear algebraic equations can be divided into two groups - direct and iterative methods. Direct methods are usually based on some factorization of the system matrix and give the exact solution of the linear system (assuming exact arithmetic). They are robust, but very expensive in terms of computer memory and CPU time. Therefore, direct solution is almost not feasible for large problems, especially in 3D. On the other hand, iterative methods, which compute an approximate solution, are economical with respect to computer memory, but also less robust and can require many iterations to converge.

The simplest class of iterative methods are the stationary methods, which can be expressed by a simple formula that does not change from iteration to iteration. The other major classes of iterative methods are non-stationary methods, including Krylov subspace methods, and multilevel methods such as multigrid.

Classical representatives of the stationary methods are Jacobi, Gauss-Seidel and SOR. These methods are easy to implement, very cheap in terms of memory, but their convergence can be very slow and is not guaranteed for general matrices. There are also stationary methods developed specifically for saddle-point systems, namely the Uzawa and Arrow-Hurwicz method [108, 5]. Nowadays, the stationary methods are used as preconditioners for Krylov subspace methods or smoothers for multigrid methods rather than standalone solvers.

Krylov subspace methods belong to the most commonly used iterative methods in practice. They are a special case of projection methods, that are based on searching an approximate solution of a linear system in a given m-dimensional subspace $\mathcal{S}_m$ of $\mathbb{R}^n$, $m < n$, called search space, such that the new residual is orthogonal to another m-dimensional subspace $\mathcal{C}_m$ of $\mathbb{R}^n$ called constrained space. We refer, e.g., to [87, 71, 104] for more details on projection methods. Krylov subspace methods form a sequence of nested search spaces $\mathcal{K}_m$ spanned by the initial residual and vectors obtained by repeated multiplication of the initial residual and the system matrix. The choice of the constraint space yields different versions of Krylov subspace methods. A typical example is the conjugate gradient (CG) method, which can be used for solving linear systems with a symmetric positive definite matrix. For general nonsymmetric indefinite problems, the

generalized minimal residual method (GMRES) or the biconjugate gradient stabilized method (BiCGSTAB) belong to the best known methods. GMRES is a stable method which can not break down [88, 104], but its disadvantage is that the work per iteration and memory requirements increase with the iteration number. Therefore, a restarted variant GMRES(k) is often used, which is restarted after every k iterations using the last approximate solution as a new initial guess.

The key ingredient to efficient iterative solution of linear systems is suitable preconditioning. In this context, preconditioning means a transformation of the original system such that the resulting transformed linear system is easier to solve with the given iterative method. The transformation is done by applying a so-called preconditioner to the system matrix and the right hand side vector. The preconditioner should be easy to construct and cheap to apply. This means that linear systems with the preconditioner matrix should be efficiently solvable.

The preconditioner can be either purely algebraic (e.g. incomplete LU factorization or sparse approximate inverse) or based on knowledge of the problem origin, discretization, matrix structure etc. In principle, the algebraic preconditioners can be applied as a black-box, but they often show poor convergence for saddle-point systems [5]. Moreover, the incomplete LU preconditioner will break down when applied to a saddle-point problem due to zero pivots if no fill-in is allowed. A suitable apriori renumbering of unknowns can be used to avoid zero pivots, decrease the memory and time requirements and also to improve the convergence of ILU-preconditioned solvers [94, 105].

In recent years, a lot of attention has been paid to the so-called block triangular preconditioners in context of solving linear systems associated with the incompressible Navier–Stokes equations. These methods belong to the second class of preconditioners, which exploit the knowledge of the system matrix structure and the physics behind individual blocks. Similarly to the segregation methods mentioned above, they are based on splitting the system into the velocity and pressure part. They can be derived using a block LDU decomposition of the system matrix leading to an upper (or lower) block triangular preconditioner matrix. The application of the preconditioner requires solving two subsystems, one with the pressure Schur complement matrix and one with the (1,1) block of the original saddle point matrix (a discrete convection–diffusion operator). Since it is impractical to construct the Schur complement explicitly, it has to be approximated. The choice of the approximation then yields different preconditioners. In the "ideal" versions of the block preconditioners, the subsystems are solved with a direct solver. However, in practice it is necessary to use approximate solution methods for these subsystems to obtain an efficient preconditioner.

One of the widely used block triangular preconditioners is the pressure convection–diffusion preconditioner (PCD), proposed by Kay et al. [68], where the Schur complement approximation is derived based on fundamental solution operators, and Silvester et al. [96] based on the idea of approximate commutators. A disadvantage of this method is the need to assemble new matrices – a discrete convection–diffusion and a discrete Poisson operator on the pressure space – that are not readily available in the standard FEM or FVM codes. To overcome this, Elman et al. in [35] developed the least-squares commutator preconditioner (LSC), where the pressure convection–diffusion operator is computed from a weighted least-squares problem and the discrete Poisson operator is not explicitly needed. It corresponds to a different interpretation of a scaled variant of the BFBt preconditioner proposed earlier by Elman [33].

Another approach was proposed by Benzi and Olshanskii in [6]. They start with the

augmented Lagrangian (AL) formulation of the saddle-point problem and construct a block triangular preconditioner, which is applied to the augmented system instead of the original one. This preconditioner has an attractive property of convergence independent of mesh refinement and Reynolds number, but its efficiency strongly depends on the availability of a good approximate solver for the (1,1) block of the augmented matrix.

As already mentioned above, the SIMPLE algorithm is a popular solver for flow problems, which often requires many iterations to converge to a steady state. In [112, 113], Vuik et al. proposed a Krylov accelerated version of SIMPLE-type methods, treating these algorithms as block preconditioners for Krylov subspace methods. They show that it can result in much faster convergence to the steady state than if SIMPLE is used as a solver, see also [69].

The mentioned block preconditioners have been developed and successfully used for finite element discretizations of the incompressible Navier–Stokes equations. The goal of this thesis is to study and implement the selected state-of-the-art preconditioning techniques and investigate their behavior for linear systems arising from the IgA discretization of the incompressible Navier–Stokes equations. The question we try to answer is whether they are applicable to the IgA discretizations of high degree and high continuity with the same success. A similar study was done for IgA discretizations of the Stokes equations by Côrtes et al. [22], where the authors combine a block triangular strategy with several "black-box" solvers to get a scalable preconditioner. We consider the steady-state and time-dependent Navier–Stokes equations linearized by Picard method and present a comparison of the block preconditioners for several test problems in two and three dimensions and a variety of IgA discretizations. Based on our results, we give recommendations for the choice of preconditioner.

The text of this thesis is organized as follows. In Chapter 2, we formulate the incompressible Navier–Stokes equations and describe the steps leading to the saddle-point linear systems. This includes the weak formulation of the equations, their linearization and Galerkin discretization. Chapter 3 is devoted to the isogeometric analysis. We review the basic definitions and properties of B-spline and NURBS, describe important aspects of the use of B-splines as a Galerkin discretization basis in general and also for the Navier–Stokes equations specifically. The first part of Chapter 4 provides a brief overview of solution methods for general linear systems with the main focus on the Krylov subspace methods. The second part then deals with methods designed for saddle-point problems. Chapter 5 begins with a description of the concept of preconditioning and the preconditioned GMRES algorithm and its main part is devoted to the preconditioners for linear systems arising from discretization of the Navier–Stokes equations. Results of our numerical experiments are presented in Chapter 6. Finally, we conclude the thesis with a summary of observations and our recommendations and outline possible topics for future work.

Chapter 2

The incompressible Navier–Stokes equations

The Navier–Stokes equations represent the fundamental mathematical model describing the motion of viscous fluids. In this chapter, we formulate the model equations for an incompressible Newtonian fluid. Further, we outline the steps needed for the numerical solution of the steady-state and time-dependent Navier–Stokes equations using Galerkin discretization method, leading to saddle-point linear systems.

2.1 Problem formulation

Consider a domain $\Omega \subset \mathbb{R}^d$ filled with a fluid, where $d \in \{2, 3\}$ is the space dimension, and a time interval $[0, T]$. The flow of the fluid is described by the following physical quantities: density $\rho(\boldsymbol{x}, t)$, velocity $\boldsymbol{u}(\boldsymbol{x}, t)$ and pressure $P(\boldsymbol{x}, t)$, for $\boldsymbol{x} \in \Omega, t \in [0, T]$. In the following, we assume that all quantities are sufficiently smooth functions.

The Navier–Stokes equations are derived based on two physical conservation laws, the conservation of mass and the conservation of momentum. The conservation of mass means that the rate of change of mass in an arbitrary volume $\omega \subset \Omega$ is equal to the flux of mass across its boundary $\partial\Omega$. The local form of the mass conservation law takes the form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0, \quad \forall \boldsymbol{x} \in \Omega, t \in (0, T], \tag{2.1}$$

which is called the continuity equation. If the fluid is incompressible and homogeneous, the density ρ is constant in space and time, i.e., $\rho(\boldsymbol{x}, t) \equiv \rho_0 > 0$. Thus, the continuity equation (2.1) simplifies to

$$\nabla \cdot \boldsymbol{u} = 0, \quad \forall \boldsymbol{x} \in \Omega, t \in (0, T]. \tag{2.2}$$

The conservation of momentum states that the momentum in ω is neither created nor destroyed, but only changed through the action of forces as described by Newton's laws of motion. For a homogeneous, incompressible fluid, using the fact that the density ρ is constant as well as the continuity equation (2.2), the local form of the law of conservation of momentum takes the form

$$\rho_0 \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = \nabla \cdot \mathbb{S} + \boldsymbol{f}_{\text{ext}}, \quad \forall \boldsymbol{x} \in \Omega, t \in (0, T], \tag{2.3}$$

where $\mathbb{S} = \mathbb{S}(\boldsymbol{x}, t)$ is the Cauchy stress tensor and $\boldsymbol{f}_{\text{ext}} = \boldsymbol{f}_{\text{ext}}(\boldsymbol{x}, t)$ represents the external forces acting on the fluid. The external forces can include, e.g., gravity, buoyancy or electromagnetic forces. The relation (2.3) is called the momentum equation.

The Cauchy stress tensor $\mathbb{S}$ is a symmetric $(d \times d)$-tensor of order 2 representing all internal forces in the fluid. Its divergence is defined row-wise, i.e.,

$$\nabla \cdot \mathbb{S} = \frac{\partial \mathbb{S}_{ik}}{\partial x_k} \vec{e}_i, \tag{2.4}$$

where $\vec{e}_i$ is the unit vector in the direction of i-th Cartesian coordinate axis and Einstein summation convention is used. For Newtonian fluids, $\mathbb{S}$ depends linearly on the velocity deformation tensor

$$\mathbb{D}(\boldsymbol{u}) = \frac{\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T}{2}, \tag{2.5}$$

i.e., the symmetric part of the velocity gradient. The Cauchy stress tensor can be expressed as

$$\mathbb{S} = 2\mu \mathbb{D}(\boldsymbol{u}) - P\mathbb{I}, \tag{2.6}$$

where μ is the dynamic viscosity of the fluid and $\mathbb{I}$ is the identity tensor.

After substituting the relation (2.6) into the momentum equation (2.3) and dividing both sides of the equation by the constant density ρ_0, we get the momentum equation for an incompressible Newtonian fluid in the form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - 2\nu \nabla \cdot \mathbb{D}(\boldsymbol{u}) + \nabla p = \boldsymbol{f}, \tag{2.7}$$

where $\nu = \mu/\rho_0$ is the kinematic viscosity of the fluid, $p = P/\rho_0$ is the kinematic pressure and $\boldsymbol{f} = \boldsymbol{f}_{\text{ext}}/\rho_0$. This relation can be further simplified thanks to the incompressibility constraint (2.2), since

$$\begin{aligned} \nabla \cdot \mathbb{D}(\boldsymbol{u}) &= \frac{1}{2}\left(\nabla \cdot (\nabla \boldsymbol{u}) + \nabla \cdot (\nabla \boldsymbol{u})^T\right) \\ &= \frac{1}{2}(\Delta \boldsymbol{u} + \nabla(\nabla \cdot \boldsymbol{u})) = \frac{1}{2}\Delta \boldsymbol{u}. \end{aligned} \tag{2.8}$$

Note that the Laplacian of a vector function is a vector of Laplacians of the corresponding components.

By substituting (2.8) into the momentum equation (2.7) and combining it with the continuity equation (2.2), we obtain the formulation of the Navier–Stokes equations for an incompressible Newtonian fluid

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \qquad \text{in } \Omega \times (0, T], \\ \nabla \cdot \boldsymbol{u} &= 0 \qquad \text{in } \Omega \times (0, T]. \end{aligned} \tag{2.9}$$

For more details on the derivation of the Navier–Stokes equations see, e.g., [66, 45].

The incompressible Navier–Stokes equations bring several difficulties for the numerical simulation as well as mathematical analysis. One of them is the fact that the continuity equation does not involve the pressure variable. This kind of coupling is called saddle-point problem. As a result, if we solve the problem using Galerkin-based method such as finite elements, the solution spaces for velocity and pressure cannot be chosen arbitrarily. We will comment on that issue in more detail later. Another difficulty

is the nonlinearity of the convective term $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$. In order to solve the Navier–Stokes equations numerically, we have to use some linearization method first. Two well known linearization methods, Picard and Newton, will be described in the next section.

Moreover, the numerical solution becomes challenging if the convective effects dominate the viscous effects. The relative contributions of convection and diffusion are defined by a dimensionless quantity called the Reynolds number

$$Re = \frac{UL}{\nu}, \tag{2.10}$$

where L is a characteristic length scale of the domain Ω and U is a reference velocity. The characteristic length scale L is chosen by convention for specific types of domain geometries. For example, it is the pipe diameter for pipe flow, the chord length for flow around an airfoil etc. The characteristic velocity U is usually chosen as the maximum velocity of the fluid relative to the walls or an object moving in the fluid. The Reynolds number is used to characterize the flow regime (laminar or turbulent) and predict the flow patterns. For very low Reynolds numbers, the flow is diffusion-dominated and the fluid tends to move in non-mixing layers (laminar flow). On the other hand, for high Reynolds numbers, the flow is convection-dominated and turbulent flow occurs, which is characterized by chaotic behavior and random fluctuations. For Reynolds number values in some range starting from a so-called critical Reynolds number, there is a transition phase where the flow loses stability and becomes turbulent. The value of critical Re varies for different fluids and domains.

Initial and boundary conditions

The time-dependent incompressible Navier–Stokes system (2.9) has to be formulated with an initial condition at $t = 0$ and boundary conditions on the domain boundary $\partial\Omega$. An initial velocity field $\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x})$ is prescribed as the initial condition at $t = 0$. It has to be divergence-free and fulfill the given boundary conditions for $t \to 0^+$.

There are several types of boundary conditions that can be specified for an incompressible flow problem. Here we do not give a full list of options, we only mention the boundary conditions considered in this work. For more detailed overview see, e.g., [66].

One of the basic types of boundary conditions is a Dirichlet condition, which means that we prescribe the velocity at a part of the boundary

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{g}(\boldsymbol{x}, t) \quad \text{in } \partial\Omega_D \times (0, T], \tag{2.11}$$

where $\partial\Omega_D \subset \partial\Omega$ and $\boldsymbol{g}$ is a given function. At fixed walls, we usually set $\boldsymbol{g} \equiv \boldsymbol{0}$, which is called no-slip boundary condition. Generally, the Dirichlet boundary can be divided into three parts

- the inflow part ($\boldsymbol{g} \cdot \boldsymbol{n} < 0$),
- the outflow part ($\boldsymbol{g} \cdot \boldsymbol{n} > 0$),
- the characteristic part ($\boldsymbol{g} \cdot \boldsymbol{n} = 0$),

where $\boldsymbol{n}$ denotes the outward-pointing unit normal to the boundary.

If the Dirichlet condition is specified on the whole boundary $\partial\Omega$, the pressure is determined only up to an additive constant, sometimes referred to as the hydrostatic

pressure. To obtain a unique pressure solution, we have to fix that constant by introducing an additional condition for pressure, e.g., that its integral mean value over Ω is equal to zero. Moreover, the prescribed velocity has to satisfy the compatibility condition obtained by integrating the incompressibility constraint (2.2) over Ω and using the divergence theorem

$$0 = \int_{\Omega} \nabla \cdot \boldsymbol{u} = \int_{\partial\Omega} \boldsymbol{u} \cdot \boldsymbol{n} = \int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{n} \quad \forall t \in (0, T]. \tag{2.12}$$

A special case is the enclosed flow for which $\boldsymbol{g} \cdot \boldsymbol{n} = 0$ everywhere on $\partial\Omega$.

Another common type of boundary condition is the so-called do-nothing boundary condition

$$(2\nu\mathbb{D}(\boldsymbol{u}) - p\mathbb{I})\,\boldsymbol{n} = \boldsymbol{0} \text{ in } \partial\Omega_N \times (0, T], \tag{2.13}$$

where $\partial\Omega_N \subset \partial\Omega$ and $\mathbb{I}$ is the identity tensor. This condition models zero normal stress on the boundary part $\partial\Omega_N$ and it is often used at free outflow boundaries. It arises as a natural boundary condition for the Navier–Stokes equations, when the momentum equation formulation (2.7) is used. The reason why it is called "do-nothing" is that the term $(2\nu\mathbb{D}(\boldsymbol{u}) - p\mathbb{I})\,\boldsymbol{n}$ appears in the boundary integral in the weak formulation and due to the condition (2.13), this boundary integral vanishes at $\partial\Omega_N$. Thus, if no further boundary integrals are added to the weak formulation, the do-nothing condition is satisfied automatically.

However, this outflow condition does not allow the Poisseuille flow, which is an analytical solution to a simple two-dimensional channel problem with a parabolic inflow velocity. An extension of the inflow velocity solves the incompressible Navier–Stokes equations together with a pressure, which cannot satisfy the outflow condition (2.13) (see e.g. [66]). If the same problem is solved such that (2.13) is satisfied, the velocity vectors at the outlet are directed to the boundaries of the channel. It corresponds to a situation, when the channel ends in an open space.

The modified do-nothing condition

$$(\nu\nabla\boldsymbol{u} - p\mathbb{I})\,\boldsymbol{n} = \boldsymbol{0} \text{ in } \partial\Omega_N \times (0, T], \tag{2.14}$$

where the symmetric part of the velocity gradient is replaced by the whole gradient, does allow the Poisseuille flow. This condition does not have a physical meaning anymore, but it can be interpreted as an artificial boundary condition such that the computational domain Ω is only a restriction of a larger physical domain, which is assumed to continue further. It arises as a natural boundary condition for the Navier–Stokes equations formulation with the velocity Laplacian (2.9), since the term $(\nu\nabla\boldsymbol{u} - p\mathbb{I})\,\boldsymbol{n}$ appears in the boundary integral in the weak formulation.

Note that if (2.13) or (2.14) is prescribed at some part of the boundary, we do not need to introduce any additional conditions for pressure, since a pressure term is already included in the do-nothing boundary condition.

It is also important to mention that the do-nothing condition is not suitable if there is some inflow at the outlet boundary (e.g., if a vortex crosses the outlet). Therefore, a modification called directional do-nothing condition was introduced, see [11] for details. However, we do not consider this modification in this work.

Steady-state Navier–Stokes equations

If the velocity and pressure do not change in time, we talk about a stationary flow. Hence, the time derivative of the velocity is equal to zero, which leads to the stationary

or steady-state Navier–Stokes equations

$$\begin{aligned} \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0 \qquad \text{in } \Omega, \end{aligned} \tag{2.15}$$

where the function $\boldsymbol{f}$ also does not depend on time. In practice, such flows can be expected if the viscosity ν is sufficiently large (i.e., the Reynolds number Re is sufficiently small).

The steady-state Navier–Stokes equations need to be equipped with boundary conditions on $\partial\Omega$. The same types of boundary conditions can be prescribed as in the case of unsteady Navier–Stokes equations, assuming that all given data are time-independent.

Stokes equations

A stationary flow of a viscous fluid with a very low velocity such that the convection effects can be neglected, is modeled by the so-called Stokes equations

$$\begin{aligned} -\nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0 \qquad \text{in } \Omega. \end{aligned} \tag{2.16}$$

This simplified model is obtained from the stationary Navier–Stokes equations by omitting the nonlinear convective term $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$. Thus, the resulting equations are linear.

Oseen equations

Another simplification of the Navier–Stokes equations are the Oseen equations

$$\begin{aligned} \boldsymbol{w} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0 \qquad \text{in } \Omega, \end{aligned} \tag{2.17}$$

which take convection into account. The convection field $\boldsymbol{w}$ (which is assumed to be divergence-free) is given and thus the Oseen equations are again linear.

The Oseen equations often arise as a part of numerical solution of the Navier–Stokes equations. Specifically, if the Picard linearization method is applied, we obtain a sequence of Oseen problems, where $\boldsymbol{w}$ corresponds to the velocity field obtained in the most recent iteration.

2.2 Galerkin discretization and linearization

Since the isogeometric analysis is based on the Galerkin method, before we go into the description of the IgA discretization itself, we briefly summarize the steps leading to the Galerkin discretization of the steady-state and time-dependent Navier–Stokes equations. In the steady case, these steps include weak formulation of the problem and linearization of the convective term. In the case of time-dependent problem, there are several possible approaches to the discretization. We choose to discretize in time first and proceed similarly to the steady case at each time level.

2.2.1 The steady-state problem

We consider a boundary value problem for the steady-state Navier–Stokes equations in a bounded domain $\Omega \subset \mathbb{R}^d$ with a Lipschitz continuous boundary $\partial\Omega$ consisting of two complementary parts $\partial\Omega_D$ and $\partial\Omega_N$, with no external forces, i.e. $\boldsymbol{f} = \boldsymbol{0}$. The problem is given as

$$\begin{aligned} \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{0} && \text{in } \Omega, && (2.18)\\ \nabla \cdot \boldsymbol{u} &= 0 && \text{in } \Omega, && (2.19)\\ \boldsymbol{u} &= \boldsymbol{g} && \text{on } \partial\Omega_D, && (2.20)\\ \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - \boldsymbol{n} p &= \boldsymbol{0} && \text{on } \partial\Omega_N, && (2.21) \end{aligned}$$

where the Dirichlet boundary condition $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x})$ does not depend on time.

Weak formulation

Let $(\boldsymbol{u}, p)$ be a classical solution of the steady-state Navier–Stokes problem, that is, let $\boldsymbol{u} \in [C^2(\bar{\Omega})]^d$ and $p \in C^1(\bar{\Omega})$ satisfy (2.18) - (2.21). The weak formulation is derived by multiplying the momentum equation (2.18) with a test function $\boldsymbol{v}$ and the continuity equation (2.19) with a test function q and integrating over the domain Ω. We obtain the identities

$$\begin{aligned} \int_\Omega (\boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p) \cdot \boldsymbol{v} = 0,\\ \int_\Omega q \nabla \cdot \boldsymbol{u} = 0, \end{aligned} \tag{2.22}$$

for all $\boldsymbol{v}$ and q from suitably chosen spaces of test functions. To reduce the continuity requirements on the weak solution, integration by parts is applied, leading to

$$\begin{aligned} \int_\Omega (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v} + \nu \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - \int_{\partial\Omega} \left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n} \right) \cdot \boldsymbol{v} - \int_\Omega p \nabla \cdot \boldsymbol{v} = 0,\\ \int_\Omega q \nabla \cdot \boldsymbol{u} = 0. \end{aligned} \tag{2.23}$$

The convective term should be understood as follows

$$(\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v} = \sum_{i=1}^{d} \sum_{j=1}^{d} u_j \frac{\partial u_i}{\partial x_j} v_i, \tag{2.24}$$

and the expression $\nabla \boldsymbol{u} : \nabla \boldsymbol{v}$ represents the componentwise scalar product, i.e.,

$$\nabla \boldsymbol{u} : \nabla \boldsymbol{v} = \sum_{i=1}^{d} \nabla u_i \cdot \nabla v_i = \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j}, \tag{2.25}$$

where u_i, v_i are the components of $\boldsymbol{u}, \boldsymbol{v}$. The appropriate velocity solution space $\boldsymbol{\mathcal{V}}_g$, velocity test space $\boldsymbol{\mathcal{V}}$ and pressure solution and test space $\mathcal{Q}$ are defined as follows

$$\begin{aligned} \boldsymbol{\mathcal{V}}_g &= \{\boldsymbol{u} \in [H^1(\Omega)]^d \,|\, \boldsymbol{u} = \boldsymbol{g} \text{ on } \partial\Omega_D\},\\ \boldsymbol{\mathcal{V}} &= \{\boldsymbol{v} \in [H^1(\Omega)]^d \,|\, \boldsymbol{v} = \boldsymbol{0} \text{ on } \partial\Omega_D\},\\ \mathcal{Q} &= L^2(\Omega), \end{aligned} \tag{2.26}$$

where the boundary values of $\boldsymbol{u}$ and $\boldsymbol{v}$ are understood in the sense of traces, assuming that $\boldsymbol{g} \in H^{1/2}(\partial\Omega_D)$. The boundary integral in (2.23) vanishes on $\partial\Omega_D$, since the test functions $\boldsymbol{v}$ vanish on that part of the boundary, and also on $\partial\Omega_N$ due to the do-nothing boundary condition. Thus, the weak formulation is: find $\boldsymbol{u} \in \boldsymbol{\mathcal{V}}_g$ and $p \in \mathcal{Q}$ such that

$$\begin{aligned} \int_\Omega (\boldsymbol{u}\cdot\nabla\boldsymbol{u})\cdot\boldsymbol{v} + \nu\int_\Omega \nabla\boldsymbol{u} : \nabla\boldsymbol{v} - \int_\Omega p\nabla\cdot\boldsymbol{v} = 0 \qquad & \forall \boldsymbol{v} \in \boldsymbol{\mathcal{V}}, \\ \int_\Omega q\nabla\cdot\boldsymbol{u} = 0 \qquad & \forall q \in \mathcal{Q}. \end{aligned} \tag{2.27}$$

In the case $\partial\Omega_D = \partial\Omega$, the existence of weak solution can be proven (under additional conditions on the Dirichlet data $\boldsymbol{g}$), see, e.g., [45, 50] for the proof. The uniqueness of the solution is guaranteed only for small data (Reynolds number, external force and boundary conditions). Moreover, Galdi in [50] gives examples of problems for the steady-state Navier-Stokes equations with Dirichlet boundary conditions in three dimensions that admit more than one weak solution. For discussion of more general boundary conditions, see [45] and references therein.

Linearization

To be able to solve the Navier–Stokes equations numerically, the nonlinear problem (2.27) needs to be linearized and solved iteratively. Given an initial guess $(\boldsymbol{u}^0, p^0) \in \boldsymbol{\mathcal{V}}_g \times \mathcal{Q}$, we compute a sequence of approximate solutions $(\boldsymbol{u}^k, p^k) \in \boldsymbol{\mathcal{V}}_g \times \mathcal{Q}$ for $k = 1, 2, \ldots$, by solving certain linear problems. Here we describe two widely used linearization methods, Newton and Picard iteration, following the exposition in [40].

Assume that we have computed the iterate $(\boldsymbol{u}^k, p^k)$ and denote R^k, r^k the nonlinear residuals of the weak formulation (2.27),

$$R^k = -\int_\Omega (\boldsymbol{u}^k\cdot\nabla\boldsymbol{u}^k)\cdot\boldsymbol{v} - \nu\int_\Omega \nabla\boldsymbol{u}^k : \nabla\boldsymbol{v} + \int_\Omega p^k\nabla\cdot\boldsymbol{v}, \tag{2.28}$$

$$r^k = -\int_\Omega q\nabla\cdot\boldsymbol{u}^k. \tag{2.29}$$

Further assume that $(\boldsymbol{u}, p)$ is the exact solution. Let us express it as a combination of the solution from the k-th linearization step and unknown corrections $\delta\boldsymbol{u}^k \in \boldsymbol{\mathcal{V}}$, $\delta p^k \in \mathcal{Q}$:

$$\boldsymbol{u} = \boldsymbol{u}^k + \delta\boldsymbol{u}^k, \quad p = p^k + \delta p^k, \tag{2.30}$$

and substitute it into the weak formulation. After rearranging the equations, we obtain the following relations for the corrections

$$\begin{aligned} \int_\Omega (\boldsymbol{u}^k\cdot\nabla\delta\boldsymbol{u}^k)\cdot\boldsymbol{v} + \int_\Omega (\delta\boldsymbol{u}^k\cdot\nabla\boldsymbol{u}^k)\cdot\boldsymbol{v} + \int_\Omega (\delta\boldsymbol{u}^k\cdot\nabla\delta\boldsymbol{u}^k)\cdot\boldsymbol{v} + \\ + \nu\int_\Omega \nabla\delta\boldsymbol{u}^k : \nabla\boldsymbol{v} - \int_\Omega \delta p^k\nabla\cdot\boldsymbol{v} = R^k, \end{aligned} \tag{2.31}$$

$$\int_\Omega q\nabla\cdot\delta\boldsymbol{u}^k = r^k \tag{2.32}$$

for all $\boldsymbol{v} \in \boldsymbol{\mathcal{V}}, q \in \mathcal{Q}$.

The Newton's method is based on dropping the quadratic term $\int_\Omega(\delta\boldsymbol{u}^k\cdot\nabla\delta\boldsymbol{u}^k)\cdot\boldsymbol{v}$ from (2.31) and solving the following linear problem in each iteration: find $\delta\boldsymbol{u}^k\in\boldsymbol{\mathcal{V}}$ and $\delta p^k\in\mathcal{Q}$ such that

$$\begin{aligned}\int_\Omega(\boldsymbol{u}^k\cdot\nabla\delta\boldsymbol{u}^k)\cdot\boldsymbol{v}+\int_\Omega(\delta\boldsymbol{u}^k\cdot\nabla\boldsymbol{u}^k)\cdot\boldsymbol{v}+\nu\int_\Omega\nabla\delta\boldsymbol{u}^k:\nabla\boldsymbol{v}-\int_\Omega\delta p^k\nabla\cdot\boldsymbol{v}&=R^k,\\ \int_\Omega q\nabla\cdot\delta\boldsymbol{u}^k&=r^k,\end{aligned}\tag{2.33}$$

is satisfied for all $\boldsymbol{v}\in\boldsymbol{\mathcal{V}},q\in\mathcal{Q}$. Once the corrections are obtained, the new approximations of the velocity and pressure are then defined by updating the previous iterate via

$$\boldsymbol{u}^{k+1}=\boldsymbol{u}^k+\delta\boldsymbol{u}^k,\quad p^{k+1}=p^k+\delta p^k.\tag{2.34}$$

The algorithm of the Newton's method can be also reformulated such that the new iterate $(\boldsymbol{u}^{k+1},p^{k+1})$ is determined explicitly as a solution of a linear problem. Substituting $\delta\boldsymbol{u}^k=\boldsymbol{u}^{k+1}-\boldsymbol{u}^k$ and $\delta p^k=p^{k+1}-p^k$ into the equations (2.33) leads to the following problem: find $\boldsymbol{u}^{k+1}\in\boldsymbol{\mathcal{V}}_g$ and $p^{k+1}\in\mathcal{Q}$ such that

$$\begin{aligned}\int_\Omega(\boldsymbol{u}^k\cdot\nabla\boldsymbol{u}^{k+1})\cdot\boldsymbol{v}\;+\int_\Omega(\boldsymbol{u}^{k+1}\cdot\nabla\boldsymbol{u}^k)\cdot\boldsymbol{v}+&\\ +\nu\int_\Omega\nabla\boldsymbol{u}^{k+1}:\nabla\boldsymbol{v}-\int_\Omega p^{k+1}\nabla\cdot\boldsymbol{v}=\int_\Omega(\boldsymbol{u}^k\cdot\nabla\boldsymbol{u}^k)\cdot\boldsymbol{v},&\\ \int_\Omega q\nabla\cdot\boldsymbol{u}^{k+1}=0,&\end{aligned}\tag{2.35}$$

is satisfied for all $\boldsymbol{v}\in\boldsymbol{\mathcal{V}},q\in\mathcal{Q}$. Note that the new iterate depends on the velocity approximation from the previous iteration $\boldsymbol{u}^k$, but it is independent of the pressure approximation p^k, and thus, the initial pressure guess p^0 can be arbitrary. If the Newton iteration converges, its convergence is quadratic. However, the main drawback is that the initial velocity $\boldsymbol{u}^0$ has to be sufficiently close to the exact solution for the Newton's method to converge. Moreover, the initial guess has to be better and better for increasing Reynolds number [40].

Similarly to the Newton's method, the Picard's linearization is based on the equations (2.31) - (2.32). Here, not only the quadratic term, but also the term $\int_\Omega(\delta\boldsymbol{u}^k\cdot\nabla\boldsymbol{u}^k)\cdot\boldsymbol{v}$ is omitted. Thus, the linear problem which is solved in each iteration of the Picard's method is: find $\delta\boldsymbol{u}^k\in\boldsymbol{\mathcal{V}}$ and $\delta p^k\in\mathcal{Q}$ satisfying

$$\begin{aligned}\int_\Omega(\boldsymbol{u}^k\cdot\nabla\delta\boldsymbol{u}^k)\cdot\boldsymbol{v}+\nu\int_\Omega\nabla\delta\boldsymbol{u}^k:\nabla\boldsymbol{v}-\int_\Omega\delta p^k\nabla\cdot\boldsymbol{v}&=R^k\qquad\forall\boldsymbol{v}\in\boldsymbol{\mathcal{V}},\\ \int_\Omega q\nabla\cdot\delta\boldsymbol{u}^k&=r^k\qquad\forall q\in\mathcal{Q}.\end{aligned}\tag{2.36}$$

The new approximation of the solution is obtained using (2.34). This approach can be also reformulated such that the new iterate $(\boldsymbol{u}^{k+1},p^{k+1})$ is determined explicitly as a solution of a linear problem: find $\boldsymbol{u}^{k+1}\in\boldsymbol{\mathcal{V}}_g$ and $p^{k+1}\in\mathcal{Q}$ such that

$$\begin{aligned}\int_\Omega(\boldsymbol{u}^k\cdot\nabla\boldsymbol{u}^{k+1})\cdot\boldsymbol{v}+\nu\int_\Omega\nabla\boldsymbol{u}^{k+1}:\nabla\boldsymbol{v}-\int_\Omega p^{k+1}\nabla\cdot\boldsymbol{v}&=0\qquad\forall\boldsymbol{v}\in\boldsymbol{\mathcal{V}},\\ \int_\Omega q\nabla\cdot\boldsymbol{u}^{k+1}&=0\qquad\forall q\in\mathcal{Q}.\end{aligned}\tag{2.37}$$

This corresponds to a simple fixed point iteration for solving (2.27) with the convection velocity taken from the most recent linearization step. Note that the problem (2.37) is a weak formulation of an Oseen problem with $\boldsymbol{w} = \boldsymbol{u}^k$. The convergence of the Picard iteration is generally only linear, but its advantage is that it is convergent for a wide range of initial guesses $\boldsymbol{u}^0$ (the initial pressure p^0 is again arbitrary). Often, the initial velocity $\boldsymbol{u}^0$ is set to zero, which results in solving the Stokes problem in the first Picard iteration.

Another advantage of the Picard's method is that, unlike the Newton's method, the components of the unknown velocity are decoupled. This can be beneficial for some methods for solving the discretized problem, where the velocity and pressure part are solved separately. Thanks to the velocity components decoupling, the resulting velocity matrix is block diagonal and thus the velocity linear system can be split into d smaller systems.

Sometimes, the Picard and Newton iteration are combined. First, several iterations of the Picard's method are used to start the iteration process and get closer to the solution, and then the Newton's method is used to accelerate the convergence. In this work, we limit ourselves to the Picard linearization. Thus, we are interested in solving problems of the form (2.37).

Galerkin discretization

The idea of Galerkin discretization method is to define finite dimensional subspaces of the solution and test spaces and solve the problem (2.37) projected onto these subspaces. Thus, we define $\boldsymbol{\mathcal{V}}^h \subset \boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{V}}_g^h \subset \boldsymbol{\mathcal{V}}_g$, $\mathcal{Q}^h \subset \mathcal{Q}$ and in every Picard iteration, we look for $\boldsymbol{u}_h^{k+1} \in \boldsymbol{\mathcal{V}}_g^h$, $p_h \in \mathcal{Q}^h$ such that

$$\begin{aligned} \int_\Omega (\boldsymbol{u}_h^k \cdot \nabla \boldsymbol{u}_h^{k+1}) \cdot \boldsymbol{v}_h + \nu \int_\Omega \nabla \boldsymbol{u}_h^{k+1} : \nabla \boldsymbol{v}_h - \int_\Omega p_h^{k+1} \nabla \cdot \boldsymbol{v}_h = 0 \qquad & \forall \boldsymbol{v}_h \in \boldsymbol{\mathcal{V}}^h, \\ \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h^{k+1} = 0 \qquad & \forall q_h \in \mathcal{Q}^h. \end{aligned} \tag{2.38}$$

Further, we introduce a set of velocity basis functions $\{\boldsymbol{\varphi}_i^u\}$ and pressure basis functions $\{\varphi_i^p\}$. Let us assume, for simplicity, that the vector-valued velocity basis functions $\boldsymbol{\varphi}_i^u$, consist of d equal components denoted as φ_i^u. We express the approximate velocity $\boldsymbol{u}_h^{k+1} \in \boldsymbol{\mathcal{V}}_g^h$ and pressure $p_h \in \mathcal{Q}^h$ in the form

$$\boldsymbol{u}_h^{k+1} = \sum_{i=1}^{n_u} \mathbf{u}_i^{k+1} \varphi_i^u + \sum_{i=n_u+1}^{n_u^*} \mathbf{u}_i^* \varphi_i^u, \tag{2.39}$$

$$p_h^{k+1} = \sum_{i=1}^{n_p} p_i^{k+1} \varphi_i^p, \tag{2.40}$$

such that $\sum_{i=1}^{n_u} \mathbf{u}_i^{k+1} \varphi_i^u \in \boldsymbol{\mathcal{V}}^h$. The coefficients $\mathbf{u}_i^{k+1} \in \mathbb{R}^d$ and $p_i^{k+1} \in \mathbb{R}$ are the unknown velocity and pressure coefficients and the coefficients $\mathbf{u}_i^* \in \mathbb{R}^d$ are known, assuming that

$$\boldsymbol{g} = \sum_{i=n_u+1}^{n_u^*} \mathbf{u}_i^* \varphi_i^u \tag{2.41}$$

holds on the domain boundary $\partial\Omega$.

The approach described above is referred to as strong imposition of the Dirichlet boundary conditions. Let us mention that it is not the only possible treatment of the Dirichlet boundary conditions in methods based on Galerkin method. Another approach is based on considering $\boldsymbol{u}, \boldsymbol{v} \in [H^1(\Omega)]^d$ instead of the solution and test spaces defined in (2.26) and augmenting the variational formulation by terms that enforce the Dirichlet boundary conditions in a weak sense. As we do not consider this approach in this work, we refer to the literature for more details, see e.g. [2].

After substituting (2.39) and (2.40) into the formulation (2.38), we obtain a system of linear algebraic equations that can be written in the matrix form

$$\begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{k+1} \\ \mathbf{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{2.42}$$

where the right hand side comes from the Dirichlet boundary conditions.

Consider the test functions in the form $\varphi_i^u \vec{e}_m$ for $m = 1, \ldots, d$. Then the matrix $\mathbf{F} \in \mathbb{R}^{d \cdot n_u \times d \cdot n_u}$ is block diagonal with d equal diagonal blocks,

$$\mathbf{F} = \operatorname{diag}\Big(\underbrace{\mathbf{N}(\boldsymbol{u}_h^k) + \mathbf{A}, \ldots, \mathbf{N}(\boldsymbol{u}_h^k) + \mathbf{A}}_{d}\Big), \tag{2.43}$$

where the diagonal blocks $\mathbf{N}(\boldsymbol{u}_h^k) + \mathbf{A}$ consist of the discretization of the linearized convective term and the viscous term. $\mathbf{B}^T \in \mathbb{R}^{d \cdot n_u \times n_p}$ and $\mathbf{B} \in \mathbb{R}^{n_p \times d \cdot n_u}$ are discrete gradient and negative divergence operators, respectively. The elements of the blocks $\mathbf{N}(\mathbf{u}_h^k)$ and $\mathbf{A}$ are as follows

$$\begin{aligned} \mathbf{N}(\boldsymbol{u}_h^k) = [N_{ij}(\boldsymbol{u}_h^k)] &= \left[\int_\Omega \varphi_i^u \big(\boldsymbol{u}_h^k \cdot \nabla \varphi_j^u \big) \right], \\ \mathbf{A} = [A_{ij}] &= \left[\nu \int_\Omega \nabla \varphi_i^u \cdot \nabla \varphi_j^u \right]. \end{aligned} \tag{2.44}$$

The matrix $\mathbf{B}$ consists of d blocks $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_d]$, where

$$\mathbf{B}_m = [B_{m,ij}] = \left[\int_\Omega \varphi_i^p \big(\nabla \varphi_j^p \cdot \vec{e}_m \big) \right] \quad \text{for } m = 1, \ldots, d. \tag{2.45}$$

Particular discretization methods based on Galerkin method are defined by the choice of the subspaces and their basis functions. The most popular example is the finite element method (FEM), where the domain Ω is first divided into simple subdomains (elements) forming a computational mesh. As a consequence, the domain boundary $\partial\Omega$ has to be approximated in most cases. The basis functions are defined as piecewise polynomial functions, polynomial in the interior of each element, with local support containing only a few elements. This leads to a sparse coefficient matrix of the resulting linear system.

The inf-sup condition

It follows from the theory of saddle-point problems that the finite dimensional spaces $\boldsymbol{\mathcal{V}}^h$ and $\mathcal{Q}^h$ cannot be chosen arbitrarily. In order for the discretized problem (2.38) to be well-posed, the spaces $\boldsymbol{\mathcal{V}}^h$ and $\mathcal{Q}^h$ have to satisfy the so-called discrete inf-sup condition

$$\inf_{q_h \in \mathcal{Q}^h \setminus \{0\}} \sup_{\boldsymbol{v}_h \in \boldsymbol{\mathcal{V}}^h \setminus \{\mathbf{0}\}} \frac{\int_\Omega q_h \nabla \cdot \boldsymbol{v}_h}{||\boldsymbol{v}_h||_{\boldsymbol{\mathcal{V}}^h} ||q_h||_{\mathcal{Q}^h}} \geq \gamma > 0, \tag{2.46}$$

where γ is a constant independent of the mesh. In standard finite elements, so-called inf-sup stable pairs of finite element spaces are used to ensure that this condition is satisfied. Some of the most popular are the Taylor–Hood [98], Nédélec [77] or Raviart–Thomas [85] element. Alternatively, the inf-sup condition can be circumvented by suitable stabilization. Some examples of stabilized elements are given in [40].

2.2.2 The time-dependent problem

In this section, we briefly describe the discretization and linearization of the initial-boundary value problem for the time-dependent Navier–Stokes equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{0} \qquad \text{in } \Omega \times (0,T), \tag{2.47}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in } \Omega \times (0,T), \tag{2.48}$$

$$\boldsymbol{u} = \boldsymbol{g} \qquad \text{on } \partial\Omega_D \times [0,T], \tag{2.49}$$

$$\nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - \boldsymbol{n} p = \boldsymbol{0} \qquad \text{on } \partial\Omega_N \times [0,T], \tag{2.50}$$

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0 \qquad \text{in } \Omega, \tag{2.51}$$

where $\boldsymbol{u}_0 = \boldsymbol{u}_0(\boldsymbol{x})$ is a given initial condition. The Dirichlet boundary condition $\boldsymbol{g}$ can be generally dependent on both space and time variable, however, we will assume that it depends only on the space variable, i.e., $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x})$. The discretization considered in this work consists of discretization in time first, leading to a sequence of spatial problems that are linearized and discretized using Galerkin method similarly to the stationary problem.

We can choose any time stepping scheme for the time discretization. One of the simplest choice is the one-stage finite difference discretization using the Θ-scheme which includes the well known Crank-Nicolson method and backward Euler method. Unlike backward Euler method, which is only first-order accurate, the Crank-Nicolson method is second-order accurate. Both mentioned schemes are unconditionally stable, however, the approximate solution obtained with the Crank-Nicolson method can still contain some spurious oscillations. It can require very small time steps to avoid the oscillations, leading to strict condition on the time step size similar as for the explicit (forward) Euler method. The backward Euler method is immune to oscillations and allows large time steps. It is especially suitable for problems where time accuracy is not of importance, i.e., if we are interested only in the steady state. We refer to [40] for more details on time discretization of the time-dependent Navier–Stokes equations, including adaptive time stepping.

Here we consider the backward Euler method with a constant time step Δt. Denoting $\boldsymbol{u}^n, p^n$ the velocity and pressure at the n-th time step, respectively, we obtain the following set of equations

$$\begin{aligned} \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} + \boldsymbol{u}^{n+1} \cdot \nabla \boldsymbol{u}^{n+1} - \nu \Delta \boldsymbol{u}^{n+1} + \nabla p^{n+1} &= \boldsymbol{0} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u}^{n+1} &= 0 \qquad \text{in } \Omega. \end{aligned} \tag{2.52}$$

Analogously to the previous section, we derive the weak formulation: find $\boldsymbol{u} \in \boldsymbol{\mathcal{V}}_g$

and $p \in \mathcal{Q}$ such that

$$\frac{1}{\Delta t}\int_\Omega \boldsymbol{u}^{n+1}\cdot\boldsymbol{v} + \int_\Omega (\boldsymbol{u}^{n+1}\cdot\nabla\boldsymbol{u}^{n+1})\cdot\boldsymbol{v} + \nu\int_\Omega \nabla\boldsymbol{u}^{n+1} : \nabla\boldsymbol{v} - \int_\Omega p^{n+1}\nabla\cdot\boldsymbol{v} = \frac{1}{\Delta t}\int_\Omega \boldsymbol{u}^{n}\cdot\boldsymbol{v},$$
$$\int_\Omega q\nabla\cdot\boldsymbol{u}^{n+1} = 0, \tag{2.53}$$

for all $\boldsymbol{v} \in \boldsymbol{\mathcal{V}}$ and $q \in \mathcal{Q}$. Note that the first equation was just rearranged by moving the term with $\boldsymbol{u}^n$ to the right-hand side.

The problem (2.53) is still nonlinear and needs to be linearized. A simple linearization strategy in this case is to replace the unknown velocity in the convective term by the velocity from the previous time step $\boldsymbol{u}^n$, i.e., to replace the convection term by the integral $\int_\Omega(\boldsymbol{u}^n\cdot\nabla\boldsymbol{u}^{n+1})\cdot\boldsymbol{v}$. Alternatively, we can apply the Picard iteration method at each time level, which means solving several spatial problems in every time step instead of one to obtain an approximation of $\boldsymbol{u}^{n+1}$. For simplicity of notation, let us drop the index $n+1$ and denote $\boldsymbol{u}^k$ the k-th approximation of $\boldsymbol{u}^{n+1}$. Thus, in each Picard iteration, we search $\boldsymbol{u}^{k+1} \in \boldsymbol{\mathcal{V}}_g$ and $p^{k+1} \in \mathcal{Q}$ as a solution of the following Oseen problem

$$\frac{1}{\Delta t}\int_\Omega \boldsymbol{u}^{k+1}\cdot\boldsymbol{v} + \int_\Omega (\boldsymbol{u}^{k}\cdot\nabla\boldsymbol{u}^{k+1})\cdot\boldsymbol{v} + \nu\int_\Omega \nabla\boldsymbol{u}^{k+1} : \nabla\boldsymbol{v} - \int_\Omega p^{k+1}\nabla\cdot\boldsymbol{v} = \frac{1}{\Delta t}\int_\Omega \boldsymbol{u}^{n}\cdot\boldsymbol{v},$$
$$\int_\Omega q\nabla\cdot\boldsymbol{u}^{k+1} = 0, \tag{2.54}$$

for all $\boldsymbol{v} \in \boldsymbol{\mathcal{V}}$ and $q \in \mathcal{Q}$, where $\boldsymbol{u}^0 = \boldsymbol{u}^n$. Note that performing only one Picard iteration corresponds to the simple linearization strategy mentioned above.

The Galerkin discretization of the problem (2.54) is also analogous to the steady-state problem and leads to a linear system of the same structure as (2.42). For the time-dependent problem, the diagonal blocks of the matrix $\mathbf{F}$ include the (scaled) velocity mass matrix,

$$\mathbf{F} = \operatorname{diag}\Big(\underbrace{\frac{1}{\Delta t}\mathbf{M}_u + \mathbf{N}(\boldsymbol{u}_h^k) + \mathbf{A}, \ldots, \frac{1}{\Delta t}\mathbf{M}_u + \mathbf{N}(\boldsymbol{u}_h^k) + \mathbf{A}}_{d}\Big), \tag{2.55}$$

where

$$\mathbf{M}_u = [M_{u,ij}] \quad = \left[\int_\Omega \varphi_i^u\varphi_j^u\right]. \tag{2.56}$$

Further, the vector $\mathbf{f}$ on the right hand side of the resulting linear system includes the discretization of the term $\frac{1}{\Delta t}\int_\Omega \boldsymbol{u}^n\cdot\boldsymbol{v}$.

Chapter 3

Isogeometric analysis

One of the most widely used approaches to solving problems modeled by partial differential equations is the finite element method (FEM). As already mentioned above, it is a discretization method based on the Galerkin method, where the weak formulation of the problem is derived and then solved in finite dimensional subspaces of the corresponding solution and test spaces. The subspaces are defined as a linear span of suitable basis functions. In order to define the FEM basis functions and thus the finite element spaces, the computational domain Ω is subdivided into simple non-overlapping subdomains called elements, e.g., triangles, quadrilaterals, tetrahedra, etc., forming a computational mesh.

The process of creating the mesh takes several steps. In industrial practice, the computational domain is usually a complex model created using CAD (Computer Aided Design) tools. Such geometries typically consist of trimmed surfaces with small gaps and overlaps and therefore they are not analysis-suitable. Thus, before the meshing step, an analysis-suitable geometry has to be created. Then the computational mesh is generated, which is only an approximation of the original geometry in most cases, since only simple domains (e.g. polygons in two dimensions), can be described exactly as a union of simple finite elements. This approximation can result in errors in the subsequent analysis, especially in some applications such as analysis of thin shells, problems with solutions containing boundary layers, etc. Moreover, it is often necessary to refine the mesh or create a new mesh during the analysis phase which requires access to the original CAD model. This complicates, for example, the use of adaptive mesh refinement or automatic shape optimization.

Hughes et al. [63, 23] proposed a new discretization approach called isogeometric analysis (IgA) with the aim to unify CAD and finite element analysis. The main goals of this approach are to always work with the exact geometry no matter how coarse the discretization and to simplify the refinement process. It is based on the Galerkin method together with the isoparametric concept, which means that the basis functions used for the geometry representation are also used as basis for the solution space. IgA is based on NURBS (Non-Uniform Rational B-Splines), which is a standard technology used to represent the geometry in CAD.

One of the main differences between FEM and IgA is that the classical FEM basis is interpolatory and thus the degrees of freedom correspond to the nodal values of the finite element solution, which is not true for the NURBS-based IgA. Another difference is that the FEM solution is only C^0-continuous across the element boundaries, whereas the IgA solution can be generally of higher-order continuity. On the other hand, one of

the important common features of the two methods is a compact support of the basis functions which results in sparse linear systems. However, the sparsity pattern is not the same for FEM and high-continuity IgA due to larger overlaps of the IgA basis function supports.

In this chapter, we outline the basics of B-spline and NURBS and briefly describe how they are used as a basis for isogeometric analysis. We also mention several inf-sup stable combinations of IgA bases that can be used for the discretization of the Navier–Stokes equations.

3.1 Preliminaries

We begin this section with the definition of a spline function and B-splines as a basis of the spline space. We mention some of the important features of B-splines and describe how they are used to represent geometric objects like curves and surfaces. We also define NURBS as a rational generalization of B-splines. Further details on the properties of B-spline and NURBS objects and some related algorithms can be found in Piegl and Tiller [84].

3.1.1 Splines and B-splines

Consider a partition of an interval $[a, b] \subset \mathbb{R}$ into ℓ subintervals determined by a vector $\mathbf{x} = (x_0, x_1, \ldots, x_\ell)$, where $a = x_0 < x_1 < \ldots < x_{\ell-1} < x_\ell = b$. Further consider a nonnegative integer k and a vector of regularities $\mathbf{r} = (r_1, r_2, \ldots, r_{\ell-1})$. A spline function $f : [a, b] \to \mathbb{R}$ is a piecewise polynomial function that satisfies the following conditions:

- f is a polynomial of degree at most k in $[x_i, x_{i+1})$ for $i = 0, \ldots, \ell - 1$,
- f has r_i continuous derivatives at x_i for $i = 1, \ldots, \ell - 1$.

The space of all such spline functions is denoted as $\mathcal{S}_k^{\mathbf{r}}$.

A classical basis of the spline space $\mathcal{S}_k^{\mathbf{r}}$ are the B-splines (or basis splines). The B-splines are defined by the spline degree k and a so-called knot vector Ξ. The knot vector is obtained from the vector $\mathbf{x}$ by repeating x_i such that its multiplicity is $m_i = k - r_i$ for $i = 1, \ldots, \ell - 1$. In IgA, we usually work with open knot vectors, where the first and the last knot are repeated $(k + 1)$ times which corresponds to no continuity conditions at the endpoints of the interval $[a, b]$. Thus, the (open) knot vector takes the form

$$\begin{aligned} \Xi &= (\underbrace{x_0, \ldots, x_0}_{k+1}, \underbrace{x_1, \ldots, x_1}_{k-r_1}, \ldots, \underbrace{x_{\ell-1}, \ldots, x_{\ell-1}}_{k-r_{\ell-1}}, \underbrace{x_\ell, \ldots, x_\ell}_{k+1}) \\ &= (\xi_1, \xi_2, \ldots, \xi_{n+k+1}), \end{aligned} \tag{3.1}$$

where $\xi_i \le \xi_{i+1}$ are the knots and n is the dimension of $\mathcal{S}_k^{\mathbf{r}}$. The i-th B-spline of degree k, $N_{i,k}(\xi), i = 1, 2, \ldots, n$, for the knot vector Ξ is defined recursively as follows

$$\begin{aligned} N_{i,0}(\xi) &= \begin{cases} 1, & \xi_i \le \xi < \xi_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \\ N_{i,k}(\xi) &= \omega_{i,k}(\xi) N_{i,k-1}(\xi) + (1 - \omega_{i+1,k}(\xi))\, N_{i+1,k-1}(\xi), \qquad \text{for } k > 0, \end{aligned} \tag{3.2}$$

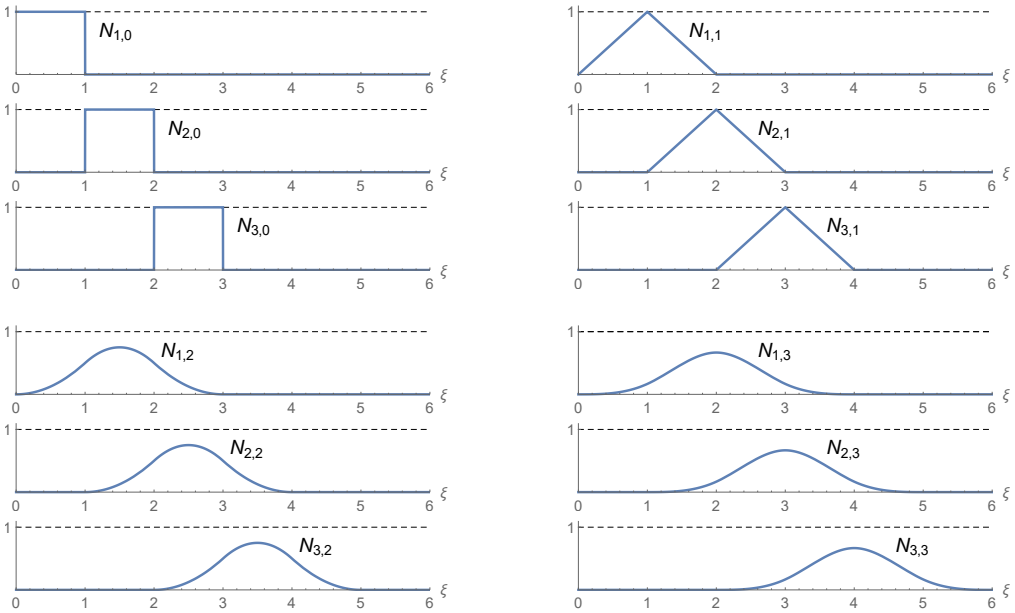
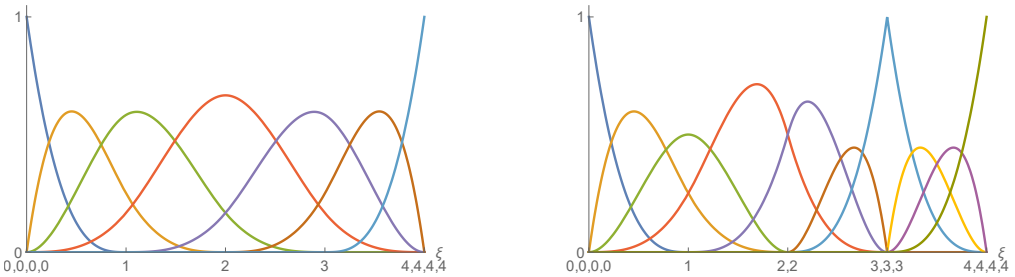


Figure 3.1: B-splines of degree $k = 0, 1, 2, 3$ for a uniform knot vector $\Xi = (0, 1, 2, 3, 4, 5, 6)$.

where

$$\omega_{i,k}(\xi) = \begin{cases} \frac{\xi - \xi_i}{\xi_{i+k} - \xi_i}, & \xi_i \neq \xi_{i+k}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

Figure 3.1 shows the first three B-spline basis functions (for i = 1, 2, 3) of degree $k = 0, 1, 2, 3$ for a uniform vector $\Xi = (0, 1, 2, 3, 4, 5, 6)$. Figure 3.2 shows two examples of a cubic ($k = 3$) B-spline basis for two open knot vectors. The knot vector of the basis in Figure 3.2a is $\Xi = (0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4)$, i.e., all internal knots have multiplicity 1, which results in the highest possible continuity C^2 over the whole interval (0,4). The knot vector of the basis in Figure 3.2b is $\Xi = (0, 0, 0, 0, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4)$, thus the continuity of the basis functions is C^2 at $\xi = 1$, C^1 at $\xi = 2$ and C^0 at $\xi = 3$.



(a) A cubic basis with C^2 continuity at all internal knots.

(b) A cubic basis with continuity C^2 at $\xi = 1$, C^1 at $\xi = 2$ and C^0 at $\xi = 3$.

Figure 3.2: Cubic ($k = 3$) B-spline bases for two different open knot vectors. The knots are indicated on the ξ-axis of the graphs, including their multiplicity.

The B-spline basis functions have several important features. The basis function $N_{i,k}(\xi)$ is nonzero only in the interval $[\xi_i, \xi_{i+k+1}]$, i.e., the support of each basis function of degree k is $k + 1$ knot spans. In any given knot span, at most $k + 1$ basis functions

are nonzero. They are all pointwise nonnegative and form a partition of unity on the interval $[\xi_{k+1}, \xi_{n+1}]$, i.e.,

$$\forall \xi \in [\xi_{k+1}, \xi_{n+1}] : \quad \sum_{i=1}^{n} N_{i,k}(\xi) = 1. \tag{3.4}$$

Note that for an open knot vector $[\xi_{k+1}, \xi_{n+1}] = [a, b]$.

Multivariate spline functions are defined by tensor-product of univariate splines. For example, given two nonnegative integers k, l, two vectors of regularities $\mathbf{r}_1, \mathbf{r}_2$ and the corresponding knot vectors $\Xi = (\xi_1, \xi_2, \ldots, \xi_{n+k+1}), \Psi = (\psi_1, \psi_2, \ldots, \psi_{m+l+1})$, the space of bivariate spline functions is defined as

$$\mathcal{S}_{k,l}^{\mathbf{r}_1,\mathbf{r}_2} = \mathcal{S}_{k}^{\mathbf{r}_1} \otimes \mathcal{S}_{l}^{\mathbf{r}_2}, \tag{3.5}$$

with tensor-product B-spline basis functions $N_{i,k}(\xi)N_{j,l}(\psi)$. For more details on splines, we refer to de Boor [25].

3.1.2 B-spline objects

The B-spline basis can be used to describe objects like curves, surfaces, solids, etc., in d-dimensional space $\mathbb{R}^d$. For example, a B-spline curve of degree k is constructed as a linear combination of the B-spline basis functions

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i N_{i,k}(\xi), \tag{3.6}$$

where the coefficients $\mathbf{P}_i \in \mathbb{R}^d, i = 1, \ldots, n$, are called control points. The piecewise linear interpolation of the control points is referred to as control polygon.

An example of a B-spline curve in $\mathbb{R}^2$, together with its control points and control polygon, is shown in Figure 3.3. It is a piecewise cubic curve constructed as a linear combination of the basis from Figure 3.2b corresponding to the open knot vector $\Xi = (0, 0, 0, 0, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4)$. Note that the curve is generally not interpolatory at the control points, except the first and last control point and also the seventh control point, where the curve is only C^0-continuous. Furthermore, the control polygon is tangent to the curve at the first, last and seventh control point, and also at the knot $\xi = 2$, where the curve is C^1-continuous.

The features of the B-spline basis functions result in several important features of the B-spline curves. For example, they satisfy a strong convex hull property, which means that the B-spline curve of degree k is contained in the union of all convex hulls of $k + 1$ neighboring control points. Another important property is the variation diminishing property meaning that no plane has more intersections with the curve than with the control polygon.

Similarly to a curve, a B-spline surface is defined as a linear combination of the tensor-product B-spline basis functions

$$\mathbf{S}(\xi, \psi) = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{P}_{i,j} N_{i,k}(\xi) N_{j,l}(\psi), \tag{3.7}$$

where $\mathbf{P}_{i,j} \in \mathbb{R}^d, i = 1, \ldots, n, j = 1, \ldots, m$, are the control points forming a bidirectional control net. A B-spline solid is constructed analogously.

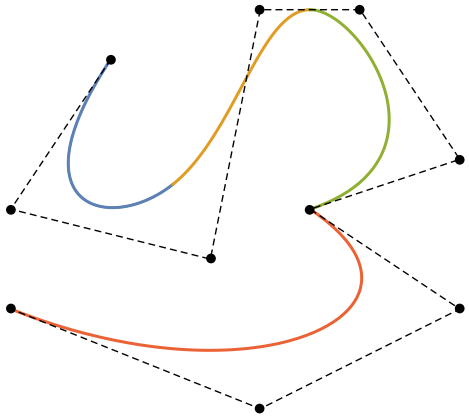


Figure 3.3: A cubic B-spline curve in $\mathbb{R}^2$ constructed as a linear combination of the basis shown in Figure 3.2b, its control points (black dots) and control polygon (dashed line). The parts of the curve corresponding to different knot spans are distinguished by different colors.

3.1.3 NURBS

Since B-splines are piecewise polynomial, it is not possible to represent some elementary shapes like circle or ellipse as B-spline objects. Therefore NURBS objects were introduced. The piecewise rational NURBS basis functions are given by

$$R_{i,k}(\xi) = \frac{w_i N_{i,k}(\xi)}{\sum_{j=1}^{n} w_j N_{j,k}(\xi)}, \tag{3.8}$$

where $N_{i,k}(\xi)$ is the i-th B-spline basis function of degree k defined in (3.2) and w_i is the i-th weight. The weights are usually assumed to be positive real numbers. Note that if all weights are equal, then $R_{i,k}(\xi) = N_{i,k}(\xi)$ for $i = 1, \ldots, n$, thanks to the partition of unity property, i.e., B-splines are special cases of NURBS.

Similarly to B-spline objects, the NURBS objects are constructed as linear combinations of the NURBS basis functions with control points as coefficients. For example, a quarter circle can be represented as a NURBS curve with $k = 2$ for a knot vector $\Xi = (0, 0, 0, 1, 1, 1)$, the weights $w_1 = w_3 = 1, w_2 = \sqrt{2}/2$ and a control polygon consisting of two perpendicular line segments of equal length. The quarter circle is shown in Figure 3.4 (in red color) together with another three NURBS curves obtained for different values of w_2. For $w_2 = 1$, the curve is a quadratic B-spline curve.

Since we do not work with NURBS discretizations in this work, we limit ourselves to B-spline basis in the rest of this thesis.

3.2 B-splines as a basis for IgA

The computational domain Ω can be represented a single B-spline object or it may consist of multiple separate B-spline objects called patches, which is common for complex geometries in industrial practice. Such geometry is referred to as a multipatch. Each patch has its own parametric space and basis independent of the other patches. For the purpose of analysis, it is necessary to connect the patches in some way. One possibility

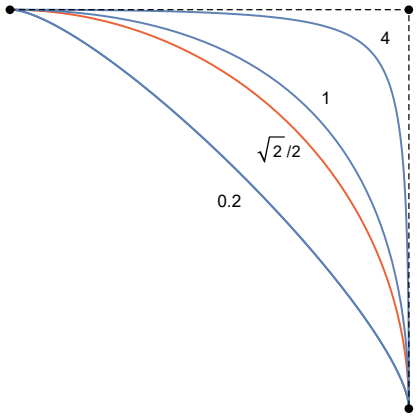


Figure 3.4: NURBS curves of degree $k = 2$ for a knot vector $\Xi = (0, 0, 0, 1, 1, 1)$. The weights $w_1 = w_3 = 1$ for all four curves and w_2 varies (its values are displayed beside the corresponding curves). The quarter circle is displayed in red.

is to consider conforming patches, which means that the bases on both sides of the patch interfaces match. Thus, the control points corresponding to the basis functions at the interface can be identified with the control points of the respective basis functions at the other side of the interface. Let us emphasize that the geometry (and also the IgA solution) is only C^0-continuous across the interfaces in this case. Another way to connect the patches is to use the discontinuous Galerkin method at the patch level, where the connection is enforced weakly [70]. This method can be used also for nonconforming patches. However, in this work, we consider only conforming patches connected via identifying the control points at the interfaces.

3.2.1 Computational mesh and matrix assembly

For simplicity, let us consider a two-dimensional domain $\Omega \subset \mathbb{R}^2$ as an illustrative example. Let Ω be described as a single B-spline patch $\mathbf{G}(\xi, \psi)$ over a parametric domain $\widehat{\Omega}$ defined by given knot vectors Ξ, Ψ,

$$\begin{aligned} &\mathbf{G}(\xi,\psi) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{P}_{i,j} N_{i,k}(\xi) N_{j,l}(\psi) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{P}_{i,j} Q_{i,j}^{k,l}(\xi,\psi), \\ &\mathbf{G} : (\xi,\psi) \mapsto (x,y), \quad (\xi,\psi) \in \widehat{\Omega}, \quad (x,y) \in \Omega, \end{aligned} \tag{3.9}$$

where we denoted the product of two univariate B-spline basis functions $N_{i,k}(\xi)N_{j,l}(\psi)$ as one bivariate function $Q_{i,j}^{k,l}(\xi, \psi)$. For convenience, let us re-index the control points and basis functions with one index such that

$$\mathbf{G}(\xi,\psi) = \sum_{i=1}^{N} \mathbf{P}_i Q_i^{k,l}(\xi,\psi), \quad N = n \cdot m. \tag{3.10}$$

A computational mesh is given by the product of the knot vectors $\Xi \times \Psi$. Thus, if the subsequent knots are different from each other, i.e., $\xi_i \neq \xi_{i+1}$ and $\psi_j \neq \psi_{j+1}$ for some given i, j, then $[\xi_i, \xi_{i+1}] \times [\psi_j, \psi_{j+1}]$ defines an element. In the parametric space, the elements form a structured rectangular mesh, and its image in the physical space is generally a curvilinear tensor-product mesh.

The IgA discrete spaces on the physical domain Ω are generated by the push-forwards of the B-spline basis functions $Q_i^{k,l}(\xi,\psi)$ from the parametric space, i.e., by the functions

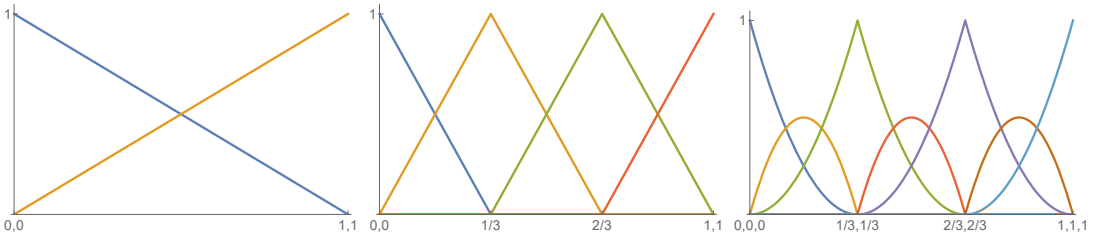
$$\widetilde{Q}_i^{k,l}(x,y) = (Q_i^{k,l} \circ \mathbf{G}^{-1})(x,y), \quad r = 1,\dots,N, \tag{3.11}$$

where $\mathbf{G}^{-1} : \Omega \to \widehat{\Omega}$. The transformation (3.11) is nothing but a change of variables. The coefficient matrices in IgA can be assembled in a similar way as in standard FEM, that is, the integrals over Ω are computed element by element. The element integrals are transformed into the parametric space using classical change of variables rules and computed using Gaussian quadrature. We obtain a local coefficient matrix from each element. Finally, the local matrices from all elements are assembled into a global matrix. However, the computational costs of such assembly can be very high, especially for higher degree discretizations, and therefore specialized assembly procedures for IgA have been developed, see, for example [15, 74].

3.2.2 Mesh refinement

An important feature of isogeometric analysis is the possibility to enrich the discretization space in several ways such that the geometry of the computational domain can be still represented exactly in the basis of the enriched space. Also the parametrization of the geometry stays unchanged. The analogues of the finite element h-refinement and p-refinement are knot insertion and degree elevation, respectively. Moreover, IgA offers another refinement strategy referred to as k-refinement.

Knot insertion and degree elevation belong to fundamental B-spline/NURBS algorithms described in [84]. For a B-spline curve of degree k in the form (3.6), the knot insertion algorithm consists of inserting new knots into the knot vector Ξ, generating a new B-spline basis of degree k using (3.2) and computing new control points from the original ones such that the shape and parametrization of the curve is not changed. Figure 3.5a shows an example of a linear B-spline basis for a knot vector $\Xi = (0,0,1,1)$ and a refined basis obtained by inserting new knots $\xi_1 = 1/3$ and $\xi_2 = 2/3$ into the knot vector is shown in Figure 3.5b.



(a) Linear basis for knot vector $\Xi = (0,0,1,1)$.

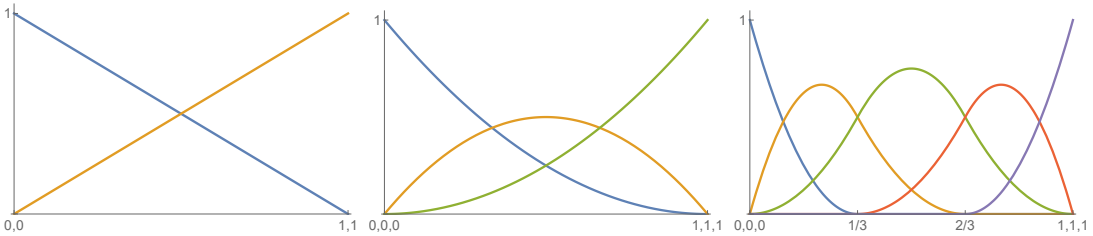
(b) Linear basis for knot vector $\Xi = (0,0,\frac{1}{3},\frac{2}{3},1,1)$.

(c) Quadratic basis for $\Xi = (0,0,0,\frac{1}{3},\frac{1}{3},\frac{2}{3},\frac{2}{3},1,1,1)$

Figure 3.5: An example of basis refinement by knot insertion ((a)-(b)) and degree elevation ((b)-(c)).

The degree elevation is achieved by increasing the multiplicity of all knots by one in order to preserve the continuity of the curve, constructing a new B-spline basis of degree $k+1$ for the new knot vector and finding new control points for the obtained basis. Figure 3.5c shows an example of a quadratic B-spline basis obtained from the basis in Figure 3.5b by degree elevation.

In many situations, we would like to perform both h- and p-refinement. In isogeometric analysis, these operations do not commute. If we first refine the mesh by inserting new knots and elevate the basis degree afterwards, the number of basis functions will increase considerably and the resulting basis will have a decreased continuity at each knot. On the contrary, elevating the degree of the basis on the coarsest level and then inserting new knots into the knot vector results in less basis functions with less points of decreased continuity. The latter strategy is denoted as k-refinement [63]. An example of k-refinement is shown in Figure 3.6 (compare with Figure 3.5).



(a) Linear basis for knot vector $\Xi = (0, 0, 1, 1)$.

(b) Quadratic basis for knot vector $\Xi = (0, 0, 0, 1, 1, 1)$.

(c) Quadratic basis for $\Xi = (0, 0, 0, \frac{1}{3}, \frac{2}{3}, 1, 1, 1)$

Figure 3.6: An example of k-refinement.

Of course, all mentioned algorithms can be generalized for surfaces etc. Note that the h-refinement using knot insertion does not allow true local refinement. This is not a consequence of using IgA itself, but rather that the B-spline/NURBS-based IgA is limited to tensor-product meshes. By inserting knots, the mesh is always refined in the sense of tensor refinement and in the case of multipatch with conforming interfaces, the refinement propagates to adjacent patches. The situation is similar in standard FEM with tensor-product meshes. True local refinement in IgA is an active research area, there are several generalizations of B-splines that allow local refinement, such as T-splines [92], THB-splines [52] and LR-splines [27].

3.3 Poisson example

To illustrate the process of solving a PDE with isogeometric analysis, we describe the discretization and solution of a model Poisson problem in this section.

Let $\Omega \in \mathbb{R}^2$ be a B-spline approximation of a quarter annulus with the quarter circle boundaries approximated by quadratic B-spline curves as in Figure 3.4, where the B-spline curve corresponds to $w_2 = 1$. The domain can be described as a B-spline surface of degree $k = 1$ in the radial direction and degree $l = 2$ in the other direction with the knot vectors $\Xi = (0, 0, 1, 1)$ and $\Psi = (0, 0, 0, 1, 1, 1)$, respectively. Thus, the parametric domain is $\widehat{\Omega} = [0, 1]^2$ and the geometry mapping $\mathbf{G}(\xi, \psi)$ is defined as a linear combination of the tensor-product basis functions obtained from the bases displayed in Figures 3.6a and 3.6b. The parametric domain and the physical domain with its control points and control polygon are displayed in Figure 3.7. Note that the mesh on this domain consists of only one element.

Consider the following Poisson problem with a constant source function and constant Dirichlet conditions

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega, \\ u &= g \qquad \text{on } \partial\Omega, \end{aligned} \tag{3.12}$$

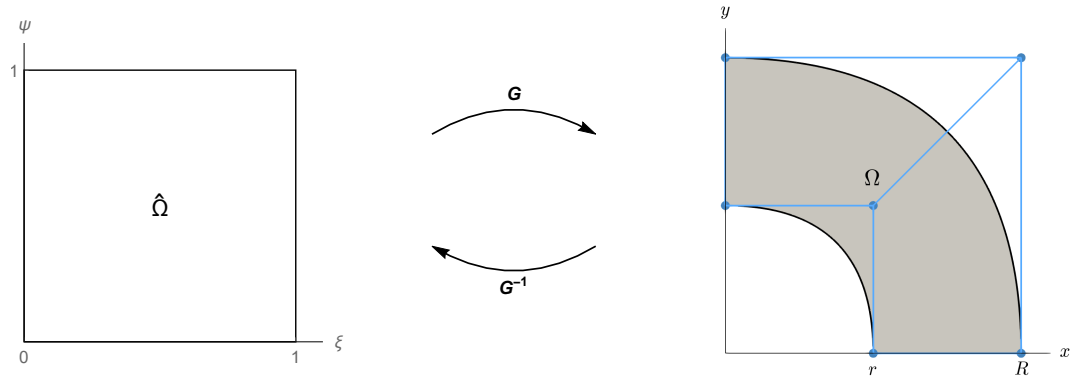


Figure 3.7: The parametric domain (left) and the physical domain (right) with its control points and control polygon (in blue).

where $f, g > 0$. The weak formulation is as follows: find $u \in \mathcal{V}_g$ such that for all $v \in \mathcal{V}$

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v, \tag{3.13}$$

where the solution and test spaces are given as

$$\begin{aligned} \mathcal{V}_g &= \{u \in H^1(\Omega) \,|\, u = g \text{ on } \partial\Omega\}, \\ \mathcal{V} &= \{v \in H^1(\Omega) \,|\, u = 0 \text{ on } \partial\Omega\}, \end{aligned} \tag{3.14}$$

respectively. Further, we define the IgA finite dimensional spaces $\mathcal{V}_g^h \subset \mathcal{V}_g$ and $\mathcal{V}^h \subset \mathcal{V}$ and search for an approximate solution $u_h \in \mathcal{V}_g^h$ such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega f v_h, \tag{3.15}$$

for all $v_h \in \mathcal{V}^h$. Let us emphasize that the finite dimensional spaces in IgA are defined on the actual physical domain Ω, in contrast with standard finite element method where Ω is often replaced by its approximation.

Assume that we want to approximate the solution on a 3×3 mesh using a B-spline basis of degree 2 with C^1 continuity in both directions. Since our representation of the geometry is only linear in the ξ-direction, we need to elevate the degree in this direction. We also need to insert two new knots into the knot vectors in both directions in order to obtain a 3×3 mesh. If we refined the knot vectors first (h-refinement) and elevated the degree in the ξ-direction afterwards (p-refinement), we would obtain a basis that is C^0-continuous at the newly inserted knots in the ξ-direction (see Figure 3.5). In order to obtain a basis with global C^1 continuity in both directions, we need to elevate the degree in the ξ-direction in the first place and then insert new knots (k-refinement, see Figure 3.6). Assuming uniform knot vectors, this results in $\Xi = \Psi = (0, 0, 0, \frac{1}{3}, \frac{2}{3}, 1, 1, 1)$. The left panel of Figure 3.8 shows the parametric domain with the refined mesh and the univariate B-spline bases forming the tensor-product basis and the right panel shows the physical domain with the refined mesh and its new control points and control polygon.

The refined spline space over the parametric domain is generated by 25 basis functions, where 9 of them vanish on the boundary and the remaining 16 have a nonzero value on some part of the boundary. Let us denote the basis functions as $Q_i(\xi, \psi), i = 1, \ldots, 25$,

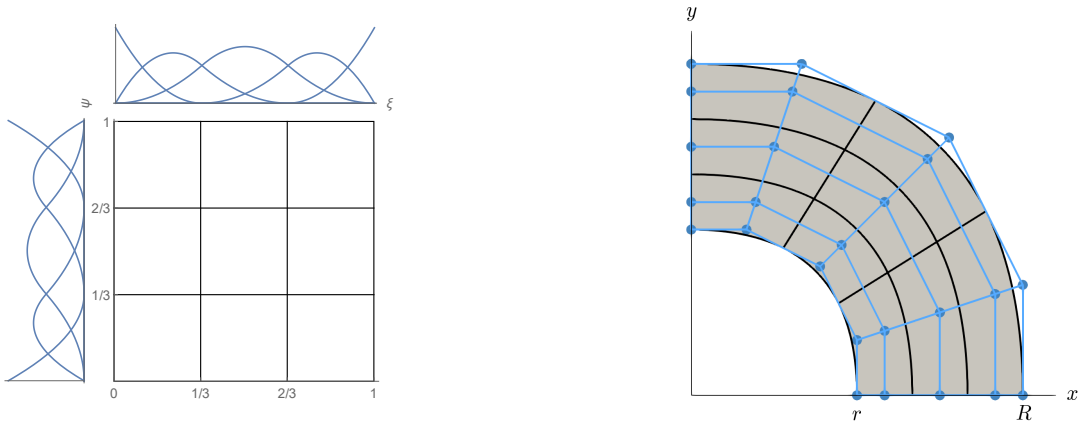


Figure 3.8: Left: The parametric domain with the refined mesh and the univariate B-spline bases forming the tensor-product basis over the parameter space. Right: The physical domain with the refined mesh (black) and the new control points and control polygon (blue).

such that the basis functions with zero value on the boundary are numbered first. As already mentioned in Section 3.2.1, the discrete spaces on the physical domain are generated by the functions

$$\widetilde{Q}_i(x,y) = (Q_i \circ \mathbf{G}^{-1})(x,y), \quad i = 1,\dots,25, \tag{3.16}$$

that are shown in Figure 3.9. The finite dimensional space $\mathcal{V}^h$ is defined as the space

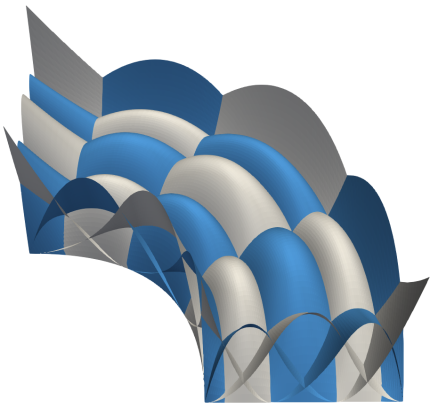


Figure 3.9: The push-forwards of the B-spline basis functions on the physical domain.

spanned by the basis functions vanishing on the boundary, i.e., $\mathcal{V}^h = \text{span}\{\widetilde{Q}_i\}_{i=1}^{9}$.

The approximate solution $u_h \in \mathcal{V}_g^h$ is constructed as a linear combination of all functions $\widetilde{Q}_i$. In this case, it is possible to satisfy the Dirichlet boundary condition exactly by setting the coefficients of all basis functions on the boundary equal to g. Thus, the solution is considered in the form

$$u_h = \sum_{j=1}^{9} u_j \widetilde{Q}_j + \widetilde{g}, \quad \text{where } \widetilde{g} = \sum_{j=10}^{25} g\, \widetilde{Q}_j, \tag{3.17}$$

$u_j \in \mathbb{R}$ are unknown coefficients and $\widetilde{g} = \widetilde{g}(x, y)$ is a function that has a constant value g on the whole boundary $\partial\Omega$. We substitute (3.17) into (3.15) and utilize the fact that (3.15) holds for all $v_h \in \mathcal{V}^h$ if it holds for all basis functions of $\mathcal{V}^h$. We obtain a linear system $\mathbf{Au} = \mathbf{f}$, where $\mathbf{A} \in \mathbb{R}^{9\times 9}$ and the elements of the matrix $\mathbf{A}$ and the right hand side $\mathbf{f}$ are

$$\mathbf{A} = [A_{ij}] = \left[\int_\Omega \nabla\widetilde{Q}_i \cdot \nabla\widetilde{Q}_j\right], \quad \mathbf{f} = [f_i] = \left[\int_\Omega (f - \widetilde{g})\,\widetilde{Q}_i\right]. \tag{3.18}$$

Using the standard element-by-element assembly procedure as in FEM, the integral over Ω in (3.15) is rewritten as a sum of integrals over the individual elements

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \sum_{e=1}^{9} \int_{\Omega_e} \nabla u_h \cdot \nabla v_h. \tag{3.19}$$

On each element, the indices of nonzero basis functions are identified and a local element matrix is computed. Let i, j be indices of two basis functions that are nonzero on a given element Ω_e. The integral over the element is transformed into the parametric variables,

$$\int_{\Omega_e} \nabla\widetilde{Q}_i \cdot \nabla\widetilde{Q}_j = \int_{\widehat{\Omega}_e} \nabla Q_i \cdot \nabla Q_j \,|\det \mathbf{J_G}|, \tag{3.20}$$

where $\mathbf{J_G}$ is the Jacobian matrix of the geometric mapping $\mathbf{G}$, and computed using Gauss quadrature. The contributions from all elements are assembled into the global matrix $\mathbf{A}$.

After solving the global linear system we obtain the coefficients u_i, $i = 1, \ldots, 9$, of the linear combination in (3.17). The approximate solution u_h can be also described as a B-spline surface in $\mathbb{R}^3$, that is

$$\begin{aligned} u_h(x, y) = \widehat{u}_h(\xi, \psi) = \sum_{i=1}^{25} \mathbf{P}_i Q_i(\xi, \psi), \quad (\xi, \psi) \in \widehat{\Omega} = [0, 1]^2, \\ (x, y) = \mathbf{G}(\xi, \psi), \end{aligned} \tag{3.21}$$

where the first two components of the control points $\mathbf{P}_i \in \mathbb{R}^3$ are equal to the components of the corresponding control points of the physical domain Ω displayed in Figure 3.8 (right) and the third component is u_i for $i = 1, \ldots, 9$, and g for $i = 10, \ldots, 25$. The approximate solution together with its control points and control net is shown in Figure 3.10.

It is interesting to take a closer look at the sparsity pattern of the matrix $\mathbf{A}$ and compare it to the matrices obtained from standard FEM. Since the mesh 3×3 is so coarse that one of the basis functions is nonzero on the whole domain, the global matrix for our example is dense. Therefore, we display the sparsity pattern of a matrix obtained for a finer mesh 7×7, see the left panel of Figure 3.11. For comparison, the sparsity pattern of a matrix for the same mesh with a biquadratic C^0-continuous B-spline basis is shown in the right panel of Figure 3.11, which corresponds to the sparsity pattern for the standard Q2 finite element. Obviously, the FEM-like matrix is larger since there are more basis functions than for IgA with higher continuity on the same mesh. Both matrices are banded, however, the band of the matrix for higher continuity basis is denser. The reason is that in standard FEM (or IgA with C^0 continuity), on each element, there are basis functions with supports containing only the given element, whereas the supports of higher-continuity IgA basis functions always contain more than one element. The

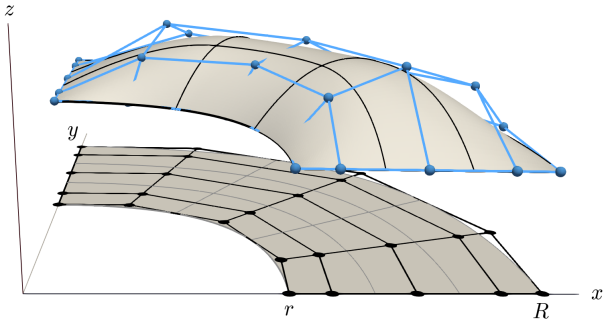


Figure 3.10: The approximate solution u_h with its control points and control net (in blue).

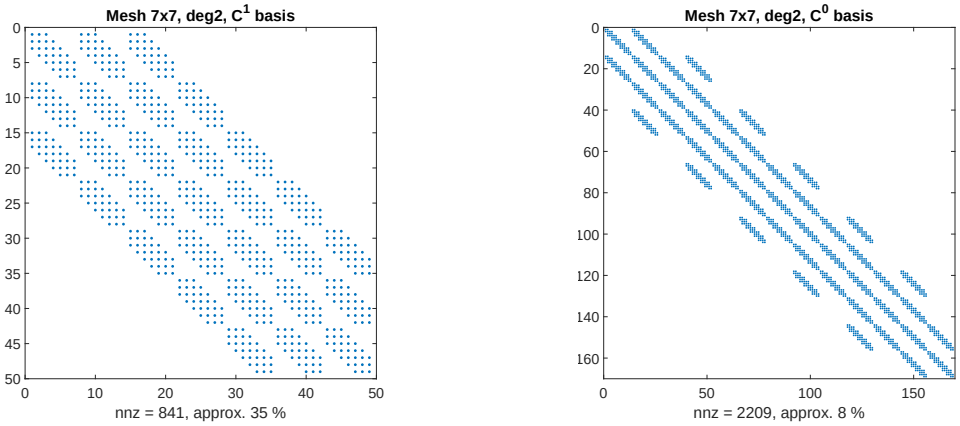


Figure 3.11: Sparsity pattern of the global matrix on a 7×7 mesh with a biquadratic C^1- (left) and C^0-continuous basis (right). The number of nonzero values (nnz) and their percentage is displayed for both matrices.

only exceptions are the basis functions that are nonzero on the boundary. Compare the bases in Figures 3.5c and 3.6c. This results in larger overlaps of the local element contributions.

Thus, smaller size of the IgA matrices does not mean that the solution of the linear systems will be equally expensive as of the FEM linear systems of the same size. The higher the continuity, the denser the matrices. For example, we compare the matrix in the left panel of Figure 3.11 with a matrix resulting from a C^0 biquadratic discretization on a 4×4 mesh which is of exactly the same size. However, this matrix has 529 nonzero elements, which is only 22 % of the total number of elements (compare with the number of nonzeros and their percentage displayed in Figure 3.11 (left)).

3.4 Discretization of the Navier–Stokes equations

The IgA discretization of the Navier–Stokes equations proceeds as described in Section 2.2. As already mentioned, the finite dimensional spaces $\boldsymbol{\mathcal{V}}^h$ and $\mathcal{Q}^h$ have to be chosen such that they satisfy the inf-sup condition (2.46). In this work, we do not consider the case where the inf-sup condition is circumvented by stabilization. The inf-sup stability of some particular combinations of IgA discrete spaces has been shown in the

literature. For example, three stable pairs of spline spaces that can be viewed as extensions of the Taylor–Hood (TH), Nédélec (N) and Raviart-Thomas (RT) elements were introduced by Buffa et al. in [13]. The isogeometric TH pair was already known from Bazilevs et al. [1]. Another stable isogeometric element referred to as subgrid (SG) element, was proposed by Bressan and Sangalli in [12]. The inf-sup stability of some other combinations of the discrete solution spaces were also addressed by Nielsen et al. in [78].

Consider a two-dimensional physical domain $\Omega \subset \mathbb{R}^2$ represented as a single B-spline patch. Further consider a B-spline basis obtained by h-, p- of k-refinement of the original basis in which Ω is described. Assume that the refined basis is of degree k in both directions and that all inner knots of its knot vectors Ξ and Ψ have the same multiplicity $m = k - r$, where $0 \leq r \leq k-1$. Denote the spline space generated by such basis as $\mathcal{S}^{r,r}_{k,k}$. Recall that the number r corresponds to C^r continuity at the inner knots. Denote the finite dimensional velocity and pressure space on the parametric domain $\widehat{\Omega}$ as $\widehat{\boldsymbol{\mathcal{V}}}^h$ and $\widehat{\mathcal{Q}}^h$, respectively. The velocity spaces on the parametric domain for the isogeometric Taylor-Hood, Nédélec and Raviart-Thomas element are given as

$$\begin{aligned} \widehat{\boldsymbol{\mathcal{V}}}^h_{TH} &= \mathcal{S}^{r,r}_{k+1,k+1} \times \mathcal{S}^{r,r}_{k+1,k+1}, \\ \widehat{\boldsymbol{\mathcal{V}}}^h_{N} &= \mathcal{S}^{r+1,r}_{k+1,k+1} \times \mathcal{S}^{r,r+1}_{k+1,k+1}, \\ \widehat{\boldsymbol{\mathcal{V}}}^h_{RT} &= \mathcal{S}^{r+1,r}_{k+1,k} \times \mathcal{S}^{r,r+1}_{k,k+1}, \end{aligned} \tag{3.22}$$

and the pressure spaces are the same for all elements,

$$\widehat{\mathcal{Q}}^h_{TH} = \widehat{\mathcal{Q}}^h_{N} = \widehat{\mathcal{Q}}^h_{RT} = \mathcal{S}^{r,r}_{k,k}. \tag{3.23}$$

An interesting feature of the isogeometric Raviart-Thomas pair is that the following equality holds

$$\left\{ \nabla \cdot \widehat{\boldsymbol{v}} \,|\, \widehat{\boldsymbol{v}} \in \widehat{\boldsymbol{\mathcal{V}}}^h_{RT} \right\} = \widehat{\mathcal{Q}}^h_{RT}, \tag{3.24}$$

which guarantees that exactly divergence-free velocities $\widehat{\boldsymbol{v}}$ are obtained. However, special care must be taken if the unknown velocity is searched in a subspace of $\widehat{\boldsymbol{\mathcal{V}}}^h_{RT}$, for example if there are Dirichlet boundary conditions imposed strongly. See [13] for more details.

The simplest way to construct the spaces $\boldsymbol{\mathcal{V}}^h$ and $\mathcal{Q}^h$ on the physical domain is to map the pressure and each component of the velocity via the geometrical parametrization $\mathbf{G} : \widehat{\Omega} \to \Omega$, that is

$$\begin{aligned} \boldsymbol{\mathcal{V}}^h &= \{ \boldsymbol{v} \,|\, \boldsymbol{v} \circ \mathbf{G} \in \widehat{\boldsymbol{\mathcal{V}}}^h \}, \\ \mathcal{Q}^h &= \{ p \,|\, p \circ \mathbf{G} \in \widehat{\mathcal{Q}}^h \}, \end{aligned} \tag{3.25}$$

giving isoparametric discretization spaces. However, Buffa et al. [13] demonstrate experimentally that the inf-sup stability is preserved only for the Taylor-Hood pair of spaces. For the RT- and N-spaces it is necessary to use a Piola mapping for the velocity space to preserve stability, i.e.,

$$\boldsymbol{\mathcal{V}}^h = \left\{ \frac{\mathbf{J}_\mathbf{G}}{\det(\mathbf{J}_\mathbf{G})} \boldsymbol{v} \,|\, \boldsymbol{v} \circ \mathbf{G} \in \widehat{\boldsymbol{\mathcal{V}}}^h \right\}, \tag{3.26}$$

where $\mathbf{J}_\mathbf{G}$ is the Jacobian matrix of the mapping $\mathbf{G}$.

For the isogeometric SG element, the velocity space is defined on a subgrid of the pressure grid obtained by subdividing each pressure element into 2^d velocity elements in

the sense of k-refinement, where d is the space dimension. This allows to take both pressure and velocity fields with the highest possible continuity, thus, for example, k-degree C^{k-1}-continuous pressure and $(k+1)$-degree C^k-continuous velocity. The isoparametric mapping (3.25) is considered in this case.

Due to simplicity of implementation, we consider only the isogeometric TH element in this work.

Chapter 4

Solution methods

In practical applications, the linear systems arising from discretizations of the linearized incompressible Navier–Stokes equations are usually very large, especially for problems in three dimensions or flows with high Reynolds number, where proper resolution of the flow phenomena (for example boundary layers) requires very fine meshes. Recall that we are dealing with sparse nonsymmetric saddle-point linear systems of the form

$$\begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{4.1}$$

where the block $\mathbf{F}$ depends on the velocity solution from the most recent Picard iteration or time step. Let us denote the matrix of the linear system (4.1) as

$$\boldsymbol{\mathcal{A}} := \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \tag{4.2}$$

for convenience. These systems have to be solved in every Picard iteration for steady problems and at least once in every time step for time-dependent problems. Thus, it is often necessary to solve these large linear systems many times during the solution of a single flow problem, and therefore, it is very important to be able to solve them efficiently. Moreover, we need a robust solver that is able to deal with a wide range of problem and discretization parameters, such as Reynolds number and mesh refinement level. The solver should also perform well for stretched grids, since the meshes needed in the real-world flow computations are often far from uniform. In isogeometric analysis, robustness with respect to the discretization basis degree and continuity is also desirable.

An overview of solution methods for general saddle-point linear systems can be found, e.g., in Benzi et al. [5], where several particular applications including the incompressible flow problems are commented on in more detail. There are two possible approaches to the solution of saddle-point systems: coupled and segregated. Coupled methods deal with the linear system (4.1) as a whole, computing both unknown vectors (velocity and pressure in the case of the incompressible flow) simultaneously. On the other hand, segregated methods compute the velocity and pressure separately. The solution methods for general linear systems can be divided into two main categories: direct and iterative methods. The coupled methods for saddle-point systems can be either direct or iterative. The segregated methods can involve direct or iterative methods or a combination of both as their components.

Direct methods are usually based on a factorization of the system matrix, such as LU factorization. Assuming exact arithmetic, they give the exact solution in a finite

number of steps. Direct methods are robust, however, their main disadvantage is that their time and memory requirements are too high and for large problems, especially in three dimensions, the direct solution can be even unfeasible. Moreover, all steps of the algorithm have to be always performed, since no meaningful approximations of the solution are obtained during the solution process.

On the other hand, iterative methods are based on constructing a sequence of approximations of the exact solution. The iterative process is stopped when a "sufficiently good" solution is obtained, for example if the residual norm is reduced by a given number of orders of magnitude.

A widely used family of methods for solving large linear systems are the Krylov subspace methods. These methods have some common features with both direct and iterative methods. Theoretically, they find the exact solution of the linear system in a finite number of steps, similarly to direct methods. In exact arithmetic, the solution is obtained in at most N iterations, where N is the number of unknowns. At the same time, an approximation of the solution is constructed in each iteration. Even if exact arithmetic was used, if N is large, performing all iterations needed to obtain the exact solution would be impractical. The situation in finite precision arithmetic is even more complicated. Therefore, Krylov subspace methods are used as iterative methods, i.e., the iteration process is stopped as soon as a "sufficiently good" approximation of the solution is obtained.

Another very important class of methods for solving linear systems arising from discretization of partial differential equations that was not mentioned so far are the multilevel methods. This class includes the multigrid and domain decomposition methods which can be both used as coupled solvers for the saddle-point system or as a part of segregated methods or preconditioners for Krylov subspace solvers. We do not describe the details of these methods in this work. For a comprehensive survey of multigrid methods including applications to fluid flow problems, we refer to [101]. An overview of domain decomposition algorithms and also their use as preconditioners for saddle-point problems can be found, e.g., in [100].

This work is devoted to the coupled solution of the linear systems of the form (4.1). In this chapter, we give a brief overview the basic principles of direct and iterative methods with focus on the Krylov subspace methods. Then we mention several methods specifically for saddle-point problems including some segregated methods, since these principles are frequently exploited in the construction of preconditioners for the coupled system.

4.1 Basic principles

In this section, we briefly describe of the principles of direct methods, stationary iterative methods and Krylov subspace methods for a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with a general nonsingular square matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$.

4.1.1 Direct methods

Direct methods are often based on the LU factorization of the system matrix, i.e., on the decomposition $\mathbf{A} = \mathbf{L}\mathbf{U}$, where the factors $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular, respectively. The solution $\mathbf{x}$ can be written as

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{b} = \mathbf{U}^{-1}\mathbf{y}, \tag{4.3}$$

where $\mathbf{y} = \mathbf{L}^{-1}\mathbf{b}$, and $\mathbf{x}$ is computed in two steps. First, the vector $\mathbf{y}$ is obtained as a solution of the system $\mathbf{L}\mathbf{y} = \mathbf{b}$ and then the system $\mathbf{U}\mathbf{x} = \mathbf{y}$ is solved. These are two triangular linear systems that can be solved using forward and back substitution, respectively. Thus, once the LU decomposition of the matrix $\mathbf{A}$ is available, the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is easy to solve.

The basic Gaussian elimination algorithm realizes the solution process described above. The LU decomposition and the vector $\mathbf{y}$ are obtained in the forward elimination step and the system with the upper triangular matrix $\mathbf{U}\mathbf{x} = \mathbf{y}$ is solved in the back substitution step. We note that pivoting (choosing suitable diagonal elements of $\mathbf{U}$ by permutation of the rows and/or columns of $\mathbf{A}$) may be necessary to prevent breakdown caused by division by zero. Further details can be found, e.g., in Golub and Van Loan [54].

In finite precision, only an approximation $\widehat{\mathbf{x}}$ of the exact solution is always computed due to roundoff errors. Pivoting is also helpful to control the roundoff errors. In fact, using a suitable pivoting strategy can guarantee that the residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\widehat{\mathbf{x}}$ will be small in some sense. However, a small residual does not imply a small error of the solution $\mathbf{e} = \mathbf{x} - \widehat{\mathbf{x}}$. After substituting $\widehat{\mathbf{x}} = \mathbf{x} - \mathbf{e}$ into the expression for residual, we obtain the relation between the residual and error $\mathbf{r} = \mathbf{A}\mathbf{e}$ and hence $\mathbf{e} = \mathbf{A}^{-1}\mathbf{r}$. Thus, if the absolute value of the elements in $\mathbf{A}^{-1}$ is large, the error can also be large even for a small residual.

The solution of a linear system with a dense $N \times N$ matrix using Gaussian elimination requires $O(N^3)$ floating point operations (flops) and $O(N^2)$ storage memory. The computational costs of direct methods are often very high also for sparse matrices, since new nonzeros typically arise in the factors $\mathbf{L}$ and $\mathbf{U}$ compared to the matrix $\mathbf{A}$ (fill-in), which can result in $\mathbf{L}, \mathbf{U}$ being much denser than $\mathbf{A}$. The state-of-the-art implementations of direct methods for sparse matrices exploit sophisticated combinatorial algorithms and matrix reordering heuristics to reduce fill-in and thus the time and memory requirements, see, for example [10, 5].

As already mentioned in Section 3.3, the costs of direct methods for linear systems resulting from high continuity IgA discretizations are higher than for linear systems of a similar size in standard FEM or C^0 IgA. Collier et al. presented a study of performance of direct solvers for linear systems resulting from various IgA discretizations of a model three-dimensional Laplace problem in [21]. This topic was further studied in [51], where the use of high continuity IgA with C^0 separators was proposed, which is referred to as refined isogeometric analysis (rIgA). For illustration of the increasing costs, we present computational times (in seconds) of the matrix factorization for various IgA discretizations with maximum continuity in Table 4.1. The linear systems were obtained from discretizations of the Poisson problem described in Section 3.3 of degree $k = 2, \ldots, 7$ with C^{k-1} continuity for a mesh with 256×256 elements. Notice that the size of the matrices does not change significantly. Two different sparse direct solvers based on LU factorization were used: the sparse LU solver available in the Eigen library [55] and the Intel MKL PARDISO solver [91]. All computations were performed on a single thread.

4.1.2 Stationary iterative methods

Iterative methods are a more economical choice for large sparse linear systems than direct methods, especially in terms of computer memory. Stationary iterative methods

discr.	N	Eigen	Pardiso
$k = 2, C^1$	65536	6.6	1.0
$k = 3, C^2$	66049	23.4	2.4
$k = 4, C^3$	66564	52.1	4.2
$k = 5, C^4$	67081	83.2	7.2
$k = 6, C^5$	67600	201.6	10.9
$k = 7, C^6$	68121	541.1	16.2

Table 4.1: Computational times (in seconds) of the factorization of the matrices obtained from various IgA discretizations of the Poisson problem.

form a class of simple methods based on an iteration formula of the form

$$\mathbf{x}_{n+1} = \mathbf{H}\mathbf{x}_n + \mathbf{c}, \quad n = 1, 2, \ldots, \tag{4.4}$$

starting from an initial guess $\mathbf{x}_0$. The reason why they are called stationary is that the iteration matrix $\mathbf{H}$ and the vector $\mathbf{c}$ do not change during the iteration process.

The method is called consistent if the relation (4.4) holds for the exact solution $\mathbf{x}$ of the linear system, i.e., $\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{c}$. Then, the error $\mathbf{e}_n$ of the solution generated in the n-th iteration of a consistent stationary iterative method can be expressed as

$$\mathbf{e}_n = \mathbf{x} - \mathbf{x}_n = \mathbf{H}\mathbf{x} + \mathbf{c} - (\mathbf{H}\mathbf{x}_{n-1} + \mathbf{c}) = \mathbf{H}\mathbf{e}_{n-1} = \mathbf{H}^n\mathbf{e}_0 \tag{4.5}$$

and thus

$$||\mathbf{e}_n|| \leq ||\mathbf{H}^n||\,||\mathbf{e}_0||. \tag{4.6}$$

It is a classical result that such method converges to the exact solution of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ for any initial vector $\mathbf{x}_0$ if and only if $\rho(\mathbf{H}) < 1$, where $\rho(\mathbf{H})$ is the spectral radius of the iteration matrix. See, for example, [54, 87] for more details.

If the method is consistent, the iteration formula (4.4) can be rewritten as

$$\mathbf{M}\mathbf{x}_{n+1} = \mathbf{N}\mathbf{x}_n + \mathbf{b} \tag{4.7}$$

or also in the form

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_n), \tag{4.8}$$

where $\mathbf{M} = \mathbf{A}(\mathbf{I} - \mathbf{H})^{-1}, \mathbf{N} = \mathbf{M}\mathbf{H}$ and hence $\mathbf{A} = \mathbf{M} - \mathbf{N}$. Thus, every consistent stationary iterative method corresponds to a splitting of the matrix $\mathbf{A}$. In order to obtain a potentially efficient method, the splitting has to be chosen such that linear systems with the matrix $\mathbf{M}$ are relatively easy to solve. If $\mathbf{A}$ is a so-called M-matrix, the iteration process given by (4.7) is guaranteed to be convergent for certain splittings called regular, which can be found relatively easily. For more details, see [87, 115].

The convergence of stationary iterative methods is often very slow and therefore they are rarely used as standalone solvers nowadays. They are much more often used as smoothers for multigrid methods or preconditioners for Krylov subspace methods that will be described in the following section.

4.1.3 Krylov subspace methods

Individual Krylov subspace methods can be described from several points of view. To explain the idea of this class of methods, we begin the exposition by introducing general

projection methods and we describe the Krylov subspace methods as their special case as, for example, in the books by Saad [87] and Liesen, Strakoš [71]. Then we state the classical algorithms of two widely used Krylov subspace methods, the probably best known conjugate gradient (CG) method and GMRES (generalized minimal residual) method, which is used in all numerical experiments in this work. We refer to [87, 71] for detailed derivations and proofs of the statements mentioned in this section.

Projection methods

Given an initial vector $\mathbf{x}_0$, we generate a sequence of approximations $\mathbf{x}_n$ to the solution of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ in the form

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{z}_n, \quad \mathbf{z}_n \in \mathcal{S}_n, \tag{4.9}$$

for $n = 1, 2, \ldots$, where $\mathcal{S}_n$ is an n-dimensional subspace of $\mathbb{R}^N$ called search space. For the n-th residual, it holds

$$\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n = \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{z}_n) = \mathbf{r}_0 - \mathbf{A}\mathbf{z}_n, \tag{4.10}$$

i.e. $\mathbf{r}_n \in \mathbf{r}_0 + \mathbf{A}\mathcal{S}_n$, where $\mathbf{r}_0$ is the initial residual for the vector $\mathbf{x}_0$.

Since the search space $\mathcal{S}_n$ has dimension n, we need n constraints to determine $\mathbf{z}_n$ (and hence $\mathbf{x}_n$). These constraints are imposed on the residual. We define another n-dimensional subspace $\mathcal{C}_n \subset \mathbb{R}^N$ called constraint space and require the n-th residual vector $\mathbf{r}_n$ to be orthogonal to that space, i.e.

$$\mathbf{r}_n \perp \mathcal{C}_n \quad \text{ or equivalentnly } \quad \mathbf{r}_n \in \mathcal{C}_n^{\perp}. \tag{4.11}$$

If the spaces $\mathcal{S}_n$ and $\mathcal{C}_n$ are chosen such that $\mathbb{R}^N$ is a direct sum of $\mathbf{A}\mathcal{S}_n$ and $\mathcal{C}_n^{\perp}$,

$$\mathbb{R}^N = \mathbf{A}\mathcal{S}_n \oplus \mathcal{C}_n^{\perp}, \tag{4.12}$$

then the vectors $\mathbf{A}\mathbf{z}_n$ and $\mathbf{r}_n$ are uniquely determined for any $\mathbf{r}_0 \in \mathbb{R}^N$, which can be written as

$$\begin{aligned} \mathbf{r}_0 &= \mathbf{r}_0\big|_{\mathbf{A}\mathcal{S}_n} + \mathbf{r}_0\big|_{\mathcal{C}_n^{\perp}}, \\ &= \mathbf{A}\mathbf{z}_n + \mathbf{r}_n. \end{aligned} \tag{4.13}$$

Thus, the vector $\mathbf{A}\mathbf{z}_n$ represents a projection of the initial residual $\mathbf{r}_0$ onto the space $\mathbf{A}\mathcal{S}_n$ and orthogonal to $\mathcal{C}_n$ with the complement given by $\mathbf{r}_n$. Moreover, if $\mathbf{A}\mathcal{S}_n = \mathcal{C}_n$, the decomposition (4.13) is orthogonal.

Consider arbitrary bases of the search and constraint space and two matrices $\mathbf{S}_n, \mathbf{C}_n \in \mathbb{R}^{N\times n}$ whose columns are formed by the basis vectors of $\mathcal{S}_n$ and $\mathcal{C}_n$, respectively. Then, the n-th approximation $\mathbf{x}_n$ can be written in the form

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{S}_n\mathbf{t}_n, \tag{4.14}$$

where $\mathbf{t}_n \in \mathbb{R}^n$ is a vector of unknown coefficients of $\mathbf{z}_n$ in the basis represented by $\mathbf{S}_n$. The orthogonality constraints (4.11) are applied to determine $\mathbf{t}_n$, i.e., we require

$$\mathbf{0} = \mathbf{C}_n^T\mathbf{r}_n = \mathbf{C}_n^T(\mathbf{b} - \mathbf{A}\mathbf{x}_n) = \mathbf{C}_n^T\mathbf{r}_0 - \mathbf{C}_n^T\mathbf{A}\mathbf{S}_n\mathbf{t}_n. \tag{4.15}$$

Thus, instead of solving the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ of order N, we solve a reduced system of order n

$$\mathbf{C}_n^T \mathbf{A}\mathbf{S}_n \mathbf{t}_n = \mathbf{C}_n^T \mathbf{r}_0 \tag{4.16}$$

in the n-th iteration of a projection method, which is referred to as the projected system. The idea of projection methods is to obtain a good approximation $\mathbf{x}_n \approx \mathbf{x}$ for some $n \ll N$. The projection process is said to be well defined at step n if the projected matrix $\mathbf{C}_n^T \mathbf{A}\mathbf{S}_n$ is nonsingular. This is true if and only if (4.12) holds.

There are some conditions on the matrix $\mathbf{A}$ and the choice of the search and constraint space for which the projection process is guaranteed to be well defined at step n for any bases $\mathbf{S}_n, \mathbf{C}_n$. Two important cases are:

(i) $\mathbf{A}$ is symmetric positive definite and $\mathcal{C}_n = \mathcal{S}_n$,

(ii) $\mathbf{A}$ is nonsingular and $\mathcal{C}_n = \mathbf{A}\mathcal{S}_n$.

A projection process is said to have a finite termination property if the exact solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is obtained in a finite number of steps. Assume that a projection process is well defined at step n and that $\mathbf{x}_n$ is the exact solution, i.e. $\mathbf{A}\mathbf{x}_n = \mathbf{b}$. Then we have $\mathbf{r}_n = \mathbf{0}$ and according to (4.13), this can be true if and only if $\mathbf{r}_0 = \mathbf{r}_0\big|_{\mathbf{A}\mathcal{S}_n}$, which means that $\mathbf{r}_0 \in \mathbf{A}\mathcal{S}_n$. A sufficient condition for this is that $\mathbf{r}_0 \in \mathcal{S}_n$ and $\mathcal{S}_n = \mathbf{A}\mathcal{S}_n$. Therefore, it is beneficial to build a sequence of nested search spaces such that $\mathcal{S}_1 = \text{span}\{\mathbf{r}_0\}$, $\mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{S}_3 \subset \ldots$ and $\mathbf{A}\mathcal{S}_n = \mathcal{S}_n$ for some n.

Krylov subspaces

In this paragraph, we introduce Krylov subspaces that are a suitable choice of search spaces such that the resulting projection method has the finite termination property. For a given matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ and a nonzero vector $\mathbf{v} \in \mathbb{R}^N$, the Krylov subspace $\mathcal{K}_n(\mathbf{A}, \mathbf{v})$ is defined as

$$\mathcal{K}_n(\mathbf{A}, \mathbf{v}) = \text{span}\{\mathbf{v}, \mathbf{A}\mathbf{v}, \mathbf{A}^2\mathbf{v}, \ldots, \mathbf{A}^{n-1}\mathbf{v}\}. \tag{4.17}$$

For $n = 1, 2, \ldots$, the sequence of Krylov subspaces $\mathcal{K}_n(\mathbf{A}, \mathbf{v})$ is obviously nested.

By definition, $\mathcal{K}_n(\mathbf{A}, \mathbf{v})$ is a subspace of $\mathbb{R}^N$ containing all vectors $\mathbf{x}$ that can be written in the form $\mathbf{x} = p(\mathbf{A})\mathbf{v}$, where p is a polynomial of degree at most $n-1$. The monic polynomial of the lowest degree for which $p(\mathbf{A})\mathbf{v} = \mathbf{0}$ is called the minimal polynomial of $\mathbf{v}$ with respect to $\mathbf{A}$ and its degree $d = d(\mathbf{A}, \mathbf{v})$ is called the grade of $\mathbf{v}$ with respect to $\mathbf{A}$. It means that d is the lowest integer such that

$$\mathbf{A}^d\mathbf{v} + \alpha_{d-1}\mathbf{A}^{d-1}\mathbf{v} + \ldots + \alpha_1\mathbf{A}\mathbf{v} + \alpha_0\mathbf{v} = \mathbf{0} \tag{4.18}$$

with some $\alpha_i \in \mathbb{R}, i = 0, \ldots, d-1$. In other words, $\mathbf{A}^d\mathbf{v}$ is a linear combination of the vectors $\mathbf{v}, \mathbf{A}\mathbf{v}, \ldots, \mathbf{A}^{d-1}\mathbf{v}$ that are linearly independent. Thus, the dimension of $\mathcal{K}_n(\mathbf{A}, \mathbf{v})$ is equal to n for $n = 1, \ldots, d$, and $\mathcal{K}_{d+j}(\mathbf{A}, \mathbf{v}) = \mathcal{K}_d(\mathbf{A}, \mathbf{v})$ for all $j \geq 0$. Moreover, if $\mathbf{A}$ is nonsingular, then $\mathbf{A}\mathcal{K}_d(\mathbf{A}, \mathbf{v}) = \mathcal{K}_d(\mathbf{A}, \mathbf{v})$, i.e. the subspace becomes $\mathbf{A}$-invariant.

The properties described above make Krylov subspaces suitable search spaces for projection methods. In particular, if $\mathcal{S}_n = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ for $n = 1, 2, \ldots$, then $\mathbf{r}_0 \in \mathcal{S}_1 \subset \mathcal{S}_2 \subset \ldots \subset \mathcal{S}_d$ and $\mathbf{A}\mathcal{S}_d = \mathcal{S}_d$, where d is the grade of $\mathbf{r}_0$ with respect to $\mathbf{A}$. If the projection process with such choice of search spaces is well defined at step d, then $\mathbf{r}_d = 0$, i.e. the exact solution is obtained in $d \leq N$ steps. Projection methods with

$\mathcal{S}_n = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ are called Krylov subspace methods. Different choices of constraint spaces $\mathcal{C}_n$ yield different Krylov subspace methods.

Let us mention another property which will be important later in the text. The minimal polynomial of a matrix $\mathbf{A}$ is the monic polynomial of the lowest degree such that $p(\mathbf{A}) = \mathbf{0}$, denote it as $p_{\min}(\mathbf{A})$ and its degree as $D = D(\mathbf{A})$. Then it holds $d \leq D$ for any vector $\mathbf{v}$, since $p_{\min}(\mathbf{A})\mathbf{v}$ is always equal to zero. Thus, a Krylov subspace method finds the exact solution in at most D steps for any initial residual $\mathbf{r}_0$.

Although the individual Krylov subspace methods are mathematically completely determined by the choice of $\mathcal{S}_n$ and $\mathcal{C}_n$, their numerical behavior depends on the choice of the bases of these spaces. Recalling the power method for finding the dominant eigenvalue of a matrix, we know that the vectors of the Krylov sequence $\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots$ converge to an eigenvector of $\mathbf{A}$ corresponding to the dominant eigenvalue. It means that they eventually become very close which can lead to loss of linear independence in finite precision. Therefore, orthogonal bases are used in practice. Practical implementations of particular Krylov subspace methods rely on variants of the Gram–Schmidt orthogonalization algorithm for generating an orthonormal basis of the Krylov subspace: the Arnoldi and Lanczos algorithms.

In the rest of this section, we describe the mentioned orthogonalization algorithms and two Krylov subspace methods based on the conditions (i) and (ii) stated above: the conjugate gradient (CG) method, and the GMRES (generalized minimal residual) method.

Arnoldi algorithm

Consider a Krylov subspace $\mathcal{K}_n(\mathbf{A}, \mathbf{v})$ and assume that $n \leq d$, where $d = d(\mathbf{A}, \mathbf{v})$ is the grade of $\mathbf{v}$ with respect to $\mathbf{A}$. Denote as $\mathbf{v}_1, \ldots, \mathbf{v}_n$ the orthonormal basis of $\mathcal{K}_n(\mathbf{A}, \mathbf{v})$ generated by the Arnoldi algorithm. The algorithm proceeds as follows: the initial vector $\mathbf{v}_1$ is chosen such that $\mathbf{v}_1 = \mathbf{v}/||\mathbf{v}||$, where $||.||$ denotes the classical Euclidean vector norm. At the n-th step, the previously generated Arnoldi vector $\mathbf{v}_n$ is multiplied by the matrix $\mathbf{A}$ and the obtained vector is orthonormalized against all previous Arnoldi vectors $\mathbf{v}_i$, $i = 1, \ldots, n$, using the Gram–Schmidt procedure. The individual steps are summarized in Algorithm 1, where the modified implementation of the Gram–Schmidt algorithm is used.

Algorithm 1: Arnoldi algorithm

Input: matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, vector $\mathbf{v} \in \mathbb{R}^N$.
Output: orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ with $\text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_n\} = \mathcal{K}_n(\mathbf{A}, \mathbf{v})$ for $n = 1, \ldots, d$.

```
Set v_1 = v/||v||.
for n = 1, 2, ... do
    w_n = A v_n
    for i = 1, 2, ..., n do
        h_{i,n} = w_n^T v_i
        w_n = w_n − h_{i,n} v_i
    h_{n+1,n} = ||w_n||
    If h_{n+1,n} = 0, then stop.
    v_{n+1} = w_n / h_{n+1,n}
```

It can be shown that the Arnoldi algorithm stops at step 8 ($h_{n+1,n} = 0$) if and only if $n = d$, and thus, the Krylov subspace spanned by the generated vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is $\mathbf{A}$-invariant.

Denote by $\mathbf{V}_n$ the $N \times n$ matrix with column vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Further, denote by $\underline{\mathbf{H}}_n$ the $(n+1) \times n$ matrix whose nonzero entries $h_{i,j}$ are defined in Algorithm 1 and by $\mathbf{H}_n$ the matrix obtained from $\underline{\mathbf{H}}_n$ by omitting the last row, i.e.,

$$\mathbf{H}_n = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,n} \\ & \ddots & \ddots & \vdots \\ & & h_{n,n-1} & h_{n,n} \end{bmatrix}. \tag{4.19}$$

The matrix $\mathbf{H}_n$ is in an unreduced upper Hessenberg form, which means that all entries below the first subdiagonal are zero, while no entry of the first subdiagonal vanishes. After n steps of the algorithm, it holds

$$\begin{aligned} \mathbf{A}\mathbf{V}_n &= \mathbf{V}_n\mathbf{H}_n + h_{n+1,n}\mathbf{v}_{n+1}\vec{e}_n^T \\ &= \mathbf{V}_{n+1}\underline{\mathbf{H}}_n, \end{aligned} \tag{4.20}$$

where $\vec{e}_n$ denotes the n-th column of the $n \times n$ identity matrix $\mathbf{I}_n$. Further, since $\mathbf{V}_n^T\mathbf{V}_n = \mathbf{I}_n$ and $\mathbf{V}_n^T\mathbf{v}_{n+1} = \mathbf{0}$, we have

$$\mathbf{V}_n^T\mathbf{A}\mathbf{V}_n = \mathbf{H}_n. \tag{4.21}$$

Symmetric Lanzos algorithm

The symmetric Lanczos algorithm can be viewed as a simplification of the Arnoldi algorithm with a symmetric matrix $\mathbf{A}$. If $\mathbf{A}$ is symmetric, then, since (4.21) holds, the upper Hessenberg matrix $\mathbf{H}_n$ has to be also symmetric and thus it is a tridiagonal matrix. It is usually denoted by $\mathbf{T}_n$ and written in the form

$$\mathbf{T}_n = \begin{bmatrix} \gamma_1 & \delta_2 & & & \\ \delta_2 & \gamma_2 & \delta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{n-1} & \gamma_{n-1} & \delta_n \\ & & & \delta_n & \gamma_n \end{bmatrix}, \tag{4.22}$$

where $\gamma_i = h_{i,i}$ for $i = 1, \ldots, n$, and $\delta_i = h_{i-1,i}$ for $i = 2, \ldots, n$. This leads to Algorithm 2 which is the standard implementation of the symmetric Lanczos algorithm corresponding to the modified Gram–Schmidt procedure.

The most important difference from the Arnoldi algorithm with a general nonsymmetric matrix is that, in the n-th step, the vector $\mathbf{A}\mathbf{v}_n$ is orthonormalized only against the two preceding vectors using the three-term recurrence relation

$$\delta_{n+1}\mathbf{v}_{n+1} = \mathbf{A}\mathbf{v}_n - \gamma_n\mathbf{v}_n - \delta_n\mathbf{v}_{n-1}. \tag{4.23}$$

This leads to constant work and memory requirements per iteration. On the contrary, the work and memory requirements of the Arnoldi algorithm with a nonsymmetric matrix increase in each iteration.

Algorithm 2: Symmetric Lanczos algorithm

Input: symmetric matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, vector $\mathbf{v} \in \mathbb{R}^N$.
Output: orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ with $\text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_n\} = \mathcal{K}_n(\mathbf{A}, \mathbf{v})$ for $n = 1, \ldots, d$.

Set $\mathbf{v}_0 = \mathbf{0}$, $\delta_1 = 0$, $\mathbf{v}_1 = \mathbf{v}/||\mathbf{v}||$.
for $n = 1, 2, \ldots$ **do**
  $\mathbf{w}_n = \mathbf{A}\mathbf{v}_n - \delta_n \mathbf{v}_{n-1}$
  $\gamma_n = \mathbf{v}_n^T \mathbf{w}_n$
  $\mathbf{w}_n = \mathbf{w}_n - \gamma_n \mathbf{v}_n$
  $\delta_{n+1} = ||\mathbf{w}_n||$
  If $\delta_{n+1} = 0$, then stop.
  $\mathbf{v}_{n+1} = \mathbf{w}_n/\delta_{n+1}$

Conjugate gradient method

The conjugate gradient method, proposed by Hestenes and Stiefel in [59], is characterized by the condition (i), i.e., it can be used for linear systems with a symmetric positive definite (SPD) matrix and the search and constraint spaces are chosen such that $\mathcal{S}_n = \mathcal{C}_n = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$. Thus, the orthogonality condition (4.11) becomes $\mathbf{r}_n \perp \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$, or in other words

$$\mathbf{v}^T \mathbf{r}_n = 0, \quad \forall \mathbf{v} \in \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0). \tag{4.24}$$

Since the n-th residual can be written as

$$\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}_n = \mathbf{A}(\mathbf{x} - \mathbf{x}_n) = \mathbf{A}\mathbf{e}_n, \tag{4.25}$$

where $\mathbf{e}_n$ is the error of the approximation in the n-th iteration of the method, it holds

$$\mathbf{v}^T \mathbf{A}\mathbf{e}_n = 0, \quad \forall \mathbf{v} \in \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0), \tag{4.26}$$

i.e., the n-th error is $\mathbf{A}$-orthogonal to the n-th Krylov subspace, $\mathbf{e}_n \perp_{\mathbf{A}} \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$. This orthogonality condition is equivalent to the optimality property

$$||\mathbf{e}_n||_{\mathbf{A}} = ||\mathbf{x} - \mathbf{x}_n||_{\mathbf{A}} = \min_{\mathbf{y} \in \mathbf{x}_0 + \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)} ||\mathbf{x} - \mathbf{y}||_{\mathbf{A}}, \tag{4.27}$$

which means that the solution found in the n-th iteration of the CG method is the vector in the space $\mathbf{x}_0 + \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ which is closest to the exact solution $\mathbf{x}$ in the $\mathbf{A}$-norm $||.||_{\mathbf{A}}$, where $||\mathbf{v}||_{\mathbf{A}} = \sqrt{\mathbf{v}^T \mathbf{A}\mathbf{v}}$.

Since the $\mathbf{A}$-norm of error is minimized over nested spaces of growing dimension during the CG iterations, the sequence of $||\mathbf{e}_n||_{\mathbf{A}}$ is obviously non-increasing (in fact, it is strictly decreasing). The most widely known error bound for the CG method is

$$\frac{||\mathbf{e}_n||_{\mathbf{A}}}{||\mathbf{e}_0||_{\mathbf{A}}} \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^n, \quad n = 1, \ldots, d, \tag{4.28}$$

where $\kappa(\mathbf{A}) = \lambda_{max}/\lambda_{min}$ is the condition number of the matrix $\mathbf{A}$ and $\lambda_{max}, \lambda_{min}$ are the maximal and minimal eigenvalue of $\mathbf{A}$, respectively. However, this bound is too pessimistic and often not descriptive even for the worst-case convergence of CG. It is possible to derive a bound that describes the worst-case behavior of the method for a

given matrix $\mathbf{A}$ based on all eigenvalues of $\mathbf{A}$, see [71] for its derivation and more insight into the convergence of the CG method.

The implementation of the CG method is based on the symmetric Lanczos algorithm for generating an orthonormal basis of $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$. In this case, the projected system at step n (4.16) becomes

$$\mathbf{V}_n^T \mathbf{A} \mathbf{V}_n \mathbf{t}_n = \mathbf{V}_n^T \mathbf{r}_0. \tag{4.29}$$

Since (4.21) holds with $\mathbf{T}_n$ on the right-hand side for the symmetric Lanczos method and $\mathbf{r}_0 = ||\mathbf{r}_0||\mathbf{v}_1$, the system (4.29) takes the form

$$\mathbf{T}_n \mathbf{t}_n = ||\mathbf{r}_0|| \vec{e}_1. \tag{4.30}$$

The individual steps of one iteration of the CG method are based on solving this linear system using an LDLT decomposition of $\mathbf{T}_n$ and determining the new approximation $\mathbf{x}_n$ from (4.14), i.e., $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n \mathbf{t}_n$. The resulting algorithm is summarized in Algorithm 3. The residuals $\mathbf{r}_0, \ldots, \mathbf{r}_n$ are scalar multiples of the Lanczos vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}$ and form

Algorithm 3: Conjugate gradient (CG) method

Input: SPD matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, right-hand side $\mathbf{b} \in \mathbb{R}^N$, initial approximation $\mathbf{x}_0 \in \mathbb{R}^N$, stopping tolerance ε.

Output: approximation $\mathbf{x}_n$.

1 Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{r}_0$.

2 **for** $n = 1, 2, \ldots$ **do**

3 $\quad \alpha_{n-1} = \frac{||\mathbf{r}_{n-1}||^2}{\mathbf{p}_{n-1}^T \mathbf{A} \mathbf{p}_{n-1}}$

4 $\quad \mathbf{x}_n = \mathbf{x}_{n-1} + \alpha_{n-1} \mathbf{p}_{n-1}$

5 $\quad \mathbf{r}_n = \mathbf{r}_{n-1} - \alpha_{n-1} \mathbf{A} \mathbf{p}_{n-1}$

6 $\quad$ If the stopping criterion is satisfied (e.g., $||\mathbf{r}_n||/||\mathbf{r}_0|| < \varepsilon$), stop the iteration.

7 $\quad \beta_n = \frac{||\mathbf{r}_n||^2}{||\mathbf{r}_{n-1}||^2}$

8 $\quad \mathbf{p}_n = \mathbf{r}_n + \beta_n \mathbf{p}_{n-1}$

an orthogonal basis of the Krylov subspace $\mathcal{K}_{n+1}(\mathbf{A}, \mathbf{r}_0)$. The vectors $\mathbf{p}_0, \ldots, \mathbf{p}_n$ form an $\mathbf{A}$-orthogonal (or conjugate) basis of $\mathcal{K}_{n+1}(\mathbf{A}, \mathbf{r}_0)$. They are called direction vectors, since they represent a search direction in which a new approximation to the solution is searched starting from the previous one. In the classical implementation of CG, the vectors $\mathbf{r}_n$ and $\mathbf{p}_n$ are generated using two coupled two-term recurrences. It is also possible to reformulate the algorithm such that two decoupled three-term recurrences are used, however, such implementations tend to be more sensitive to rounding errors [71].

GMRES method

The GMRES method, proposed by Saad and Schultz in [88], is characterized by the condition (ii), i.e., it can be used for linear systems with a general nonsingular matrix and $\mathcal{S}_n = \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ and $\mathcal{C}_n = \mathbf{A}\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$. The orthogonality condition (4.11) yields

$$\mathbf{w}^T \mathbf{r}_n = 0, \quad \forall \mathbf{w} \in \mathbf{A}\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0) \tag{4.31}$$

or equivalently

$$\mathbf{v}^T \mathbf{A}^T \mathbf{r}_n = 0, \quad \forall \mathbf{v} \in \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0). \tag{4.32}$$

Thus, the error $\mathbf{e}_n$ satisfies

$$\mathbf{v}^T\mathbf{A}^T\mathbf{A}\mathbf{e}_n = 0, \quad \forall \mathbf{v} \in \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0), \tag{4.33}$$

that is, the n-th error is $\mathbf{A}^T\mathbf{A}$-orthogonal to the n-th Krylov subspace, $\mathbf{e}_n \perp_{\mathbf{A}^T\mathbf{A}} \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$. These orthogonality conditions are equivalent to the optimality properties

$$||\mathbf{r}_n|| = \min_{\mathbf{y} \in \mathbf{x}_0 + \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)} ||\mathbf{b} - \mathbf{A}\mathbf{y}|| \tag{4.34}$$

and

$$||\mathbf{e}_n||_{\mathbf{A}^T\mathbf{A}} = ||\mathbf{x} - \mathbf{x}_n||_{\mathbf{A}^T\mathbf{A}} = \min_{\mathbf{y} \in \mathbf{x}_0 + \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)} ||\mathbf{x} - \mathbf{y}||_{\mathbf{A}^T\mathbf{A}}, \tag{4.35}$$

which means that the solution found in the n-th iteration of the GMRES method is the vector in the space $\mathbf{x}_0 + \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ with the smallest residual measured in the Euclidean norm and the smallest error measured in the $\mathbf{A}^T\mathbf{A}$-norm.

Again, the sequence $||\mathbf{r}_n||$ (and also $||\mathbf{e}_n||_{\mathbf{A}^T\mathbf{A}}$) is obviously non-increasing. Unlike CG, the convergence of GMRES is not that well understood. An important difference from CG is that it is generally not possible to derive a descriptive convergence bound based only on eigenvalues of the matrix $\mathbf{A}$.

The implementation of the GMRES method is based on the Arnoldi algorithm for generating an orthonormal basis of $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$. The projected system (4.16) takes the form

$$\mathbf{V}_n^T\mathbf{A}^T\mathbf{A}\mathbf{V}_n\mathbf{t}_n = \mathbf{V}_n^T\mathbf{A}^T\mathbf{r}_0, \tag{4.36}$$

which can be rewritten as

$$\underline{\mathbf{H}}_n^T\underline{\mathbf{H}}_n\mathbf{t}_n = \underline{\mathbf{H}}_n^T||\mathbf{r}_0||\vec{e}_1 \tag{4.37}$$

using the relation (4.20). Thus, the new approximation $\mathbf{x}_n$ is determined using (4.14) with $\mathbf{t}_n = \left(\underline{\mathbf{H}}_n^T\underline{\mathbf{H}}_n\right)^{-1}\underline{\mathbf{H}}_n^T||\mathbf{r}_0||\vec{e}_1$.

The algorithm can be also derived directly from the optimality property (4.34), where $\mathbf{y}$ is written as $\mathbf{y} = \mathbf{x}_0 + \mathbf{V}_n\mathbf{t}$ and the optimality property takes the form

$$\begin{aligned} ||\mathbf{r}_n|| &= \min_{\mathbf{t} \in \mathbb{R}^n} ||\mathbf{r}_0 - \mathbf{A}\mathbf{V}_n\mathbf{t}|| \\ &= \min_{\mathbf{t} \in \mathbb{R}^n} || \, ||\mathbf{r}_0||\mathbf{v}_1 - \mathbf{V}_{n+1}\underline{\mathbf{H}}_n\mathbf{t}|| \\ &= \min_{\mathbf{t} \in \mathbb{R}^n} ||\mathbf{V}_{n+1}(||\mathbf{r}_0||\vec{e}_1 - \underline{\mathbf{H}}_n\mathbf{t})|| \\ &= \min_{\mathbf{t} \in \mathbb{R}^n} || \, ||\mathbf{r}_0||\vec{e}_1 - \underline{\mathbf{H}}_n\mathbf{t}||, \end{aligned} \tag{4.38}$$

where the fact that the columns of $\mathbf{V}_{n+1}$ are orthonormal has been used in the last step. This is a least-squares minimization problem with a unique solution $\mathbf{t} = \mathbf{t}_n$ given as the solution of the corresponding normal equations (4.37). In the practical implementation of GMRES, this least-squares problem is solved using a QR factorization of the matrix $\underline{\mathbf{H}}_n$. The QR factorization is obtained by a sequence of Givens rotations. Once the factors are computed, the residual norm $||\mathbf{r}_n||$ can be evaluated easily. The individual steps of the GMRES algorithm are summarized in Algorithm 4.

Algorithm 4: GMRES method

Input: nonsingular matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, right-hand side $\mathbf{b} \in \mathbb{R}^N$, initial approximation $\mathbf{x}_0 \in \mathbb{R}^N$, stopping tolerance ε.

Output: approximation $\mathbf{x}_n$.

1 Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_1 = \mathbf{r}_0/||\mathbf{r}_0||$.

2 **for** $n = 1, 2, \ldots$ **do**

3 Compute the n-th step of the Arnoldi algoritnm for $\mathbf{A}$ and $\mathbf{v} = \mathbf{r}_0$.

4 Update the QR factorization of $\underline{\mathbf{H}}_n$.

5 If the stopping criterion is satisfied (e.g., $||\mathbf{r}_n||/||\mathbf{r}_0|| < \varepsilon$), stop the iteration and go to 6.

6 Compute $\mathbf{t}_n$ as the minimizer of $||\,||\mathbf{r}_0||\vec{e}_1 - \underline{\mathbf{H}}_n t||$ and $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n\mathbf{t}_n$.

Other Krylov subspace methods

The CG method is the method of choice whenever we deal with a linear system with a symmetric positive definite matrix. In the case where the matrix is symmetric indefinite, the minimal residual (MINRES) method or the symmetric LQ (SYMMLQ) method proposed in [80] can be used.

For linear systems with a general nonsymmetric matrix, there is quite a wide variety of Krylov subspace methods that can be applied besides GMRES. We mention, for example, the generalized conjugate residual (GCR) method [32] or the orthogonal direction (ORTHODIR) method [116] that are mathematically equivalent to GMRES and also use long recurrences, i.e., their computational costs and memory requirements increase during the iteration process. There are some Krylov subspace methods that overcome this difficulty, such as the biconjugate gradient (BiCG) method [47] and its variant BiCGstab [109]. The implementation of these methods is based on a nonsymmetric Lanczos algorithm (also called Lanczos biorthogonalization or two-sided Lanczos algorithm) which generates two sets of biorthogonal basis vectors instead of one orthogonal basis. This leads to three-term recurrence relations and thus constant work and memory per iteration. However, these method suffer from serious breakdowns, which means that they can stop although no invariant subspace has been computed.

The memory requirements of GMRES or equivalent methods can be also reduced by using their restarted variants, e.g., GMRES(k) where the GMRES algorithm is restarted every k iterations. On the other hand, the convergence of the restarted versions can be slow or they can stagnate completely.

There are many other Krylov subspace methods that were not mentioned here, see, for example, [89, 97].

4.2 Solution of saddle-point systems

All solution techniques described in the previous section can be used as coupled solvers for the saddle-point system (4.1). Direct solvers can be applied in a black-box manner. There is no specialized direct solver for nonsymmetric saddle-point systems, nevertheless, we refer to [5] for a discussion of coupled direct solvers for symmetric saddle-point linear systems.

Krylov subspace methods for nonsymmetric linear systems can be also used in a straightforward way. However, their convergence is strongly dependent on the precon-

ditioning. As already mentioned, the preconditioning techniques for saddle-point linear systems often exploit the segregated approach or stationary iterative methods. Several preconditioners suitable for linear systems obtained from discretization of the Navier–Stokes equations will be described in Chapter 5.

In this section, we briefly mention one of the segregated approaches to the solution of (4.1) called Schur complement reduction and then we describe two examples of stationary iterative methods for saddle-point linear systems.

4.2.1 Schur complement reduction

We consider the linear system (4.1) and write it in the form

$$\begin{aligned} \mathbf{F}\mathbf{u} + \mathbf{B}^T\mathbf{p} &= \mathbf{f}, \\ \mathbf{B}\mathbf{u} &= \mathbf{g}. \end{aligned} \tag{4.39}$$

Assuming that $\mathbf{F}$ is nonsingular, if we multiply the first equation of (4.39) by the matrix $\mathbf{B}\mathbf{F}^{-1}$ and subtract it from the second equation, we obtain a reduced linear system for $\mathbf{p}$,

$$-\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T\mathbf{p} = \mathbf{g} - \mathbf{B}\mathbf{F}^{-1}\mathbf{f}, \tag{4.40}$$

where the matrix $\mathbf{S} := -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T$ is the Schur complement of $\mathbf{F}$ in $\boldsymbol{\mathcal{A}}$. If we also assume that the whole system matrix $\boldsymbol{\mathcal{A}}$ is nonsingular, it can be easily shown that $\mathbf{S}$ is also nonsingular, see [5]. Thus, the pressure solution $\mathbf{p}$ can be computed from (4.40) and the velocity solution $\mathbf{u}$ is then obtained from another reduced system

$$\mathbf{F}\mathbf{u} = \mathbf{f} - \mathbf{B}^T\mathbf{p}. \tag{4.41}$$

This is actually a block Gaussian elimination applied to (4.1) which leads to a block LU decomposition of the matrix $\boldsymbol{\mathcal{A}}$,

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{U}} = \begin{bmatrix} \mathbf{I}_u & \mathbf{0} \\ \mathbf{B}\mathbf{F}^{-1} & \mathbf{I}_p \end{bmatrix} \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{S} \end{bmatrix}, \tag{4.42}$$

where $\mathbf{I}_u$ and $\mathbf{I}_p$ are identity matrices of size $(d \cdot n_u) \times (d \cdot n_u)$ and $n_p \times n_p$, respectively. The two reduced systems (4.40) and (4.41) correspond to the block back substitution, i.e. solving the linear system with the matrix $\boldsymbol{\mathcal{U}}$. We note that forming the right-hand side of (4.40) requires solving of a linear system with the matrix $\mathbf{F}$ (if the vector $\mathbf{f}$ is nonzero), and thus the Schur complement reduction method requires solving one linear system with the matrix $\mathbf{S}$ and two linear systems with the matrix $\mathbf{F}$. These systems can be solved directly or iteratively. However, in the case of linear systems arising from discretizations of the Stokes and Navier–Stokes equations, the Schur complement is typically dense and very expensive to construct. Therefore, this method is not of practical interest for our problem, but it forms a basis for some of the most effective preconditioners for the Navier–Stokes equations.

4.2.2 Uzawa method

The Uzawa method is one of the first iterative schemes for saddle-point systems proposed in [108] in the context of constrained optimization. It became popular also in the field

of fluid dynamics. The classical Uzawa method is based on the following stationary iteration: given an initial guess $\mathbf{p}_0$, compute

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{F}^{-1}(\mathbf{f} - \mathbf{B}^T\mathbf{p}_n), \\ \mathbf{p}_{n+1} &= \mathbf{p}_n + \omega(\mathbf{B}\mathbf{u}_{n+1} - \mathbf{g}), \quad n = 1, 2, \ldots, \end{aligned} \tag{4.43}$$

where $\omega > 0$ is a relaxation parameter. Thus, a linear system with the matrix $\mathbf{F}$ is solved in every iteration of the algorithm. It can be solved directly or by an inner iterative method.

As any stationary iterative method, the iteration formula (4.43) corresponds to a splitting of the matrix $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{N}}$ and it can be written in the form of a coupled stationary iteration

$$\mathbf{x}_{n+1} = \boldsymbol{\mathcal{M}}^{-1}\boldsymbol{\mathcal{N}}\mathbf{x}_n - \boldsymbol{\mathcal{M}}^{-1}\mathbf{b}, \tag{4.44}$$

where $\mathbf{x}_n = [\mathbf{u}_n, \mathbf{p}_n]^T, \mathbf{b} = [\mathbf{f}, \mathbf{g}]^T$ and

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & -\frac{1}{\omega}\mathbf{I}_p \end{bmatrix}, \quad \boldsymbol{\mathcal{N}} = \begin{bmatrix} \mathbf{0} & -\mathbf{B}^T \\ \mathbf{0} & -\frac{1}{\omega}\mathbf{I}_p \end{bmatrix}. \tag{4.45}$$

However, it can be also seen as a segregated method. If we eliminate $\mathbf{u}_{n+1}$ from the second equation of (4.43) using the first equation, we obtain

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \omega(\mathbf{B}\mathbf{F}^{-1}\mathbf{f} - \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T\mathbf{p}_n - \mathbf{g}), \tag{4.46}$$

which is, in fact, a Richardson iteration applied to the Schur complement system (4.40) (multiplied by -1). For saddle-point systems with symmetric positive definite matrix $\mathbf{F}$, this fact can be exploited to determine the optimal value of ω in terms of maximal and minimal eigenvalues of the negative Schur complement $-\mathbf{S} = \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T$, see [34].

The convergence of the Uzawa method is rather slow for linear systems obtained from discretization of the Navier–Stokes equations, especially for high Reynolds numbers. It can be improved by suitable preconditioning. For example, a preconditioned Uzawa algorithm in the form

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{F}^{-1}(\mathbf{f} - \mathbf{B}^T\mathbf{p}_n), \\ \mathbf{p}_{n+1} &= \mathbf{p}_n + \omega\mathbf{P}^{-1}(\mathbf{B}\mathbf{u}_{n+1} - \mathbf{g}), \end{aligned} \tag{4.47}$$

is considered in [34]. This is equivalent to a Richardson iteration applied to a preconditioned Schur complement system

$$\mathbf{P}^{-1}\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T\mathbf{p} = \mathbf{P}^{-1}(\mathbf{B}\mathbf{F}^{-1}\mathbf{f} - \mathbf{g}), \tag{4.48}$$

which indicates that the preconditioner $\mathbf{P}$ should be an approximation of the negative Schur complement.

4.2.3 SIMPLE

The SIMPLE method (Semi-Implicit Method for Pressure Linked Equations) is an iterative method widely used for numerical solution of the stationary incompressible Navier–Stokes equations. It is a segregated method proposed by Patankar and Spalding in [82] for finite volume discretization of the Navier–Stokes equations. It belongs to the class of pressure-correction schemes, where the momentum equations are solved first to obtain an intermediate velocity which does not fulfill the incompressibility condition. Then a

pressure correction is computed and used to project the intermediate velocity field onto the space of divergence-free vector fields.

In the original literature, the description of the SIMPLE method is usually mixed with the discretization of the equations itself and often not very clear. In [115], Wesseling gives a unified formulation of several stationary iterative methods for the Navier–Stokes equations in the framework of distributive iteration, including the SIMPLE algorithm. We follow his exposition here.

A distributive iterative method for a general linear system $\mathbf{Ax} = \mathbf{b}$ is based on right preconditioning by a matrix $\mathbf{B}$, i.e. solving

$$\mathbf{ABy} = \mathbf{b}, \quad \mathbf{x} = \mathbf{By} \tag{4.49}$$

instead of the original system. The preconditioner $\mathbf{B}$ is chosen such that the matrix $\mathbf{AB}$ has some desirable properties for iterative solution, for example it is an M-matrix while $\mathbf{A}$ is not. Then, one proceeds similarly to Section 4.1.2: a splitting of the system matrix $\mathbf{AB} = \mathbf{M} - \mathbf{N}$ is chosen, which corresponds to the following splitting of the original matrix

$$\mathbf{A} = \mathbf{MB}^{-1} - \mathbf{NB}^{-1}. \tag{4.50}$$

The resulting stationary iteration takes the form

$$\mathbf{MB}^{-1}\mathbf{x}_{n+1} = \mathbf{NB}^{-1}\mathbf{x}_n + \mathbf{b}, \tag{4.51}$$

which can be also written as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{BM}^{-1}(\mathbf{b} - \mathbf{Ax}_n). \tag{4.52}$$

The method is called distributive because the correction $\mathbf{M}^{-1}(\mathbf{b}-\mathbf{Ax}_n)$ corresponding to a non-distributive iterative method (compare with (4.8)) is distributed over the elements of $\mathbf{x}_{n+1}$.

Since the method defined by (4.52) is obviously consistent for any nonsingular $\mathbf{B}$ and $\mathbf{M}$, these matrices do not need to be exactly the same as in (4.50). For example, a sophisticated preconditioner $\mathbf{B}$ can be used in order to design a suitable $\mathbf{M}$ and then both matrices can be slightly changed to make (4.52) more practical.

For saddle-point systems, a preconditioning matrix $\boldsymbol{\mathcal{B}}$ can be chosen such that the matrix $\boldsymbol{\mathcal{AB}}$ is block lower triangular. This leads to decoupling of the two variables (velocity and pressure in our case) and the splitting $\boldsymbol{\mathcal{AB}} = \boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{N}}$ can be obtained by splitting the diagonal blocks. For example, the matrix $\boldsymbol{\mathcal{B}}$ can be defined as

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{I}_u & -\mathbf{F}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix} \tag{4.53}$$

leading to the preconditioned matrix

$$\boldsymbol{\mathcal{AB}} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & \mathbf{S} \end{bmatrix}, \tag{4.54}$$

where $\mathbf{S}$ is the Schur complement.

The original SIMPLE algorithm is obtained for the splitting $\boldsymbol{\mathcal{AB}} = \boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{N}}$ with

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & \widehat{\mathbf{S}}_D \end{bmatrix}, \tag{4.55}$$

where $\widehat{\mathbf{S}}_D = -\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T$ is an approximation of $\mathbf{S}$ obtained by replacing the inverse of $\mathbf{F}$ by the inverse of its main diagonal $\mathbf{D} = \mathrm{diag}(\mathbf{F})$. The block $\mathbf{F}$ in (4.55) could be also replaced by its approximation representing, for example, an inner iterative method for approximate solution of linear systems with $\mathbf{F}$. Further, the matrix $\boldsymbol{\mathcal{B}}$ in the distribution step (4.52) is replaced by

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{I}_u & -\mathbf{D}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix}. \tag{4.56}$$

This results in the following stationary iteration

$$\begin{bmatrix} \mathbf{u}_{n+1} \\ \mathbf{p}_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_n \\ \mathbf{p}_n \end{bmatrix} + \begin{bmatrix} \mathbf{I}_u & -\mathbf{D}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix} \begin{bmatrix} \mathbf{F}^{-1} & \mathbf{0} \\ -\widehat{\mathbf{S}}_D^{-1}\mathbf{B}\mathbf{F}^{-1} & \widehat{\mathbf{S}}_D^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{r}_{u,n} \\ \mathbf{r}_{p,n} \end{bmatrix}, \tag{4.57}$$

where $[\mathbf{r}_{u,n}, \mathbf{r}_{p,n}]^T$ is the residual vector for $[\mathbf{u}_n, \mathbf{p}_n]^T$.

Let us express the new velocity and pressure as a sum of the approximation from the previous iteration and a correction $\delta\mathbf{u}, \delta\mathbf{p}$, respectively. Then, given $\mathbf{u}_n$ and $\mathbf{p}_n$, one iteration of the SIMPLE algorithm is performed in the following steps:

1. Compute a preliminary velocity correction $\delta\mathbf{u}^*$ as a solution of

$$\mathbf{F}\delta\mathbf{u}^* = \mathbf{f} - \mathbf{F}\mathbf{u}_n - \mathbf{B}^T\mathbf{p}_n. \tag{4.58}$$

2. Compute a preliminary pressure correction $\delta\mathbf{p}^*$ by solving

$$\widehat{\mathbf{S}}_D\delta\mathbf{p}^* = \mathbf{g} - \mathbf{B}\mathbf{u}_n - \mathbf{B}\delta\mathbf{u}. \tag{4.59}$$

3. Distribute the preliminary corrections using the matrix $\boldsymbol{\mathcal{B}}$ to obtain new corrections

$$\begin{aligned} \delta\mathbf{u} &= \delta\mathbf{u}^* - \mathbf{D}^{-1}\mathbf{B}^T\delta\mathbf{p}^* \\ \delta\mathbf{p} &= \delta\mathbf{p}^*. \end{aligned} \tag{4.60}$$

4. Update the velocity and pressure solution using the new corrections. In the classical SIMPLE method, this step is performed with underrelaxation:

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{u}_n + \alpha_u\delta\mathbf{u}, \\ \mathbf{p}_{n+1} &= \mathbf{p}_n + \alpha_p\delta\mathbf{p}, \end{aligned} \tag{4.61}$$

   where the underrelaxation parameters α_u, α_p are taken from the interval $(0, 1]$.

It is usually seen it the literature that an intermediate velocity $\mathbf{u}^*$ is computed in the first step of the algorithm instead of the preliminary correction $\delta\mathbf{u}^*$. The algorithm can be easily reformulated in this sense using $\mathbf{u}^* = \mathbf{u}_n + \delta\mathbf{u}^*$, which leads to Algorithm 5. This saves some computational work, since we avoid evaluating $\mathbf{F}\mathbf{u}_n$ and $\mathbf{B}\mathbf{u}_n$ on the right-hand side of (4.58) and (4.59), respectively. In this variant of the algorithm, only an initial pressure guess $\mathbf{p}_0$ is needed at the beginning of the iteration process. We note that the velocity update step in Algorithm 5 does not involve any underrelaxation. However, the underrelaxation can be incorporated into the algorithm also in this case, see, e.g., [111].

Several variations of the SIMPLE method have been proposed in the literature, e.g., SIMPLER (SIMPLE Revised) [81], SIMPLEC (SIMPLE Consistent) [29] or PISO (Pressure-Implicit with Splitting of Operators) [64], which was originally designed specifically for time-dependent problems. We will describe the first mentioned variant, SIMPLER, in more detail.

Algorithm 5: SIMPLE method

Input: matrix $\boldsymbol{\mathcal{A}}$, right-hand side vectors $\mathbf{f}, \mathbf{g}$, initial pressure guess $\mathbf{p}_0$.
Output: approximate velocity $\mathbf{u}_n$ and pressure $\mathbf{p}_n$.

1 **for** $n = 1, 2, \ldots,$ *until convergence* **do**
2 Solve $\mathbf{F}\mathbf{u}^* = \mathbf{f} - \mathbf{B}^T\mathbf{p}_n$.
3 Solve $\widehat{\mathbf{S}}_D\delta\mathbf{p} = \mathbf{g} - \mathbf{B}\mathbf{u}^*$.
4 Update $\mathbf{u}_{n+1} = \mathbf{u}^* - \mathbf{D}^{-1}\mathbf{B}^T\delta\mathbf{p}$.
5 Update $\mathbf{p}_{n+1} = \mathbf{p}_n + \alpha_p\delta\mathbf{p}$.

SIMPLER

The idea of the SIMPLER method is that instead of using the pressure $\mathbf{p}_n$ from the previous iteration, a new pressure $\mathbf{p}^*$ is computed first. Assuming that the exact velocity is known, we have

$$\mathbf{B}^T\mathbf{p} = \mathbf{f} - \mathbf{F}\mathbf{u} \tag{4.62}$$

from the discrete momentum equation. Further, we write the matrix $\mathbf{F}$ as a sum of its diagonal and off-diagonal part, i.e., $\mathbf{F} = \mathbf{D} + (\mathbf{F} - \mathbf{D})$, and multiply both sides of (4.62) by the matrix $-\mathbf{B}\mathbf{D}^{-1}$ to obtain

$$\begin{aligned} -\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T\mathbf{p} &= -\mathbf{B}\mathbf{D}^{-1}(\mathbf{f} - \mathbf{D}\mathbf{u} - (\mathbf{F} - \mathbf{D})\mathbf{u}) \\ &= \mathbf{B}\mathbf{u} - \mathbf{B}\mathbf{D}^{-1}(\mathbf{f} - (\mathbf{F} - \mathbf{D})\mathbf{u}). \end{aligned} \tag{4.63}$$

Finally, using the discrete continuity equation $\mathbf{B}\mathbf{u} = \mathbf{g}$, we get that the pressure is a solution of the linear system

$$\widehat{\mathbf{S}}_D\mathbf{p} = \mathbf{g} - \mathbf{B}\mathbf{D}^{-1}(\mathbf{f} - (\mathbf{F} - \mathbf{D})\mathbf{u}). \tag{4.64}$$

Thus, in the first step of one SIMPLER iteration, an intermediate pressure $\mathbf{p}^*$ is obtained by solving the linear system

$$\widehat{\mathbf{S}}_D\mathbf{p}^* = \mathbf{g} - \mathbf{B}\mathbf{D}^{-1}(\mathbf{f} - (\mathbf{F} - \mathbf{D})\mathbf{u}_n) \tag{4.65}$$

and used instead of $\mathbf{p}_n$ in (4.58) or in step 2 of Algorithm 5. Only an initial velocity $\mathbf{u}_0$ is needed to start the iteration. This modification results in faster convergence compared to the original SIMPLE method, but also in increased computational cost per iteration since an additional linear system for pressure is solved in every iteration.

A formulation of the SIMPLER algorithm as a distributive iterative method can be found, for example, in [112]. Denote the matrices $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{B}}$ defined in (4.55) and (4.56), respectively, as $\boldsymbol{\mathcal{M}}_R$ and $\boldsymbol{\mathcal{B}}_R$, since their derivation was based on right preconditioning of the original system, which led to a block lower triangular preconditioned matrix. We can also use a left preconditioning with the matrix

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{I}_u & \mathbf{0} \\ -\mathbf{B}\mathbf{F}^{-1} & \mathbf{I}_p \end{bmatrix} \tag{4.66}$$

leading to a block upper triangular matrix $\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}$. Following similar steps as in the derivation of SIMPLE, we can define a stationary iteration of the form

$$\begin{bmatrix} \mathbf{u}_{n+1} \\ \mathbf{p}_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_n \\ \mathbf{p}_n \end{bmatrix} + \boldsymbol{\mathcal{M}}_L^{-1}\boldsymbol{\mathcal{B}}_L \begin{bmatrix} \mathbf{r}_{u,n} \\ \mathbf{r}_{p,n} \end{bmatrix}, \tag{4.67}$$

where

$$\boldsymbol{\mathcal{M}}_L = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & \widehat{\mathbf{S}}_D \end{bmatrix}, \quad \boldsymbol{\mathcal{B}}_L = \begin{bmatrix} \mathbf{I}_u & \mathbf{0} \\ -\mathbf{B}\mathbf{D}^{-1} & \mathbf{I}_p \end{bmatrix}. \tag{4.68}$$

One iteration of the SIMPLER algorithm can be expressed as a combination of one iteration defined by the formula (4.67) and one iteration of SIMPLE (4.57), that is, performing

$$\begin{bmatrix} \mathbf{u}^* \\ \mathbf{p}^* \end{bmatrix} = \begin{bmatrix} \mathbf{u}_n \\ \mathbf{p}_n \end{bmatrix} + \boldsymbol{\mathcal{M}}_L^{-1} \boldsymbol{\mathcal{B}}_L \begin{bmatrix} \mathbf{r}_{u,n} \\ \mathbf{r}_{p,n} \end{bmatrix}, \tag{4.69}$$

followed by

$$\begin{bmatrix} \mathbf{u}_{n+1} \\ \mathbf{p}_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}^* \\ \mathbf{p}^* \end{bmatrix} + \boldsymbol{\mathcal{B}}_R \boldsymbol{\mathcal{M}}_R^{-1} \begin{bmatrix} \mathbf{r}_u^* \\ \mathbf{r}_p^* \end{bmatrix}, \tag{4.70}$$

where $[\mathbf{r}_u^*, \mathbf{r}_p^*]^T$ is the residual vector for $[\mathbf{u}^*, \mathbf{p}^*]^T$. After substituting the intermediate solution $[\mathbf{u}^*, \mathbf{p}^*]^T$ from (4.69) into (4.70) and some algebraic manipulations, we obtain the following iteration formula of the SIMPLER method

$$\begin{bmatrix} \mathbf{u}_{n+1} \\ \mathbf{p}_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_n \\ \mathbf{p}_n \end{bmatrix} + \boldsymbol{\mathcal{Q}} \begin{bmatrix} \mathbf{r}_{u,n} \\ \mathbf{r}_{p,n} \end{bmatrix}, \tag{4.71}$$

where

$$\begin{aligned} \boldsymbol{\mathcal{Q}} &= \boldsymbol{\mathcal{B}}_R \boldsymbol{\mathcal{M}}_R^{-1} - \boldsymbol{\mathcal{B}}_R \boldsymbol{\mathcal{M}}_R^{-1} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{M}}_L^{-1} \boldsymbol{\mathcal{B}}_L + \boldsymbol{\mathcal{M}}_L^{-1} \boldsymbol{\mathcal{B}}_L, \\ &= \boldsymbol{\mathcal{B}}_R \boldsymbol{\mathcal{M}}_R^{-1} \boldsymbol{\mathcal{B}}_L^{-1} \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{0} & 2\widehat{\mathbf{S}}_D \end{bmatrix} \boldsymbol{\mathcal{B}}_R^{-1} \boldsymbol{\mathcal{M}}_L^{-1} \boldsymbol{\mathcal{B}}_L. \end{aligned} \tag{4.72}$$

As noted in [112], the SIMPLER method is closely related to the symmetric block Gauss–Seidel method.

Chapter 5

Preconditioning

Preconditioning in the context of solving linear systems can play different roles. It can be used with both direct and iterative solution methods. In the case of direct methods, preconditioning can help reduce spreading of roundoff errors and achieve maximum accuracy in finite precision, see, e.g., [43]. The goal of preconditioning for iterative methods is to accelerate the convergence to the exact solution. Generally, the idea of preconditioning is to transform the original linear system $\mathbf{Ax} = \mathbf{b}$ into another system that has some favorable properties for solution with a numerical method. This transformation is realized by a matrix called preconditioner.

Effective preconditioning techniques for problems arising in a wide range of applications is an active research area in the past decades. We refer, e.g., to survey papers [89, 3] for an overview of the developments in this area up to the beginning of the 21st century, the book [17], where many example applications are covered, and more recent survey papers [114, 83]. According to Benzi [3], the term preconditioning first appeared in the literature in 1948 in a paper by Turing [102] dealing with roundoff errors in direct methods. Twenty years later, it was used in connection with iterative methods (specifically SSOR) for the first time by Evans [42], but the idea of apriori transformation of a linear system in order to ensure or improve convergence of iterative methods was used much earlier by Jacobi [65] or Cesari [16]. In this work, we are interested specifically in preconditioning for Krylov subspace methods. First ideas on preconditioning the conjugate gradient method appeared not long after its introduction in 1952. According to [89], already Hestenes's formulation of the method in [58] is equivalent to the preconditioned CG algorithm.

The preconditioner, denote it as $\mathbf{M}$, can be designed such that either $\mathbf{M} \approx \mathbf{A}$ or $\mathbf{M} \approx \mathbf{A}^{-1}$ in some sense, which is referred to as forward and inverse type of preconditioning, respectively, in [17]. Both types can be applied in several ways: from the left, from the right or using a combination of both. Considering the forward type, the left preconditioning leads to the transformed system

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}, \tag{5.1}$$

the right preconditioning leads to

$$\mathbf{AM}^{-1}\mathbf{y} = \mathbf{b}, \quad \mathbf{x} = \mathbf{M}^{-1}\mathbf{y}. \tag{5.2}$$

If the preconditioner is available in a factorized form $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$, mixed (or split) preconditioning can be applied such that

$$\mathbf{M}_1^{-1}\mathbf{AM}_2^{-1}\mathbf{y} = \mathbf{M}_1^{-1}\mathbf{b}, \quad \mathbf{x} = \mathbf{M}_2^{-1}\mathbf{y}. \tag{5.3}$$

In the case of inverse type preconditioning, the matrix $\mathbf{M}^{-1}$ (or $\mathbf{M}_1^{-1}, \mathbf{M}_2^{-1}$) are replaced by $\mathbf{M}$ (or $\mathbf{M}_1, \mathbf{M}_2$). We note that the matrix $\mathbf{M}$ does not have to be formed explicitly, it is only important to be able to solve linear systems with $\mathbf{M}$ in the case of forward type or multiply with $\mathbf{M}$ in the case of inverse type. We consider the forward type of preconditioning in this work.

The requirements on the properties of the preconditioner $\mathbf{M}$ depend on the iterative solution method used for solving the preconditioned system. For example, we can aim at minimizing the condition number of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$, clustering its eigenvalues, etc. In general, there are two main requirements on the preconditioner:

- the preconditioned system should be easy to solve (with the given method),
- the preconditioner should be cheap to construct and apply.

These two requirements go against each other. It is important to find a balance between them such that the resulting preconditioned method converges rapidly and one iteration is not too expensive at the same time.

As already mentioned, there are descriptive convergence bounds for CG based only on eigenvalues of the system matrix. Thus, in the case of linear systems with an SPD matrix, it is possible to identify the desirable properties of a good preconditioner a priori, at least theoretically (in exact arithmetic). However, we are interested in solution of nonsymmetric linear systems. There are no such eigenvalue-based descriptive bounds for the GMRES method and the theory of other Krylov subspace methods for nonsymmetric systems is even more limited. The ideas of preconditioning techniques for nonsymmetric systems are mostly heuristics based on practical experience with the problem at hand. There are, nevertheless, some useful tools that can provide convergence bounds for Krylov subspace methods like MINRES and GMRES, including the ε-pseudospectrum or field of values of the matrix, that can be used for analysis of the preconditioners. See, e.g., [4] for more details.

There are, in principle, two approaches to the construction of preconditioners. One of them is purely algebraic, based only on the entries of the given matrix $\mathbf{A}$. This approach is suitable in situations when we have little or no knowledge of the underlying problem. Such preconditioners are (almost) universally applicable and can be quite efficient for a wide range of problems, however, they are not optimal for any particular problem and can also perform poorly in some cases. This class of preconditioners includes incomplete matrix factorizations, sparse approximate inverses or algebraic multigrid methods. For an overview of these methods, we refer to the survey paper [3] which is focused mainly on the purely algebraic preconditioning techniques. The other approach to the construction of preconditioners is problem-specific, exploiting knowledge of the underlying problem, e.g., in the case of PDEs, the original continuous equations, computational domain, discretization, boundary conditions, etc. These preconditioners can be very effective for the narrow class of problems they are tailored for. Since we are interested a specific problem, the incompressible Navier–Stokes equations, we consider methods based on the second approach in this work.

5.1 Preconditioned GMRES

As already mentioned, a preconditioner can be applied from the left, from the right or from both sides. The convergence of GMRES can be different for different variants of

preconditioning, although all matrices in (5.1), (5.2) and (5.3) are similar and thus have the same eigenvalues. Right preconditioning is often preferred for GMRES, since the method then minimizes the norm of the residual for the original system $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$. On the other hand, if left or mixed preconditioning are applied, the method minimizes the norm of the preconditioned residual. However, according to [114], a good preconditioner leads to fast convergence for all three forms of preconditioning.

In this section, we describe the algorithm of GMRES with left and right preconditioning. The algorithm with split preconditioning can be easily derived as their combination. These variants of preconditioned GMRES are discussed, e.g., in Chapter 9 of [87]. We also mention so called flexible variants of the method which allow for preconditioners that vary during the iteration process.

5.1.1 Left preconditioning

Considering the forward type preconditioning where $\mathbf{M} \approx \mathbf{A}$, the left preconditioned version of GMRES is a straightforward application of the original algorithm to the preconditioned system (5.1), see Algorithm 6 for the individual steps.

The residual vectors computed during the algorithm, denoted as $\mathbf{z}_n$ in this case, correspond to the residual vectors of the preconditioned system (or preconditioned residuals)

$$\mathbf{z}_n = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_n). \tag{5.4}$$

The search space $\mathcal{S}_n$ is the Krylov subspace generated by the preconditioned matrix and the vector $\mathbf{z}_0$, i.e., $\mathcal{S}_n = \mathcal{K}_n(\mathbf{M}^{-1}\mathbf{A}, \mathbf{z}_0)$. Thus, in the n-th iteration of the left preconditioned GMRES method, the norm of the preconditioned residual $||\mathbf{z}_n||$ is minimized over the solution space $\mathbf{x}_0 + \mathcal{K}_n(\mathbf{M}^{-1}\mathbf{A}, \mathbf{z}_0)$ (recall the optimality property (4.34)).

The stopping criterion is usually based on the preconditioned residuals $\mathbf{z}_n$, since the original residual vectors $\mathbf{r}_n$ are not available, unless they are explicitly computed from $\mathbf{z}_n$. This can result in stopping the algorithm prematurely or with delay compared to the stopping criterion based on $\mathbf{r}_n$, especially when the preconditioner $\mathbf{M}$ is very ill-conditioned [87].

Algorithm 6: GMRES method with left preconditioning

Input: nonsingular matrices $\mathbf{A}, \mathbf{M} \in \mathbb{R}^{N\times N}$, right-hand side $\mathbf{b} \in \mathbb{R}^N$, initial approximation $\mathbf{x}_0 \in \mathbb{R}^N$, stopping tolerance ε.

Output: approximation $\mathbf{x}_n$.

1 Compute $\mathbf{z}_0 = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0)$, $\mathbf{v}_1 = \mathbf{z}_0/||\mathbf{z}_0||$.

2 **for** $n = 1, 2, \ldots$ **do**

3 Compute the n-th step of the Arnoldi algoritnm for $\mathbf{M}^{-1}\mathbf{A}$ and $\mathbf{v} = \mathbf{z}_0$.

4 Update the QR factorization of $\underline{\mathbf{H}}_n$.

5 If the stopping criterion is satisfied, stop the iteration and go to 6.

6 Compute $\mathbf{t}_n$ as the minimizer of $||\,||\mathbf{z}_0||\vec{e}_1 - \underline{\mathbf{H}}_n t||$ and $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n\mathbf{t}_n$.

5.1.2 Right preconditioning

If right preconditioning is applied, the linear system to be solved takes the form

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{y} = \mathbf{b}, \tag{5.5}$$

where the new variable $\mathbf{y} = \mathbf{M}\mathbf{x}$. However, the variable $\mathbf{y}$ does not have to be used explicitly in the GMRES algorithm, see Algorithm 7.

Only the initial residual vector is needed to start the Arnoldi algorithm. Given an initial guess $\mathbf{x}_0$ for the $\mathbf{x}$ variable and $\mathbf{y}_0 = \mathbf{M}\mathbf{x}_0$, the initial residual is $\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{y}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{r}_0$. Thus, the initial residual for the preconditioned system (5.5) is equal to the initial residual for the original system $\mathbf{A}\mathbf{x} = \mathbf{b}$ and it can be computed directly from $\mathbf{x}_0$. In step 6 of Algorithm 7, the approximate solution $\mathbf{y}_n$ would be computed as $\mathbf{y}_n = \mathbf{y}_0 + \mathbf{V}_n\mathbf{t}_n$. By multiplying this by the matrix $\mathbf{M}^{-1}$, we obtain the relation for $\mathbf{x}_n$.

The Krylov subspace whose orthogonal basis is constructed during the algorithm is again generated by the preconditioned matrix and the initial residual, i.e., $\mathcal{K}_n(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$. Thus, the n-th approximate solution $\mathbf{y}_n$ lies in the affine space $\mathbf{y}_0 + \mathcal{K}_n(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$ and also $\mathbf{x}_n \in \mathbf{x}_0 + \mathbf{M}^{-1}\mathcal{K}_n(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$. It can be easily shown (see [87]) that the solution space for the $\mathbf{x}$ variable is identical to the solution space of the left preconditioned GMRES. Similarly to the initial residual, all residuals computed during the right preconditioned GMRES are equal to the corresponding residuals for the original system. Thus, the algorithm minimizes the norm of the original residual $||\mathbf{r}_n||$ over the same space as the left preconditioned version.

Algorithm 7: GMRES method with right preconditioning

Input: nonsingular matrices $\mathbf{A}, \mathbf{M} \in \mathbb{R}^{N \times N}$, right-hand side $\mathbf{b} \in \mathbb{R}^N$, initial approximation $\mathbf{x}_0 \in \mathbb{R}^N$, stopping tolerance ε.

Output: approximation $\mathbf{x}_n$.

1 Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_1 = \mathbf{r}_0/||\mathbf{r}_0||$.

2 **for** $n = 1, 2, \ldots$ **do**

3 Compute the n-th step of the Arnoldi algoritnm for $\mathbf{A}\mathbf{M}^{-1}$ and $\mathbf{v} = \mathbf{r}_0$.

4 Update the QR factorization of $\underline{\mathbf{H}}_n$.

5 If the stopping criterion is satisfied, stop the iteration and go to 6.

6 Compute $\mathbf{t}_n$ as the minimizer of $||\, ||\mathbf{r}_0||\vec{e}_1 - \underline{\mathbf{H}}_n t||$ and $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{M}^{-1}\mathbf{V}_n\mathbf{t}_n$.

5.1.3 Variable preconditioning

In the above, it was assumed that the preconditioner is fixed, i.e., the same linear operator is used as preconditioner in each iteration. However, the preconditioner can be designed such that the operator varies from iteration to iteration, for example by incorporating some information from previous iterations or approximating the application of $\mathbf{M}^{-1}$ by some inner iterative process. In such cases, so called flexible variants of Krylov subspace methods that allow for variable preconditioning have to be used. There are several flexible variants of GMRES, see, for example, FGMRES (Flexible GMRES) [86] or GMRESR [110].

5.2 Krylov acceleration of stationary iterations

As already mentioned in Section 4.1.2, stationary iterative methods are often used as preconditioners for Krylov subspace methods rather than standalone solvers. However, this does not mean that the application of the preconditioner is realized by an inner iterative process, which would lead to variable preconditioning mentioned in Section 5.1.3.

In practice, using a stationary iterative method as a preconditioner usually refers to defining a fixed preconditioning operator as described for example in Chapter 7 of [40].

Recall that every consistent stationary iterative method corresponds to a splitting of the matrix $\mathbf{A} = \mathbf{M} - \mathbf{N}$ and the iteration formula can be written in the form (4.8), i.e., the n-th iteration is obtained as

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_{n-1}). \tag{5.6}$$

Multiplication by the matrix $-\mathbf{A}$ and adding the vector $\mathbf{b}$ to both sides of (5.6) leads to the following relations for the n-th residual

$$\begin{aligned} \mathbf{r}_n &= \mathbf{r}_{n-1} - \mathbf{A}\mathbf{M}^{-1}\mathbf{r}_{n-1} \\ &= (\mathbf{I} - \mathbf{A}\mathbf{M}^{-1})\mathbf{r}_{n-1} \\ &= (\mathbf{I} - \mathbf{A}\mathbf{M}^{-1})^n\mathbf{r}_0. \end{aligned} \tag{5.7}$$

Thus, the n-th residual can be written in the form $\mathbf{r}_n = p_n(\mathbf{A}\mathbf{M}^{-1})\mathbf{r}_0$, where p_n is a polynomial of degree n with unit absolute coefficient. This means that $\mathbf{r}_n \in \mathbf{r}_0 + \mathbf{A}\mathbf{M}^{-1}\mathcal{K}_n(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$, which is the same space in which lies the residual generated by a Krylov subspace method applied to the linear system preconditioned with the matrix $\mathbf{M}$. If we use GMRES to solve such system, then because of the optimality property (4.34), the residual norm $||\mathbf{r}_n||$ is minimized over the mentioned space. Thus, solving the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ by preconditioned GMRES with the preconditioner $\mathbf{M}$ can be viewed as acceleration of the stationary iterative method defined by the splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$.

One of the simplest examples is the Jacobi method for which $\mathbf{M} = \mathbf{D}$, where $\mathbf{D}$ is the main diagonal of $\mathbf{A}$. Using Jacobi method as a preconditioner then corresponds to dividing each equation of the linear system by the corresponding diagonal element of the system matrix.

One of the best known stationary iterative methods used for solving saddle-point linear systems is the Uzawa method and its preconditioned version described in Section 4.2.2. The preconditioned version corresponds to the splitting $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{N}}$, where

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & -\frac{1}{\omega}\mathbf{P} \end{bmatrix}, \tag{5.8}$$

where the matrix $\mathbf{P}$ should be a good preconditioner for the negative Schur complement $-\mathbf{S} = \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T$. Thus, using the preconditioned Uzawa method as a preconditioner is closely related to the block triangular preconditioners that will be described in detail in the following section.

5.3 Navier–Stokes preconditioners

This section is devoted to an overview of the problem-specific preconditioners developed for the linear systems arising from discretizations of the linearized incompressible Navier–Stokes equations. Thus, we consider the saddle-point linear systems of the form (4.1) with the coefficient matrix

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix}. \tag{5.9}$$

For construction of preconditioners for such linear systems, it is beneficial to exploit the block structure of this matrix as well as the origin and structure of the individual blocks.

Note that we consider inf-sup stable pairs of discretization spaces in this work, which results in the zero (2,2) block in the matrix (5.9) and thus its block triangular structure. If, on the contrary, stabilization is used to circumvent the inf-sup condition (2.46), a nonzero block appears at the (2,2) position. However, the block structure of the matrix is still important also in this case, since the block rows of the linear system correspond to different quantities (velocity and pressure) and the individual blocks are associated with certain continuous operators, which can be exploited.

When trying to construct a good preconditioner for GMRES (or other Krylov subspace methods for nonsymmetric systems), we can aim at obtaining a preconditioned matrix, say $\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{M}}^{-1}$, with a low degree minimal polynomial $p_{\min}(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{M}}^{-1})$. Recall that we have denoted the minimal polynomial degree as $D = D(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{M}}^{-1})$. As indicated in Section 4.1.3, Krylov subspace methods give the exact solution of the linear system in at most D iterations for any initial residual $\mathbf{r}_0$ in exact arithmetic and a small D would probably lead to a fast convergence in finite precision arithmetic. However, preconditioners with such properties will often not satisfy the second requirement mentioned in the introduction of this chapter, i.e. that they should be cheap to construct and apply. Therefore, these "ideal" preconditioners are used only as a starting point for construction of practical preconditioners. Although eigenvalues of the preconditioned matrix do not fully describe the behavior of Krylov subspace methods for nonsymmetric systems, it can be seen from experiments of many authors that a clustered spectrum away from the origin often results in fast convergence. Ideally, the eigenvalues should be bounded independently of the problem parameters such as the discretization mesh size h or the Reynolds number.

Probably the most widely used group of preconditioners in the field of fluid dynamics are the so called block preconditioners. They usually possess some special block structure and are related to the segregated solution approaches described in Section 4.2. They can be also usually viewed as based on the block LDU decomposition of the system matrix $\boldsymbol{\mathcal{A}}$,

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_u & \mathbf{0} \\ \mathbf{B}\mathbf{F}^{-1} & \mathbf{I}_p \end{bmatrix} \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I}_u & \mathbf{F}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix}, \tag{5.10}$$

or one of the block LU decompositions

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_u & \mathbf{0} \\ \mathbf{B}\mathbf{F}^{-1} & \mathbf{I}_p \end{bmatrix} \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \tag{5.11}$$

$$= \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I}_u & \mathbf{F}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix}, \tag{5.12}$$

where $\mathbf{S}$ is again the Schur complement.

We refer to Loghin and Wathen [73] for a general analysis of block preconditioners for saddle-point linear systems based on the stability (inf-sup) conditions and the field-of-values equivalence.

A simple block structure for a preconditioner could be block diagonal. As pointed out by Murphy et al. [76], preconditioning the saddle-point linear system with a preconditioner of the form

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{bmatrix} \tag{5.13}$$

leads to a preconditioned matrix with minimal polynomial of degree $D = 3$ and thus termination of the Krylov subspace method in at most 3 iterations. However, we do

not consider block diagonal preconditioners in this work, see, for example, [40, 5] for more details. They are especially suitable for symmetric saddle-point systems, since they preserve the symmetry. However, we are interested in solving a nonsymmetric system and thus we do not need a symmetric preconditioner. The block triangular preconditioners (described in Section 5.3.1 below) are very similar and they require only one additional matrix-vector multiplication. Moreover, it can be shown that the GMRES iteration with a block triangular preconditioner takes half as many steps as with an analogous block diagonal preconditioner for certain initial vectors, see [40, 46].

In the rest of this section, we describe several approaches to construction of practical block triangular preconditioners and a group of block preconditioners based on the SIMPLE method.

5.3.1 Block triangular preconditioners

The family of block triangular preconditioners has been a subject of active research in the field of solution techniques for saddle-point problems during recent years. Some of the most efficient methods for solving the Navier–Stokes linear systems belong to this group.

The "ideal" block triangular preconditioner is of the form

$$\boldsymbol{\mathcal{M}}_t = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{S} \end{bmatrix}. \tag{5.14}$$

It is obvious from the block LU decomposition (5.11) that the right preconditioned matrix $\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{M}}_t^{-1}$ is block lower triangular with identity diagonal blocks, hence with a single eigenvalue equal to one. The minimal polynomial of this matrix is of degree 2 (see [76]) and thus GMRES would terminate in at most 2 iterations. The left preconditioned matrix can be obtained by the similarity transformation

$$\boldsymbol{\mathcal{M}}_t^{-1}(\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{M}}_t^{-1})\boldsymbol{\mathcal{M}}_t = \boldsymbol{\mathcal{M}}_t^{-1}\boldsymbol{\mathcal{A}}. \tag{5.15}$$

Since similar matrices share the same eigenvalues and minimal polynomial, left preconditioning with $\boldsymbol{\mathcal{M}}_t$ would also lead to termination of GMRES in at most 2 iterations. The same holds for split preconditioning.

We could also use a block lower triangular preconditioner based on the block LU decomposition (5.12). However, the block upper triangular form (5.14) is usually considered, which we will follow in this text.

The application of the preconditioner to a vector $\mathbf{r}$, i.e. computation of a vector $\mathbf{z} = \boldsymbol{\mathcal{M}}_t^{-1}\mathbf{r}$, is performed by solving the linear system

$$\begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{z}_u \\ \mathbf{z}_p \end{bmatrix} = \begin{bmatrix} \mathbf{r}_u \\ \mathbf{r}_p \end{bmatrix} \tag{5.16}$$

in the steps summarized in Algorithm 8.

Algorithm 8: Application of $\boldsymbol{\mathcal{M}}_t^{-1}$

1 Solve $\mathbf{S}\mathbf{z}_p = \mathbf{r}_p$.
2 Update $\mathbf{r}_u = \mathbf{r}_u - \mathbf{B}^T\mathbf{z}_p$.
3 Solve $\mathbf{F}\mathbf{z}_u = \mathbf{r}_u$.

This algorithm corresponds exactly to the Schur complement reduction method described in Section 4.2.1. As stated there, this method is not practical since it requires an explicit construction of the Schur complement and solution of a linear system with it. The construction of $\mathbf{S}$ would be very expensive as well as the solution of the linear system because $\mathbf{S}$ is typically a dense matrix. Therefore, practical block triangular preconditioners use some inexpensive approximation $\widehat{\mathbf{S}} \approx \mathbf{S}$. The block upper triangular preconditioner then takes the form

$$\widehat{\boldsymbol{\mathcal{M}}}_t = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & \widehat{\mathbf{S}} \end{bmatrix}. \tag{5.17}$$

The choice of the approximation $\widehat{\mathbf{S}}$ yields different preconditioners. Two most popular choices will be described in Section 5.3.2 and a slightly different approach leading to a block triangular preconditioner in Section 5.3.3. We note that the Uzawa preconditioner (5.8) is nothing but a block lower triangular preconditioner where some of the approximations below can be used for the block $\mathbf{P}$ that should also approximate the (negative) Schur complement.

The application of $\widehat{\boldsymbol{\mathcal{M}}}_t$ requires solving one linear system with the matrix $\mathbf{F}$ and one linear system with $\widehat{\mathbf{S}}$ (or multiplication by $\widehat{\mathbf{S}}^{-1}$ if the inverse is defined explicitly). In our case of Picard linearization of the Navier–Stokes equations, the matrix $\mathbf{F}$ consists of d decoupled discrete convection–diffusion operators, thus the solution of the linear system with $\mathbf{F}$ requires solving several convection–diffusion subproblems. The application of the inverse approximate Schur complement $\widehat{\mathbf{S}}^{-1}$ often requires solution of subproblems with a Poisson-type discrete operator or mass matrix. In order for $\widehat{\boldsymbol{\mathcal{M}}}_t$ to be an efficient preconditioner, we need fast approximate solvers for all subproblems. We will comment on approximate solution of the subproblems in more detail later in Section 5.3.5. For now, we assume that all subproblems are solved exactly, which is referred to as the ideal version of the preconditioner $\widehat{\boldsymbol{\mathcal{M}}}_t$ (here, the word "ideal" does not refer to (5.14)).

Eigenvalues

As mentioned above, it is desirable that the preconditioned matrix has a clustered spectrum away from the origin and bounded independently of the problem parameters. Therefore, we are interested in the eigenvalues of the matrix

$$\begin{aligned} \boldsymbol{\mathcal{A}}\widehat{\boldsymbol{\mathcal{M}}}_t^{-1} &= \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & \widehat{\mathbf{S}} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{F}^{-1} & -\mathbf{F}^{-1}\mathbf{B}^T\widehat{\mathbf{S}}^{-1} \\ \mathbf{0} & \widehat{\mathbf{S}}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I}_u & \mathbf{0} \\ \mathbf{B}\mathbf{F}^{-1} & -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T\widehat{\mathbf{S}}^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_u & \mathbf{0} \\ \mathbf{B}\mathbf{F}^{-1} & \mathbf{S}\widehat{\mathbf{S}}^{-1} \end{bmatrix}. \end{aligned} \tag{5.18}$$

It follows from the Leibniz formula for determinants that the determinant of a block triangular matrix is equal to the product of determinants of its diagonal blocks. Thus, the following holds:

$$\det(\boldsymbol{\mathcal{A}}\widehat{\boldsymbol{\mathcal{M}}}_t^{-1} - \lambda\boldsymbol{\mathcal{I}}) = \det(\mathbf{I}_u - \lambda\mathbf{I}_u) \cdot \det(\mathbf{S}\widehat{\mathbf{S}}^{-1} - \lambda\mathbf{I}_p), \tag{5.19}$$

where $\boldsymbol{\mathcal{I}}$ is the identity matrix of appropriate size. It holds analogically for any block triangular matrix. As a result, λ is an eigenvalue of the block triangular matrix if and

only if it is an eigenvalue of one of its diagonal blocks. Hence, the matrix $\boldsymbol{\mathcal{A}}\widehat{\boldsymbol{\mathcal{M}}}_t^{-1}$ has the eigenvalue 1 of multiplicity $d \cdot n_u$ and the rest of the eigenvalues are solutions of the eigenproblem

$$\mathbf{S}\widehat{\mathbf{S}}^{-1}\mathbf{p} = \lambda\mathbf{p} \tag{5.20}$$

or equivalently of the generalized eigenproblem

$$\mathbf{S}\widetilde{\mathbf{p}} = \lambda\widehat{\mathbf{S}}\widetilde{\mathbf{p}}, \quad \text{where } \widetilde{\mathbf{p}} = \widehat{\mathbf{S}}^{-1}\mathbf{p}. \tag{5.21}$$

This means that we have the same requirements for the Schur complement approximation $\widehat{\mathbf{S}}$ as for the whole preconditioner $\widehat{\boldsymbol{\mathcal{M}}}_t$: $\widehat{\mathbf{S}}$ should be constructed such that the eigenvalues of $\mathbf{S}\widehat{\mathbf{S}}^{-1}$ are clustered away from the origin and bounded independently of the problem parameters.

5.3.2 Approximations of the Schur complement

A good approximation of the Schur complement is useful not only for construction of the block diagonal and block triangular preconditioners described above, but also in other methods already mentioned in this work, such as the preconditioned Uzawa stationary iteration. In this section, we present several approaches to approximating the Schur complement $\mathbf{S} = -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T$.

Consider a Galerkin discretization of the Stokes equations for a moment. The obtained saddle-point matrix is symmetric and the Schur complement $\mathbf{S}_{\text{Stokes}} = -\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T$ is symmetric and negative definite (for inflow-outflow problems) or negative semidefinite (for enclosed flow problems). The matrix $\mathbf{A}$ is defined in (2.44) and it arises from discretization of the viscous term, i.e., the Laplace operator. It can be shown that for a stable Galerkin discretization with a shape-regular, quasi-uniform mesh, $\mathbf{S}_{\text{Stokes}}$ is spectrally equivalent to the negative pressure mass matrix $-\mathbf{M}_p$, which means that the following is satisfied

$$\gamma^2 \leq \frac{\mathbf{q}^T\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T\mathbf{q}}{\mathbf{q}^T\mathbf{M}_p\mathbf{q}} \leq \Gamma^2, \quad \forall\mathbf{q} \in \mathbb{R}^{n_p} \setminus \{\mathbf{0}\}, \tag{5.22}$$

where γ is the inf-sup constant in (2.46). Both γ and Γ are independent of the mesh parameter h and γ is bounded away from zero. See, e.g., [40] for more details. As pointed out for example in [5], it is not surprising that these two operators are related since $\mathbf{S}_{\text{Stokes}}$ can be viewed as a discrete counterpart of a pseudodifferential operator $-\operatorname{div}\Delta^{-1}\operatorname{grad}$ and the identity $\Delta = \operatorname{div}\operatorname{grad}$ holds. The relation (5.22) implies, that the eigenvalues of the generalized eigenproblem

$$\mathbf{S}_{\text{Stokes}}\widetilde{\mathbf{p}} = -\lambda\mathbf{M}_p\widetilde{\mathbf{p}} \tag{5.23}$$

lie between the two constants γ^2 and Γ^2, i.e., they are bounded independently of h. Thus, approximations based on the pressure mass matrix lead to effective preconditioners for the Stokes problem and h-independent convergence of the preconditioned Krylov subspace method. It is usually sufficient to take $\widehat{\mathbf{S}} = -\text{diag}(\mathbf{M}_\text{p})$ to obtain h-independent convergence. We again refer to [40] for more details as well as some numerical results.

A generalization of the Stokes preconditioner for the linearized Navier–Stokes equations using the approximation $\widehat{\mathbf{S}}_M = -\frac{1}{\nu}\mathbf{M}_p$ was presented and analyzed by Elman and Silvester in [38]. They show that the eigenvalues of $\mathbf{S}\widehat{\mathbf{S}}_M^{-1}$ are clustered and bounded

independently of h also in this case. However, the bounds on the eigenvalues depend on the viscosity ν such that the real part of the eigenvalues approaches zero as $\nu \to 0$. As a result, the convergence of Krylov subspace methods deteriorates as the viscosity decreases (the iteration counts increase roughly like $1/\nu$). See [38] for comparison of convergence for various mesh refinement levels and viscosity values.

Several approximations of $\mathbf{S}$ were later developed with the aim of reducing the sensitivity to ν. First of them, called the BFBt preconditioner, was introduced by Elman in [33]. Another effective and widely used approaches are the pressure convection–diffusion (PCD) preconditioner proposed by Kay et al. in [67, 68] and the least-squares commutator (LSC) preconditioner introduced by Elman et al. in [35], which is closely related to the BFBt preconditioner. The three mentioned approximations will be described in more detail in the rest of this section.

BFBt preconditioner

The construction of the approximate Schur complement in the BFBt preconditioner is based on Moore-Penrose pseudoinverse, although this term is not explicitly mentioned in the original paper [33]. The preconditioner is described using pseudoinverses for example in [5].

Recall the expression for the Schur complement $\mathbf{S} = -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T$. If $\mathbf{B}$ were a square, invertible matrix, the inverse of the Schur complement would take the form $\mathbf{S}^{-1} = -\mathbf{B}^{-T}\mathbf{F}\mathbf{B}^{-1}$. Since $\mathbf{B}$ is in fact a rectangular matrix, the inverses $\mathbf{B}^{-T}$ and $\mathbf{B}^{-1}$ do not exist, however, they can be replaced by some generalization of the inverse.

The Moore-Penrose pseudoinverse in one of the possible generalizations. It exists for any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ and it is denoted as $\mathbf{A}^\dagger$. One of its definitions is that the operators $\mathbf{A}\mathbf{A}^\dagger$ and $\mathbf{A}^\dagger\mathbf{A}$ are orthogonal projectors onto the range of $\mathbf{A}$ and the range of $\mathbf{A}^T$, respectively. It means that $\mathbf{A}\mathbf{A}^\dagger\mathbf{x} = \mathbf{x}$ if $\mathbf{x} \in \mathcal{R}(\mathbf{A})$ and $\mathbf{A}\mathbf{A}^\dagger\mathbf{x} = \mathbf{0}$ if $\mathbf{x} \in \mathcal{R}(\mathbf{A})^\perp$, analogously for the operator $\mathbf{A}^\dagger\mathbf{A}$. It is easy to verify that if the matrix $\mathbf{A}$ has full rank, the pseudoinverse can be expressed as a simple algebraic formula. If $m < n$ and $\operatorname{rank}(\mathbf{A}) = m$, i.e. the matrix $\mathbf{A}\mathbf{A}^T$ is invertible, then

$$\mathbf{A}^\dagger = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}. \tag{5.24}$$

On the other hand, if $m > n$ and $\operatorname{rank}(\mathbf{A}) = n$, the pseudoinverse is

$$\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T. \tag{5.25}$$

Thus, assuming that $\mathbf{B}$ has full rank, an approximation of the inverse Schur complement can be defined as

$$\begin{aligned}\widehat{\mathbf{S}}^{-1}_{\mathrm{BFBt}} &= -(\mathbf{B}^T)^\dagger\,\mathbf{F}\,\mathbf{B}^\dagger \\ &= -(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B}\mathbf{F}\mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}.\end{aligned} \tag{5.26}$$

The approximation of $\mathbf{S}$ is then given as

$$\widehat{\mathbf{S}}_{\mathrm{BFBt}} = -(\mathbf{B}\mathbf{B}^T)(\mathbf{B}\mathbf{F}\mathbf{B}^T)^{-1}(\mathbf{B}\mathbf{B}^T). \tag{5.27}$$

The block triangular preconditioner $\widehat{\mathcal{M}}_t$ in combination with $\widehat{\mathbf{S}}_{\mathrm{BFBt}}$ is called the BFBt preconditioner due to the occurrence of the discrete operator $\mathbf{B}\mathbf{F}\mathbf{B}^T$ in (5.27).

The application of $\widehat{\mathbf{S}}^{-1}_{\mathrm{BFBt}}$ to a vector requires solving two linear systems with the matrix $\mathbf{B}\mathbf{B}^T$, which corresponds to solving two Poisson-type problems on the pressure space. Further, matrix-vector multiplications with the matrices $\mathbf{B}^T, \mathbf{F}$ and $\mathbf{B}$ are required.

BFBt properties

It was shown in [33] that if the linear system (4.1) arises from the marker-and-cell (MAC) finite difference discretization of an Oseen problem with a constant wind $\boldsymbol{w}$ (see (2.17) for the formulation of the Oseen equations) and periodic boundary conditions, then $\mathbf{S} = \widehat{\mathbf{S}}_{\mathrm{BFBt}}$ holds. Thus, preconditioning such linear system with the BFBt preconditioner would lead to a preconditioned system with a single eigenvalue equal to one and convergence of Krylov subspace methods in at most 2 iterations. The numerical experiments in [33] indicate that the iteration counts increase proportionally to $h^{-1/2}$ for a Dirichlet problem and also proportionally to $\nu^{-1/2}$ if the wind is not constant. This is observed for both finite difference and finite element discretizations.

PCD preconditioner

The PCD preconditioner was originally proposed in [67], later published as [68], with the main goal to eliminate the dependence on the mesh parameter h. The derivation of the Schur complement approximation was based on the fundamental solution tensor which can be viewed as a continuous inverse of the Oseen operator. The choice of the approximation was later justified in a different way based on approximate commutators in [96]. We will briefly describe both approaches to the derivation of the PCD preconditioner in the following.

We do not want to go into details of fundamental solution theory in this work, therefore we comment on the first approach rather vaguely and refer to [68] for a deeper insight. Consider the following problem

$$\begin{aligned} \mathcal{L}u &= f \quad \text{in } \Omega \subseteq \mathbb{R}^d, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{5.28}$$

where $\mathcal{L}$ is a linear differential operator. A Green's function G of the operator $\mathcal{L}$ corresponding to the domain Ω and homogeneous Dirichlet boundary conditions is a solution of a problem similar to (5.28), where the source function f is replaced by the Dirac delta distribution. Then the solution u of (5.28) can be expressed as a convolution of the Green's function and the source function, i.e. $u = \int_\Omega Gf$. Unfortunately, it is not possible to obtain an explicit formula for the Green's function for a general domain Ω. A fundamental solution $\mathcal{G}$ for the operator $\mathcal{L}$ is basically the Green's function for $\Omega = \mathbb{R}^d$. The Green's function can be seen as a perturbation of the fundamental solution, $G = \mathcal{G} + g$, where the perturbation g is small everywhere inside Ω except the vicinity of the boundary. The definitions of the Green's function and the fundamental solution can be generalized for systems of differential equations giving the Green's tensor and fundamental solution tensor.

Let us consider Picard linearization of a steady-state Navier–Stokes problem leading to an Oseen problem, where the wind $\boldsymbol{w}$ represents the velocity field from the previous Picard iteration. A constant wind $\boldsymbol{w}$ is assumed instead for the derivation of the preconditioner. The idea of [68] is to approximate the Green's tensor of the Oseen operator on a bounded domain Ω by its fundamental solution tensor. Thus, the Oseen operator is considered on the whole space $\mathbb{R}^d$ and Fourier transform is used to find the fundamental solution tensor. The continuous counterpart of the inverse Schur complement $\mathbf{S}^{-1}$ is one of the components of the fundamental solution tensor and it takes the form $(-\nu\Delta + \boldsymbol{w}\cdot\nabla)(-\mathcal{G}_\Delta)$, where $\mathcal{G}_\Delta$ is the fundamental solution of the Laplacian.

Note that both mentioned operators, the Laplacian and the convection–diffusion operator $(-\nu\Delta + \boldsymbol{w}\cdot\nabla)$, now act on the pressure space, i.e. on scalar functions, unlike the corresponding operators appearing in the Oseen equations. Here we assume that the functions in the pressure space are sufficiently smooth such that it makes sense to consider these operators. Since $\mathcal{G}_\Delta$ can be again interpreted as continuous inverse of the Laplace operator, we can formally write the continuous inverse Schur complement as $(-\nu\Delta + \boldsymbol{w}\cdot\nabla)(-\Delta)^{-1}$. Intuitively, the discrete counterpart of this operator would be

$$\mathbf{S}^{-1} \approx -\mathbf{F}_p\mathbf{A}_p^{-1}, \tag{5.29}$$

where the matrices $\mathbf{F}_p$ and $\mathbf{A}_p$ are discretizations of the convection–diffusion operator and the Laplace operator on the pressure space, respectively. The pressure discretization basis has to be at least C^0-continuous in order to be able to define the discrete operators.

In the Stokes limit where $\boldsymbol{w} = \mathbf{0}$, the approximation (5.29) would simplify to $\mathbf{S}^{-1} \approx -\nu\mathbf{I}_p$. However, considering Galerkin discretization, the fundamental solution approach applied to the Stokes operator leads to the mass matrix Schur complement approximation mentioned above, $\widehat{\mathbf{S}}_M = -\frac{1}{\nu}\mathbf{M}_p$. Therefore, the pressure mass matrix is introduced into (5.29) such that the Schur complement approximation tends to $\widehat{\mathbf{S}}_M$ as $\boldsymbol{w} \to \mathbf{0}$. Thus, the PCD Schur complement approximation takes the form

$$\widehat{\mathbf{S}}_{\mathrm{PCD}} = -\mathbf{A}_p\mathbf{F}_p^{-1}\mathbf{M}_p. \tag{5.30}$$

The name pressure convection–diffusion preconditioner refers to the fact that the inverse discrete convection–diffusion operator on the pressure space appears in the Schur complement approximation instead of $\mathbf{F}^{-1}$ in the exact Schur complement $\mathbf{S}$.

Denote the convection–diffusion operator acting on the velocity space as $\mathcal{F}_u$ and the analogous operator acting on the pressure space as $\mathcal{F}_p$. The starting point for the derivation in [96, 35] is the assumption that the convection–diffusion operator and the gradient operator commute, i.e.

$$\mathcal{F}_u\nabla \equiv \nabla\mathcal{F}_p. \tag{5.31}$$

This equivalence holds for a constant wind $\boldsymbol{w}$ and $\Omega = \mathbb{R}^d$. If Ω is a bounded domain, (5.31) is true in its interior, but generally, it does not hold in real situations with a bounded domain and non-constant wind. However, the commutator

$$\mathcal{E} = \mathcal{F}_u\nabla - \nabla\mathcal{F}_p \tag{5.32}$$

is likely to be small in some sense for smooth $\boldsymbol{w}$ [35]. In the context of Galerkin discretization, the matrix representation of the gradient operator is $\mathbf{M}_u^{-1}\mathbf{B}^T$, similarly for the convection–diffusion operators on the velocity and pressure space. Note that $\mathbf{M}_u$ now denotes a block diagonal matrix with d blocks equal to the velocity mass matrix defined in (2.56). The mass matrices are needed for correct scaling, see, e.g., [40]. This results in the discrete version of the commutator in the form

$$\mathbf{E} = (\mathbf{M}_u^{-1}\mathbf{F})(\mathbf{M}_u^{-1}\mathbf{B}^T) - (\mathbf{M}_u^{-1}\mathbf{B}^T)(\mathbf{M}_p^{-1}\mathbf{F}_p). \tag{5.33}$$

The assumption that $\mathbf{E}$ is small yields that

$$(\mathbf{M}_u^{-1}\mathbf{F})(\mathbf{M}_u^{-1}\mathbf{B}^T) \approx (\mathbf{M}_u^{-1}\mathbf{B}^T)(\mathbf{M}_p^{-1}\mathbf{F}_p). \tag{5.34}$$

After several algebraic manipulations, this leads to the following approximation of the Schur complement

$$-\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T \approx -\mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T\mathbf{F}_p^{-1}\mathbf{M}_p. \tag{5.35}$$

The matrix $\mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T$ would be expensive to compute and work with, since it is dense. However, it can be viewed as a discrete Laplacian operator (acting on pressures) and thus replacing it by the matrix $\mathbf{A}_p$ considered above seems like a suitable workaround (see [96] for more details on spectral equivalence of these matrices). This leads to the Schur complement approximation (5.30).

The PCD preconditioner can be derived analogously for time-dependent problems. Assuming the backward Euler time discretization as in Section 2.2.2, the only difference is that the continuous operator represented by the matrices $\mathbf{F}$ and $\mathbf{F}_p$ is $(\frac{1}{\Delta t} - \nu\Delta + \boldsymbol{w} \cdot \nabla)$ and thus the matrix $\mathbf{F}_p$ involves the term $\frac{1}{\Delta t}\mathbf{M}_p$. The time-dependent problem is considered in [96].

The application of $\widehat{\mathbf{S}}_{\mathrm{PCD}}^{-1}$ to a vector requires solving one pressure-Poisson linear system, one linear system with the pressure mass matrix and a matrix-vector multiplication with the matrix $\mathbf{F}_p$. Thus, it is less expensive than the application of the BFBt approximation.

The obvious disadvantage of this preconditioner is the necessity to assemble the matrices $\mathbf{A}_p$ and $\mathbf{F}_p$, representing the discrete operators on the pressure space, that are usually not available in standard CFD codes. Moreover, we need to define some boundary conditions for these operators. Both derivations of the preconditioner described above are based on the assumption that $\Omega = \mathbb{R}^d$ and do not take into account any effect of boundary conditions. The choice is therefore not clear, but at the same time a poor choice can affect the performance of the preconditioner quite significantly. The original paper [68] does not address the issue of boundary conditions at all. The article [96] only mentions the case of enclosed flows, where the matrices $\mathbf{A}_p$ and $\mathbf{F}_p$ should represent the Laplace and convection–diffusion operator, respectively, with Neumann boundary conditions on the whole boundary $\partial\Omega$. It means that the elements of the matrices are simply given by the following formulas

$$\mathbf{A}_p = [A_{p,ij}] = \left[\int_\Omega \nabla\varphi_i^p \cdot \nabla\varphi_j^p\right], \tag{5.36}$$

$$\mathbf{F}_p = [F_{p,ij}] = \left[\nu\int_\Omega \nabla\varphi_i^p \cdot \nabla\varphi_j^p + \int_\Omega \varphi_i^p\big(\boldsymbol{w} \cdot \nabla\varphi_j^p\big)\right] \tag{5.37}$$

for stationary problems, or

$$\mathbf{F}_p = [F_{p,ij}] = \left[\frac{1}{\Delta t}\int_\Omega \varphi_i^p\varphi_j^p + \nu\int_\Omega \nabla\varphi_i^p \cdot \nabla\varphi_j^p + \int_\Omega \varphi_i^p\big(\boldsymbol{w} \cdot \nabla\varphi_j^p\big)\right] \tag{5.38}$$

for time-dependent problems, where $\boldsymbol{w} = \boldsymbol{u}_h^k$ is the discrete velocity from the most recent nonlinear iteration and $\{\varphi_i^p\}$ denote the pressure basis functions as in Section 2.2.

PCD boundary conditions

For inflow-outflow problems, the domain boundary $\partial\Omega$ can be divided into three parts depending on the velocity direction: the inflow part $\partial\Omega_{\mathrm{in}}$, the outflow part $\partial\Omega_{\mathrm{out}}$ and the characteristic part $\partial\Omega_{\mathrm{char}}$, that are defined as follows

$$\begin{aligned} \partial\Omega_{\mathrm{in}} &= \{\boldsymbol{x} \in \partial\Omega \,|\, \boldsymbol{u} \cdot \boldsymbol{n} < 0\}, \\ \partial\Omega_{\mathrm{out}} &= \{\boldsymbol{x} \in \partial\Omega \,|\, \boldsymbol{u} \cdot \boldsymbol{n} > 0\}, \\ \partial\Omega_{\mathrm{char}} &= \{\boldsymbol{x} \in \partial\Omega \,|\, \boldsymbol{u} \cdot \boldsymbol{n} = 0\}, \end{aligned} \tag{5.39}$$

where $\boldsymbol{n}$ is the outward-pointing unit normal to the boundary at the point $\boldsymbol{x}$. Usually, it is assumed that $\partial\Omega_N = \partial\Omega_{\text{out}}$ and $\partial\Omega_D = \partial\Omega_{\text{in}} \cup \partial\Omega_{\text{char}}$, where $\partial\Omega_D$ is the Dirichlet part of the boundary and $\partial\Omega_N$ is the part where the do-nothing condition is prescribed (see Chapter 2). The important case of inflow-outflow problems in connection with the PCD preconditioner is addressed, e.g., in the first edition of the monograph by Elman, Silvester, Wathen [39]. They suggest using Dirichlet boundary conditions along $\partial\Omega_{\text{in}}$ and Neumann conditions elsewhere for the construction of the discrete operators on the pressure space. Practically, this is done by assembling the matrices $\mathbf{A}_p, \mathbf{F}_p$ according to (5.36) and (5.37) (or (5.38)) and then modifying the rows and columns corresponding to the pressure basis functions that are nonzero on $\partial\Omega_{\text{in}}$ such that they contain only diagonal entries. We refer to PCD with this choice of boundary conditions as the original PCD preconditioner and we denote the Schur complement approximation as

$$\widehat{\mathbf{S}}_{\text{PCD}}^{\text{orig}} = -\mathbf{A}_p^{D\text{in}} \left(\mathbf{F}_p^{D\text{in}}\right)^{-1} \mathbf{M}_p, \tag{5.40}$$

where $\mathbf{A}_p^{D\text{in}}, \mathbf{F}_p^{D\text{in}}$ denote the matrices resulting from the boundary modification of $\mathbf{A}_p, \mathbf{F}_p$ described above. As pointed out in [39], the Dirichlet boundary conditions only affect the definition of the algebraic operators, thus no boundary conditions are actually imposed on the discrete pressures, there are no specific Dirichlet values to determine and no right-hand side is affected. This also means that the diagonal values in the modified rows are arbitrary and their choice can affect the quality of the preconditioner, see [41] where this issue is addressed, although on a different part of the boundary.

Elman and Tuminaro [41] suggest a slightly different approach to the PCD preconditioner and derive suitable boundary conditions for their formulation. Their starting point is the commutator of the convection–diffusion operators with the divergence operator

$$\mathcal{E}_{\text{mod}} = \nabla \cdot \mathcal{F}_u - \mathcal{F}_p \nabla \cdot \tag{5.41}$$

instead of the commutator $\mathcal{E}$ with the gradient operator (5.32). They show the advantage of (5.41) over (5.32) for a one-dimensional problem discretized using finite differences, where it is not possible to make the discretized commutator $\mathcal{E}$ zero on both inflow and outflow boundary. The discrete commutator based on (5.41) leads to the Schur complement approximation in the form

$$\widehat{\mathbf{S}}_{\text{PCD}}^{\text{mod}} = -\mathbf{M}_p \mathbf{F}_p^{-1} \mathbf{A}_p, \tag{5.42}$$

which differs from (5.30) only in the order of matrices. Moreover, instead of assembling the pressure-Poisson matrix $\mathbf{A}_p$ as in (5.36), they suggest using $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$, where $\widehat{\mathbf{M}}_u$ is a diagonal approximation of the velocity mass matrix. Thus, the only new matrix that needs to be assembled is $\mathbf{F}_p$ and no artificial boundary conditions for $\mathbf{A}_p$ are required. The boundary conditions defined implicitly by taking $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$ correspond to Dirichlet boundary conditions on $\partial\Omega_{\text{out}}$, see [40, Chapter 9] for explanation. According to [41], $\mathbf{F}_p$ should be defined with a Robin condition

$$-\nu \frac{\partial p}{\partial \boldsymbol{n}} + (\boldsymbol{w} \cdot \boldsymbol{n}) p = 0 \tag{5.43}$$

on the part of the boundary where the Navier–Stokes equations are posed with Dirichlet conditions of inflow or characteristic type. For the characteristic part of the boundary, (5.43) reduces to a Neumann condition and the new approach corresponds to the original

one. Thus, the new boundary condition is used only on the inflow part of the boundary. This leads to the pressure convection–diffusion matrix $\mathbf{F}_p^{R\text{in}}$,

$$\mathbf{F}_p^{R\text{in}} = [F_{p,ij}^{R\text{in}}] = F_{p,ij} - \int_{\partial\Omega_{\text{in}}} (\boldsymbol{w}\cdot\boldsymbol{n})\varphi_i^p\varphi_j^p, \tag{5.44}$$

which is used instead of $\mathbf{F}_p$ in $\widehat{\mathbf{S}}_{\text{PCD}}^{\text{mod}}$ (5.42). The definition (5.44) corresponds to Neumann boundary conditions on $\partial\Omega \setminus \partial\Omega_{\text{in}}$. Dirichlet boundary condition can be also used on the outflow boundary, which is done by modifying rows and columns of $\mathbf{F}_p^{R\text{in}}$ similarly as described above for $\partial\Omega_{\text{in}}$; denote the resulting matrix as $\mathbf{F}_p^{R\text{in},D\text{out}}$.

We refer to the dissertation of Blechta [9] for a theoretical analysis the PCD preconditioner with emphasis on the proper choice of boundary conditions. The analysis is based on the idea of operator preconditioning in infinite-dimensional spaces and thus independent of any particular discretization. Both variants of the commutator, $\mathcal{E}$ and $\mathcal{E}_{\text{mod}}$, are considered. Also, the construction of the discrete preconditioner and incorporation of the boundary conditions is described in a general setting as well as for some specific discretizations, leading to several variants of the PCD Schur complement approximation, $\widehat{\mathbf{S}}_{\text{PCD}}^{\text{orig}}$ being one of them. The analysis in [9] indicates that the variants based on $\mathcal{E}_{\text{mod}}$ are more robust and thus preferable. In our setting (with continuous pressure discretization), the discrete Schur complement approximation takes the form

$$\widehat{\mathbf{S}}_{\text{PCD}}^{R} = -\mathbf{M}_p \left(\mathbf{F}_p^{R\text{in},D\text{out}}\right)^{-1} \mathbf{A}_p^{D\text{out}}, \tag{5.45}$$

where $\mathbf{A}_p^{D\text{out}}$ is the discrete pressure-Poisson operator (5.36) with a boundary modification corresponding to Dirichlet conditions on the outflow boundary. This is very close to the variant of Elman and Tuminaro.

PCD properties

An analysis of the PCD preconditioner was presented by Loghin in [72]. He shows, assuming quasi-uniform finite element discretizations, that the spectrum of the preconditioned matrix is bounded independently of the mesh parameter. He also mentions that mesh-independent ε-pseudospectrum can be assumed, which results in mesh-independent convergence of GMRES. Moreover, he demonstrates that the GMRES iteration counts grow proportionally to $\nu^{-1/2}$.

LSC preconditioner

The LSC preconditioner was proposed in [35] as an alternative to PCD in order to circumvent the difficulty of assembling new matrices on the pressure space. Similarly to PCD, the derivation of LSC is based on the idea of commutators, however, the discrete convection–diffusion operator on the pressure space is constructed purely algebraically, using only the available matrices obtained from the discretization of the Navier–Stokes equations.

The starting point of the derivation is the requirement that the discrete commutator $\mathbf{E}$ defined in (5.33) is small, i.e.

$$\mathbf{M}_u^{-1}\mathbf{F}\mathbf{M}_u^{-1}\mathbf{B}^T - \mathbf{M}_u^{-1}\mathbf{B}^T\mathbf{M}_p^{-1}\mathbf{F}_p \approx \mathbf{0}, \tag{5.46}$$

which leads to the Schur complement approximation (5.35), where $\mathbf{F}_p \in \mathbb{R}^{n_p \times n_p}$ is now an unknown matrix to be determined. This can be reformulated such that the columns $\mathbf{f}_{p,j}$ of $\mathbf{F}_p$ are approximate solutions of n_p linear systems of the form

$$\mathbf{M}_u^{-1}\mathbf{B}^T\mathbf{M}_p^{-1}\mathbf{f}_{p,j} = \mathbf{b}_j, \quad j = 1, \ldots, n_p, \tag{5.47}$$

where $\mathbf{b}_j$ is the j-th column of $\mathbf{M}_u^{-1}\mathbf{F}\mathbf{M}_u^{-1}\mathbf{B}^T$. These are overdetermined linear systems for which approximate solution can be obtained, e.g., by solving a least-squares problem. That is, the vectors $\mathbf{f}_{p,j}$ are determined as solutions of the following optimization problems

$$\min_{\mathbf{f}_{p,j}} ||\mathbf{M}_u^{-1}\mathbf{B}^T\mathbf{M}_p^{-1}\mathbf{f}_{p,j} - \mathbf{b}_j||^2. \tag{5.48}$$

It can be beneficial to replace the Euclidean norm in (5.48) by some other norm suitable for the given problem. In our case, a suitable norm is $||\mathbf{x}||_{\mathbf{M}_u} = \sqrt{\mathbf{x}^T\mathbf{M}_u\mathbf{x}}$, which is a discrete analogue of the continuous L^2 norm on the velocity space. This leads to the weighted least-squares problem

$$\min_{\mathbf{f}_{p,j}} ||\mathbf{M}_u^{-1}\mathbf{B}^T\mathbf{M}_p^{-1}\mathbf{f}_{p,j} - \mathbf{b}_j||^2_{\mathbf{M}_u}. \tag{5.49}$$

The normal equations associated with this problem are

$$\mathbf{M}_p^{-1}\mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T\mathbf{M}_p^{-1}\mathbf{f}_{p,j} = \mathbf{M}_p^{-1}\mathbf{B}\mathbf{b}_j, \tag{5.50}$$

thus, the matrix $\mathbf{F}_p$ is obtained as follows

$$\begin{aligned} \mathbf{M}_p^{-1}\mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T\mathbf{M}_p^{-1}\mathbf{F}_p &= \mathbf{M}_p^{-1}\mathbf{B}\mathbf{M}_u^{-1}\mathbf{F}\mathbf{M}_u^{-1}\mathbf{B}^T, \\ \mathbf{F}_p &= \mathbf{M}_p(\mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T)^{-1}\mathbf{B}\mathbf{M}_u^{-1}\mathbf{F}\mathbf{M}_u^{-1}\mathbf{B}^T. \end{aligned} \tag{5.51}$$

By substituting this formula into (5.35), we get the Schur complement approximation

$$-\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T \approx -(\mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T)(\mathbf{B}\mathbf{M}_u^{-1}\mathbf{F}\mathbf{M}_u^{-1}\mathbf{B}^T)^{-1}(\mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T). \tag{5.52}$$

Note that if the scaling by the mass matrices was omitted in (5.33) and the non-weighted least-squares problem was considered, the procedure described above would lead to the BFBt Schur complement approximation $\widehat{\mathbf{S}}_{\mathrm{BFBt}}$ (5.27).

The application of the inverse of (5.52) would be still too expensive. Therefore, the formula is further approximated by replacing the velocity mass matrix $\mathbf{M}_u$ by a diagonal approximation $\widehat{\mathbf{M}}_u$, e.g., $\widehat{\mathbf{M}}_u = \mathrm{diag}(\mathbf{M}_u)$. This leads to the LSC Schur complement approximation

$$\widehat{\mathbf{S}}_{\mathrm{LSC}} = -(\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T)(\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{F}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T)^{-1}(\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T). \tag{5.53}$$

The LSC preconditioner is called scaled BFBt in the original paper [35].

The application of $\widehat{\mathbf{S}}_{\mathrm{LSC}}^{-1}$ to a vector requires solving two Poisson-type problems on the pressure space, matrix-vector multiplications with the matrices $\mathbf{B}^T$, $\mathbf{F}$ and $\mathbf{B}$ and two scalings with the diagonal matrix $\widehat{\mathbf{M}}_u^{-1}$.

LSC boundary conditions

Unlike PCD, there is no need to make an explicit choice of boundary conditions for the LSC preconditioner. Boundary conditions of the underlying operators on the pressure space are defined implicitly. However, Elman and Tuminaro demonstrate in [41] that a boundary modification can improve convergence also for the LSC preconditioner. They introduce a new scaling in the Schur complement approximation,

$$\widehat{\mathbf{S}}_{\mathrm{LSC}}^{\mathrm{mod}} = -(\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T)(\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{F}\mathbf{H}\mathbf{B}^T)^{-1}(\mathbf{B}\mathbf{H}\mathbf{B}^T), \tag{5.54}$$

where $\mathbf{H} = \mathbf{W}^{\frac{1}{2}}\widehat{\mathbf{M}}_u^{-1}\mathbf{W}^{\frac{1}{2}}$ and $\mathbf{W}$ is a diagonal weighting matrix. The definition of $\mathbf{W}$ in [41] is limited to the case when the domain boundaries are aligned with the coordinate axes and it is not clear how to choose its entries in a general case. Therefore, we do not consider this modification in this work.

LSC properties

Regarding dependence of the convergence on the problem parameters h and ν, a similar behavior as for BFBt was observed for LSC in [35]. It means that its convergence depends on both parameters. However, the iteration counts are significantly lower than for the original BFBt preconditioner.

5.3.3 Augmented Lagrangian approach

The Augmented Lagrangian (AL) approach to preconditioning proposed by Benzi and Olshanskii [6] belongs, in fact, to the group of block triangular preconditioners, but it is not based on the form (5.14) and finding an approximation to the Schur complement $\mathbf{S} = -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T$. Instead, the original saddle-point linear system (4.1) is replaced by an equivalent linear system for which the Schur complement can be approximated more easily. This approach is inspired by the methods used in constrained optimization, where saddle-point problems arise as the first order optimality conditions.

Constrained optimization

Let us briefly explain the context on the example of quadratic programming. Consider a symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, a vector $\mathbf{f} \in \mathbb{R}^n$ and the associated quadratic functional $J : \mathbb{R}^n \to \mathbb{R}$,

$$J(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{A}\mathbf{u} - \mathbf{f}^T\mathbf{u}. \tag{5.55}$$

Further consider the following minimization problem

$$\min_{\mathbf{u}\in\Omega} J(\mathbf{u}), \quad \Omega = \{\mathbf{u} \in \mathbb{R}^n : \mathbf{B}\mathbf{u} = \mathbf{g}\}, \tag{5.56}$$

where $\mathbf{B}$ is a $m \times n$ matrix with $m \leq n$. One of the classical methods for finding the minimizer of the constrained problem (5.56) is to define a penalized functional,

$$J_\gamma(\mathbf{u}) = J(\mathbf{u}) + \frac{\gamma}{2}||\mathbf{B}\mathbf{u} - \mathbf{g}||^2, \tag{5.57}$$

and solve the unconstrained optimization problem

$$\min_{\mathbf{u}\in\mathbb{R}^n} J_\gamma(\mathbf{u}). \tag{5.58}$$

Its solution $\mathbf{u}^*(\gamma)$ can be found by solving $\nabla J_\gamma(\mathbf{u}) = \mathbf{0}$ which leads to the linear system

$$(\mathbf{A} + \gamma \mathbf{B}^T \mathbf{B})\mathbf{u} = \mathbf{f} + \gamma \mathbf{B}^T \mathbf{g}. \tag{5.59}$$

It can be shown that $\mathbf{u}^*(\gamma)$ tends to the solution $\mathbf{u}^*$ of (5.56) for $\gamma \to +\infty$. Thus, $\mathbf{u}^*(\gamma)$ will be a good approximation of $\mathbf{u}^*$ for a sufficiently large parameter γ. However, it is difficult to obtain an accurate solution of (5.59) for large γ, since the coefficient matrix is dominated by the term $\gamma \mathbf{B}^T \mathbf{B}$, which is singular, resulting in a very large condition number of the matrix $\mathbf{A} + \gamma \mathbf{B}^T \mathbf{B}$.

Another common approach to solving the problem (5.56) is to introduce a vector of so called Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m$ and define the Lagrangian $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$,

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = J(\mathbf{u}) + \boldsymbol{\lambda}^T (\mathbf{B}\mathbf{u} - \mathbf{g}). \tag{5.60}$$

It can be shown that $\mathcal{L}$ has at least one saddle-point and all the saddle-points are in the form $(\mathbf{u}^*, \boldsymbol{\lambda})$. Provided that $\mathbf{B}$ has full rank, there is exactly one saddle point. For the quadratic programming problem, all stationary points of the Lagrangian correspond to saddle-points, therefore all saddle-points can be found by solving $\nabla \mathcal{L} = \mathbf{0}$, leading to the linear system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}. \tag{5.61}$$

In practical computations, these saddle-point linear systems arising from constrained optimization problems can be solved by the methods described in Chapter 4 of this work or other methods mentioned, for example, in [5].

Note that discretization of the Stokes problem leads exactly to a linear system of the form (5.61). Thus, solving the Stokes problem corresponds to minimizing a quadratic functional with the pressure playing the role of a Lagrange multiplier.

The augmented Lagrangian method combines both approaches described above, the penalty method and the Lagrange multipliers. The augmented Lagrangian is defined as

$$\mathcal{L}_\gamma(\mathbf{u}, \boldsymbol{\lambda}) = J(\mathbf{u}) + \boldsymbol{\lambda}^T (\mathbf{B}\mathbf{u} - \mathbf{g}) + \frac{\gamma}{2} ||\mathbf{B}\mathbf{u} - \mathbf{g}||^2. \tag{5.62}$$

Any saddle-point of $\mathcal{L}_\gamma$ is a saddle-point of $\mathcal{L}$ and vice versa, because the penalty term in $\mathcal{L}_\gamma$ vanishes when the constraint $\mathbf{B}\mathbf{u} = \mathbf{g}$ is satisfied. Finding the saddle-points of the augmented Lagrangian $\mathcal{L}_\gamma$ requires solving the following linear system

$$\begin{bmatrix} \mathbf{A} + \gamma \mathbf{B}^T \mathbf{B} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} + \gamma \mathbf{B}^T \mathbf{g} \\ \mathbf{g} \end{bmatrix}. \tag{5.63}$$

The $(1,1)$ block of the coefficient matrix is the same as the matrix in (5.59), however, unlike the penalty method, the problems (5.63) and (5.61) are equivalent for any value of γ. Thus, the need for large γ (and consequently the large condition number of the matrix $\mathbf{A} + \gamma \mathbf{B}^T \mathbf{B}$) is avoided. At the same time, the penalty term improves the convergence of some methods for solving the saddle-point linear system (5.63). More on the AL methods and their applications can be found, for example, in [48].

The AL preconditioner

The AL approach to preconditioning is based on replacing the original system (4.1) by the augmented linear system

$$\begin{bmatrix} \mathbf{F}_\gamma & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_\gamma \\ \mathbf{g} \end{bmatrix}, \tag{5.64}$$

where $\mathbf{F}_\gamma = \mathbf{F} + \gamma \mathbf{B}^T \mathbf{W}^{-1} \mathbf{B}$, $\mathbf{f}_\gamma = \mathbf{f} + \gamma \mathbf{B}^T \mathbf{W}^{-1} \mathbf{g}$, $\gamma > 0$ and $\mathbf{W}$ is a positive definite matrix. The incorporation of the matrix $\mathbf{W}^{-1}$ corresponds to using the norm $||.||_{\mathbf{W}^{-1}}$ instead of the Euclidean norm for the penalty term in (5.62).

Recall that we denoted the saddle-point matrix in (4.1) as $\boldsymbol{\mathcal{A}}$ and denote the augmented matrix in (5.64) as $\boldsymbol{\mathcal{A}}_\gamma$. It can be shown that

$$\boldsymbol{\mathcal{A}}_\gamma^{-1} = \boldsymbol{\mathcal{A}}^{-1} - \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \gamma \mathbf{W}^{-1} \end{bmatrix}, \tag{5.65}$$

see Golub and Greif [53]. Provided that the block $\mathbf{F}$ is invertible, it can be easily derived that the $(2,2)$ block of $\boldsymbol{\mathcal{A}}^{-1}$ is $\mathbf{S}^{-1}$ (see the block LDU decomposition of $\boldsymbol{\mathcal{A}}$ (5.10)). Similarly, the $(2,2)$ block of $\boldsymbol{\mathcal{A}}_\gamma^{-1}$ is $\mathbf{S}_\gamma^{-1}$, where $\mathbf{S}_\gamma = -\mathbf{B}\mathbf{F}_\gamma^{-1}\mathbf{B}^T$. This, combined with (5.65), implies the following relation between the two inverse Schur complements

$$\mathbf{S}_\gamma^{-1} = \mathbf{S}^{-1} - \gamma \mathbf{W}^{-1}. \tag{5.66}$$

Benzi and Olshanskii [6] propose to precondition the augmented system (5.64) with the block triangular preconditioner

$$\widehat{\boldsymbol{\mathcal{M}}}_{AL} = \begin{bmatrix} \widehat{\mathbf{F}}_\gamma & \mathbf{B}^T \\ \mathbf{0} & \widehat{\mathbf{S}}_\gamma \end{bmatrix}, \tag{5.67}$$

where the action of $\widehat{\mathbf{F}}_\gamma^{-1}$ is computed as an approximate solution to a linear system with the matrix $\mathbf{F}_\gamma$. The choice of the Schur complement approximation $\widehat{\mathbf{S}}_\gamma$ is based on the relation (5.66). The original Schur complement $\mathbf{S}$ is approximated by $-\frac{1}{\nu}\widehat{\mathbf{M}}_p$, where $\widehat{\mathbf{M}}_p$ is an approximation of the pressure mass matrix, which is related to the preconditioner $\widehat{\mathbf{S}}_M$ mentioned earlier, that is known to give h-independent convergence for the original problem. Thus, $\widehat{\mathbf{S}}_\gamma$ is given through its inverse as

$$\widehat{\mathbf{S}}_\gamma^{-1} = -\nu \widehat{\mathbf{M}}_p^{-1} - \gamma \mathbf{W}^{-1}, \tag{5.68}$$

The matrix $\mathbf{W}$ is often chosen to be equal to $\widehat{\mathbf{M}}_p$.

If $\widehat{\mathbf{F}}_\gamma = \mathbf{F}_\gamma$ and we assume that the subsystem with this matrix is solved exactly, we talk about the ideal version of the AL preconditioner. In practice, direct solution od this linear system is mostly unfeasible, since the additional term $\gamma \mathbf{B}^T \mathbf{W}^{-1} \mathbf{B}$ makes the matrix denser compared to the block $\mathbf{F}$ and, moreover, introduces a coupling between velocity components, which is not present in the discretization of the Picard linearization of the Navier–Stokes equations. Finding a suitable $\widehat{\mathbf{F}}_\gamma$, i.e. a robust and efficient approximate solver for linear systems with $\mathbf{F}_\gamma$, is also not trivial. See [6] for a specialized multigrid method in two dimensions, which was later generalized to three dimensions by Farrell at al. in [44]. However, these specialized multigrid methods are strongly tied to the discretization and, to our knowledge, currently limited to some particular FEM discretizations.

The modified AL preconditioner

One way to simplify the solution of the linear systems with $\mathbf{F}_\gamma$ is the modified version of the AL preconditioner (MAL) introduced by Benzi et al. in [8]. Let us denote the blocks of $\boldsymbol{F}_\gamma$ corresponding to the velocity components in two dimensions as follows

$$\mathbf{F}_\gamma = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{bmatrix}. \tag{5.69}$$

Benzi et al. suggest replacing this block by its block upper triangular part

$$\widetilde{\mathbf{F}}_\gamma =: \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{0} & \mathbf{F}_{22} \end{bmatrix}, \tag{5.70}$$

such that instead of solving the whole system at once, we solve two smaller systems with the blocks $\mathbf{F}_{11}$ and $\mathbf{F}_{22}$. These blocks can be interpreted as discrete anisotropic convection–diffusion operators, thus, applying $\widetilde{\mathbf{F}}_\gamma^{-1}$ requires solving two anisotropic convection–diffusion problems. The situation is similar in three dimensions, where we have to solve three subsystems.

AL and MAL properties

Benzi and Olshanskii show in [6] that for $\widehat{\mathbf{F}}_\gamma = \mathbf{F}_\gamma$ and $\mathbf{W} = \widehat{\mathbf{M}}_p = \mathbf{M}_p$ the eigenvalues of the preconditioned system are bounded independently of h and they tend to 1 for $\gamma \to +\infty$. This also holds if the mass matrix is approximated by a spectrally equivalent matrix $\widehat{\mathbf{M}}_p$. According to their analysis, it is sufficient to set $\gamma = O(\nu^{-1})$ to achieve eigenvalue bounds independent of ν. However, numerical experiments indicate, it is not necessary to use large values of γ in practice, even for problems with low viscosity, since much lower values of γ already lead to convergence independent of both h and ν. Specifically, they suggest to set $\gamma \approx ||\boldsymbol{w}||$, where $\boldsymbol{w}$ is the convection velocity. We refer to [31] for an experimental study, where we investigated the choice of γ for IgA discretizations.

Later, Benzi and Olshanskii presented a field-of-values analysis of the AL and MAL preconditioners [7]. They consider inf-sup stable finite element discretizations with shape-regular meshes. They prove that GMRES with the ideal AL preconditioner converges independently of h and ν. For the MAL preconditioner, they establish h-independent convergence of GMRES.

5.3.4 SIMPLE-type preconditioners

The SIMPLE algorithm was described as a stationary iterative method for solving the saddle-point linear system (4.1) in Section 4.2.3. As described in Section 5.2, any consistent stationary method can be accelerated by a Krylov subspace method, i.e., used as a preconditioner. It was first proposed to use variants of the SIMPLE method as preconditioners for Krylov subspace methods by Vuik et al. in [113, 112]. We also refer to [107, 69] for more on SIMPLE-type preconditioners. In this section, we will describe three preconditioners from this group: SIMPLE, SIMPLER and MSIMPLER.

The SIMPLE preconditioner

Recall that the SIMPLE algorithm can be written in the form of a stationary iteration based on the splitting $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{B}}^{-1} - \boldsymbol{\mathcal{N}}\boldsymbol{\mathcal{B}}^{-1}$ (see (4.57) and its derivation), leading to the iteration formula

$$\begin{bmatrix} \mathbf{u}_{n+1} \\ \mathbf{p}_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_n \\ \mathbf{p}_n \end{bmatrix} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{M}}^{-1} \begin{bmatrix} \mathbf{r}_{u,n} \\ \mathbf{r}_{p,n} \end{bmatrix}, \tag{5.71}$$

where $[\mathbf{r}_{u,n}, \mathbf{r}_{p,n}]^T$ is the n-th residual vector and the matrices $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{B}}$ are defined as

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & \widehat{\mathbf{S}}_D \end{bmatrix}, \quad \boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{I}_u & -\mathbf{D}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix}, \tag{5.72}$$

with $\widehat{\mathbf{S}}_D = -\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T$ and $\mathbf{D} = \mathrm{diag}(\mathbf{F})$.

Thus, in accordance with Section 5.2, Krylov acceleration of the SIMPLE method corresponds to using the preconditioner

$$\boldsymbol{\mathcal{M}}_{\mathrm{SIMPLE}} = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{B}}^{-1} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & \widehat{\mathbf{S}}_D \end{bmatrix} \begin{bmatrix} \mathbf{I}_u & \mathbf{D}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix}. \tag{5.73}$$

This preconditioner can be also interpreted as an approximation of the matrix $\boldsymbol{\mathcal{A}}$ based on its block LU decomposition (5.12), where both occurrences of $\mathbf{F}^{-1}$ are approximated by $\mathbf{D}^{-1}$.

The application of the preconditioner to a vector $\mathbf{r} = [\mathbf{r}_u, \mathbf{r}_p]^T$ is performed by solving the linear system $\boldsymbol{\mathcal{M}}_{\mathrm{SIMPLE}}\,\mathbf{z} = \mathbf{r}$. The individual steps are summarized in Algorithm 9, which corresponds to one step of the SIMPLE method described in Algorithm 5 with $\mathbf{p}_n = \mathbf{0}$.

Algorithm 9: Application of $\boldsymbol{\mathcal{M}}^{-1}_{\mathrm{SIMPLE}}$

1 Solve $\mathbf{F}\mathbf{z}^*_u = \mathbf{r}_u$.
2 Solve $\widehat{\mathbf{S}}_D\mathbf{z}_p = \mathbf{r}_p - \mathbf{B}\mathbf{z}^*_u$.
3 Update $\mathbf{z}_u = \mathbf{z}^*_u - \mathbf{D}^{-1}\mathbf{B}^T\mathbf{z}_p$.

The SIMPLE preconditioner is relatively cheap per iteration, since it requires only one velocity solve and one pressure solve. However, the convergence of the Krylov subspace method with the SIMPLE preconditioner is usually quite slow. The variants described below require one additional pressure solve, but they generally give much faster convergence than SIMPLE.

The SIMPLER preconditioner

The explicit matrix form of the SIMPLER preconditioner can be derived similarly to SIMPLE. Based on the iteration formula (4.71), the preconditioner takes the form

$$\boldsymbol{\mathcal{M}}_{\mathrm{SIMPLER}} = \boldsymbol{\mathcal{Q}}^{-1}, \tag{5.74}$$

where the matrix $\boldsymbol{\mathcal{Q}}$ is defined in (4.72). Algorithm 10 describes the individual steps of the application of the preconditioner to a vector $\mathbf{r}$. Again, this corresponds to one iteration of the SIMPLER method with $\mathbf{u}_n = \mathbf{0}$ in (4.65).

Algorithm 10: Application of $\boldsymbol{\mathcal{M}}^{-1}_{\mathrm{SIMPLER}}$

1 Solve $\widehat{\mathbf{S}}_D\mathbf{z}^*_p = \mathbf{r}_p - \mathbf{B}\mathbf{D}^{-1}\mathbf{r}_u$.
2 Solve $\mathbf{F}\mathbf{z}^*_u = \mathbf{r}_u - \mathbf{B}^T\mathbf{z}^*_p$.
3 Solve $\widehat{\mathbf{S}}_D\delta\mathbf{z}_p = \mathbf{r}_p - \mathbf{B}\mathbf{z}^*_u$.
4 Update $\mathbf{z}_u = \mathbf{z}^*_u - \mathbf{D}^{-1}\mathbf{B}^T\delta\mathbf{z}_p$.
5 Update $\mathbf{z}_p = \mathbf{z}^*_p + \delta\mathbf{z}_p$.

The MSIMPLER preconditioner

The MSIMPLER variant is a modification of the SIMPLER preconditioner proposed in [107]. The modification consists in replacing the diagonal matrix $\mathbf{D}$ by a diagonal approximation of the velocity mass matrix $\widehat{\mathbf{M}}_u$, leading to Algorithm 11, where $\widetilde{\mathbf{S}} = -\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$.

Algorithm 11: Application of $\boldsymbol{\mathcal{M}}^{-1}_{\text{MSIMPLER}}$

1 Solve $\widetilde{\mathbf{S}}\mathbf{z}_p^* = \mathbf{r}_p - \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{r}_u$.
2 Solve $\mathbf{F}\mathbf{z}_u^* = \mathbf{r}_u - \mathbf{B}^T\mathbf{z}_p^*$.
3 Solve $\widetilde{\mathbf{S}}\delta\mathbf{z}_p = \mathbf{r}_p - \mathbf{B}\mathbf{z}_u^*$.
4 Update $\mathbf{z}_u = \mathbf{z}_u^* - \widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T\delta\mathbf{z}_p$.
5 Update $\mathbf{z}_p = \mathbf{z}_p^* + \delta\mathbf{z}_p$.

The advantage of MSIMPLER over SIMPLER is that the Schur complement approximation $\widetilde{\mathbf{S}}$ is does not depend on the convection velocity, i.e., it does not change during the nonlinear or time stepping iteration. Thus its construction and solver setup for the linear systems with $\widetilde{\mathbf{S}}$ are performed only once.

Properties of SIMPLE-type preconditioners

The convergence of all SIMPLE-type preconditioners depends on both h an ν. From the results in the literature, it can be estimated that the growth of iteration counts is between $O(h^{-1/2})$ and $O(h^{-1})$ and also between $O(\nu^{-1/2})$ and $O(\nu^{-1})$, see, for example, [56, 69, 94, 106, 107]. The iteration counts for the SIMPLE preconditioner is always significantly higher than for the other two mentioned SIMPLE-type preconditioners, SIMPLER and MSIMPLER.

5.3.5 Solution of subproblems

As mentioned earlier in this section, efficient solution of the subsystems is an important part of the block triangular preconditioners. Of course, practical use of the ideal versions, i.e. direct solution of the subproblems, would be too expensive or even unfeasible in the case of three-dimensional problems. Thus, identifying suitable approximate solvers for these subsystems should be the next step towards an efficient preconditioner.

Ideally, replacing the direct inner solvers by approximate methods should not lead to a significant change in the convergence properties of the preconditioner. Thus, optimality in the sense that their convergence is independent of the mesh parameter h is a reasonable requirement. Therefore, some kind of multigrid (geometric or algebraic) is often the method of choice. Suitable multigrid methods for Poisson and convection–diffusion problems can be found in the literature, see, e.g., Elman, Silvester, Wathen [40] for an overview of efficient solution methods for finite element discretizations of flow problems.

However, many authors in recent years have observed that the performance of classical multigrid methods applied to IgA linear systems is highly dependent on the B-spline degree and the spatial dimension. Development of multigrid methods for IgA discretizations that would be robust with respect to these parameters has been quite an active research area recently. Several promising approaches have been already proposed that are based on different ideas, usually exploiting the tensor-product structure of the spline

spaces, see, e.g., Donatelli et al. [28], Hofreither and Takacs [61], Hofer and Takacs [60] or Riva et al. [26]. Alternatively, p-multigrid, where "coarsening" in the spline degree is performed, is considered by Tielen et al. [99]. These works are mainly focused on the Poisson problem, but some of them consider a more general convection–diffusion-reaction equation [28, 99].

Besides multigrid, one can also use a small number of iterations of a preconditioned iterative solver for the subsystems. There are some robust preconditioners developed specifically for IgA discretizations of the Poisson problem, see, for example, Harbrecht et al. [14] or Sangalli and Tani [90].

We will not go into further details of these methods, since we consider only the ideal versions of the block preconditioners in the numerical section of this work. We only have some preliminary results with several approximate inner solvers, but more work still has to be done in this direction.

Chapter 6

Numerical results

In the literature dealing with block preconditioners for the saddle-point linear systems, we often find comparisons of some subset of the preconditioners described in this work, usually for the most common low-degree inf-sup stable and stabilized finite element pairs (P2-P1, Q2-Q1, Q1-P0, P1-P1, Q1-Q1) [8, 36, 37, 56, 79, 94, 105, 107] and some papers also include the MAC finite difference discretization [35, 68] or a FVM discretization [57, 69]. One of our goals is to experimentally verify the properties of these preconditioners for linear systems resulting from various IgA discretizations, especially the ones with high-order continuity, which is a unique feature of IgA. Moreover, the experiments for higher-degree C^0 IgA discretizations can give some information about the behavior of the preconditioners for higher-degree standard finite elements.

We include PCD, LSC, SIMPLE, SIMPLER, MSIMPLER, AL and MAL preconditioners in our comparison. We consider several 2D and 3D test problems where the geometry is described as a B-spline or a B-spline multipatch and we create several IgA discretizations of the incompressible Navier–Stokes problem in computational domains represented by these geometries. The considered pairs of finite dimensional spaces correspond to the isogeometric Taylor-Hood (TH) element. We have also experience with discretizations using the isogeometric subgrid (SG) element, which we have included in the numerical experiments in the paper [62]. However, the performance of the preconditioners does not differ much qualitatively from the cases with the maximum continuity TH element of the same degree and thus, we limit ourselves only to the TH element here.

In the beginning of this chapter, we specify the used software and hardware, the general settings of the experiments and describe the considered test problems and their discretizations. Then we deal with the choice of the mass matrix approximation in the preconditioners. Another set of experiments is devoted to the choice of PCD boundary conditions and tuning the preconditioner for the IgA linear systems. Finally, a comparison of all considered preconditioners is presented.

6.1 Software and hardware

In this section, we briefly describe the software tools used to obtain the presented numerical results and parameters of the machines on which the experiments were performed. We use our own implementation of the considered preconditioning techniques as well as the isogeometric incompressible flow solver itself.

Incompressible flow solver

Our research team has developed an incompressible flow solver based on isogeometric discretization of the incompressible Navier–Stokes equations and also RANS (Reynolds–Averaged Navier–Stokes) equations that are used for modeling turbulent flow. The solver is implemented in C++ within a framework of the G+Smo library.

G+Smo (Geometry + Simulation Modules) is an open-source object-oriented template C++ library that implements a generic concept for IgA, i.e., it enables to create and work with geometries defined as B-spline or NURBS objects and provides tools for analysis on domains described by these geometric objects. Thanks to object polymorphism and inheritance, it supports various discretization bases, such as B-spline, Bernstein, NURBS and also generalizations of B-spline basis that allow local refinement (hierarchical and truncated hierarchical B-splines). It works with bases of arbitrary dimension and polynomial degree. For more information about the library, see the documentation [75].

Our incompressible flow solver is not a public part of the G+Smo library at the moment, however, it will be made available in the future. The solver can handle steady-state and time-dependent incompressible flow in two and three spatial dimensions. It is mainly based on the coupled approach to the solution of the saddle-point problem, but a segregated pressure-correction method related to the SIMPLE algorithm has been also implemented. For turbulent flow simulation based on RANS equations, several turbulence models have been implemented (see the dissertation of Turnerová [103] for more on turbulence modeling with our solver). Further, several stabilization methods can be used to suppress spurious oscillations. The linear systems resulting from the discretization of the governing equations can be solved either directly or iteratively within the solver.

Linear algebra tools and linear solvers

The tools for linear algebra that are available in G+Smo are based on the Eigen library [55]. It provides data structures for vectors, dense and sparse matrices, methods for various matrix operations, commonly used matrix decompositions, linear solvers, etc. Moreover, a common interface for some popular linear solver packages (e.g. PARDISO [91]) is provided in Eigen.

In our flow solver, we use Eigen's sparse LU and PARDISO as direct solvers. If not stated otherwise, we use PARDISO with enabled OpenMP parallelization for direct solution of all linear systems in this work. As iterative solvers, we use the Krylov subspace methods available in G+Smo and also our own implementations of some methods, such as GMRES with right preconditioning.

We have implemented the block preconditioners for saddle-point linear systems described in Section 5.3, namely PCD, LSC, AL, MAL, SIMPLE, SIMPLER and MSIMPLER. They are constructed in a similar way and use the same interface as other preconditioners available in G+Smo, so that they can be easily used with the G+Smo iterative solvers.

Hardware

The numerical experiments were performed on machines with the following parameters:

- two-dimensional test problems: laptop, Ubuntu 20.04.4 LTS, 64-bit, Intel Core i7-10510U CPU @ 1.80GHz × 8, 16 GB RAM,

- three-dimensional test problems: virtual machine, Ubuntu 20.04.4 LTS, 64-bit, Intel Xeon CPU E5-2690 v2 @ 3.00GHz × 33, 182 GB RAM.

6.2 Experiments settings

For the purpose of comparison of different preconditioners, we always consider one linear system obtained from discretization of a given problem in a particular Picard iteration (for steady-state problems) or time step (for time-dependent problems). We solve that linear system with the full (non-restarted) right-preconditioned GMRES with a zero initial solution, $\mathbf{x}_0 = \mathbf{0}$. The iteration is stopped using the criterion

$$\frac{||\mathbf{r}_n||}{||\mathbf{b}||} < 10^{-6}, \tag{6.1}$$

where $\mathbf{b}$ is the right-hand side vector of the solved linear system.

Of course, in practical computations, it would be beneficial to choose the initial solution for the Krylov subspace method as the solution vector from the previous Picard iteration or time step. However, we choose zero initial solution to ensure the same conditions for all preconditioners in the comparison. From our experience, if we use zero initial vector for the iterative solution of the linear systems in all Picard iterations for a steady-state problem, the iteration count of the linear solver settles on a constant value after a few Picard iterations. Therefore, we consider linear systems obtained after some initial phase of the Picard iteration, specifically, after the relative norm of the nonlinear residual drops below 10^{-2}. For time-dependent problems, there are almost no fluctuations in the iteration count (with zero initial solution) during the computation. In this case, we consider linear systems obtained after performing five time steps for all tested problems.

6.2.1 Discretization bases

Since we are interested in convergence behavior of the preconditioners for various IgA discretizations, we consider several B-spline discretization bases of different polynomial degree and interelement continuity for each test problem. In this section, we describe how these discretization bases are constructed and introduce their notation.

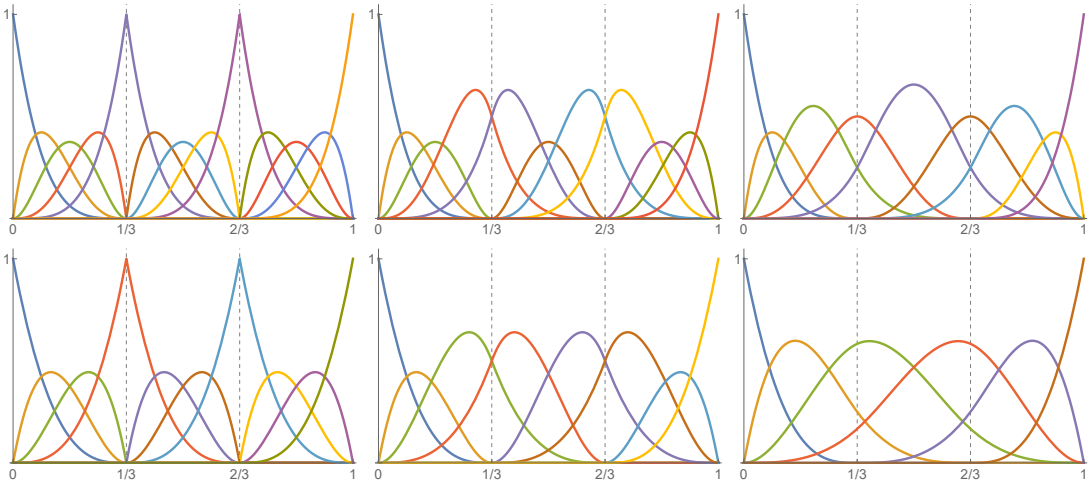
As an example, consider a rectangular domain which is a shape that can be described by a single B-spline object (patch) of arbitrary degree and continuity. Thus, in accordance with the isoparametric concept, we can use B-spline bases of arbitrary degree and continuity as our discretization basis. The simplest representation of the geometry (with continuous basis functions) is the following bilinear map

$$\mathbf{G}(\xi, \psi) = \sum_{i=1}^{2} \sum_{j=1}^{2} \mathbf{P}_{i,j} N_{i,1}(\xi) N_{j,1}(\psi), \tag{6.2}$$

where $N_{i,1}(\xi)$ and $N_{j,1}(\psi)$ are linear B-spline basis functions corresponding to knot vectors $\Xi = \Psi = (0, 0, 1, 1)$ and $\mathbf{P}_{i,j}$ are the control points corresponding to the vertices of the rectangle. Let us assume that we would like to use the isogeometric Taylor–Hood element of degree k and C^r continuity in both directions for pressure, i.e., the discrete pressure and velocity spaces on the parametric domain are $\widehat{\mathcal{Q}}^h_{TH} = \mathcal{S}^{r,r}_{k,k}$ and $\widehat{\boldsymbol{\mathcal{V}}}^h_{TH} = \mathcal{S}^{r,r}_{k+1,k+1} \times \mathcal{S}^{r,r}_{k+1,k+1}$, respectively (see Section 3.4). The pressure discretization

basis can be obtained by h-, p- and k-refinements of the basis in (6.2) and the velocity discretization basis is obtained by degree elevation (p-refinement) of the pressure basis. We denote such combination of discretization bases as $(k+1)$-k, C^r in this chapter. We refer to Section 3.2.2 for more details on the refinement algorithms.

For instance, we describe the construction of the discretization bases with $k = 3$ and $r = 0, 1, 2$ on a uniformly refined mesh with 3×3 elements. To obtain the 4-3, C^0 combination, we perform h-refinement of the basis in (6.2) first (i.e., the knots $1/3$ and $2/3$ are inserted into both knot vectors) and then we elevate the degree to 3 to get the pressure basis. For the 4-3, C^1 combination, we elevate the degree to 2 first, then we perform h-refinement and then we elevate the degree to 3. Finally, the combination 4-3, C^2 is constructed by elevating the degree to 3 first and h-refinement afterwards. The three resulting combinations of discretization bases are illustrated in Figure 6.1 in one dimension. Note that the C^0 isogeometric discretizations can be considered an analogy of standard finite elements of the same degree, at least from the matrix structure point of view.



(a) 4-3, C^0. (b) 4-3, C^1. (c) 4-3, C^2.

Figure 6.1: Examples of 1D velocity (top) and pressure (bottom) discretization bases of degree 4 and 3, respectively, with various order of continuity for the isogeometric TH element on a "mesh" with three elements.

If the computational domain is composed of rectangles in 2D or blocks in 3D such as for the first two test problems described in Section 6.3 below, we can construct discretization bases of arbitrary degree and continuity similarly as in the example above. (Recall that we consider conforming patches connected via identifying the corresponding control points at the interfaces and thus the continuity is always C^0 across the interfaces.) However, if the computational domain has curved boundaries, we cannot use discretization bases of arbitrary degree and continuity without violating the isoparametric concept. For example, if the domain geometry is described by a C^1-continuous biquadratic B-spline surface, using a bilinear discretization basis is not possible in isogeometric analysis, because the domain cannot be represented exactly in this basis.

6.3 Test problems

We have chosen three two-dimensional problems and their three-dimensional counterparts as model problems for the numerical experiments. Both enclosed and inflow/outflow types of flow are included. In this section, we describe the geometry and boundary conditions of these problems and summarize the problem parameters, discretization bases and meshes that are used in the comparisons.

The first two test problems described below are well-known benchmark problems with very simple computational domains consisting of one or more squares/rectangles (in 2D) or cubes/blocks (in 3D). These domains can be described by one or more B-spline patches with uniform orthogonal meshes that allow IgA discretization bases of arbitrary degree and continuity. Therefore, these are good first choice test examples for a comparison of the convergence properties of the selected preconditioners applied to IgA discretizations. Moreover, they are often used as benchmark problems in the literature and thus the results can be compared with results of other authors dealing with different discretization methods.

The third problem stems from industrial practice, specifically from modeling of flow in a water turbine. In this case, the B-spline geometry representation leads to curvilinear meshes and does not allow arbitrary degree and continuity of the discretization bases.

To investigate the influence of the degree and continuity of the discretization on the convergence of linear solvers, we consider several discretizations of each problem. In most cases, we present results only for selected discretizations (2-1, C^0 and some C^0 and C^{k-1} discretizations of higher degree k) and refer to Appendix B for complete results that involve all considered discretizations. Further, we always consider several different mesh refinements and values of problem parameters (viscosity, time step). For the time-dependent problems, we present results only for one time step size, $\Delta t = 0.01$, since there are no major qualitative differences in the performance of the preconditioners for different values of Δt.

6.3.1 Lid-driven cavity

The first test problem is a representative of enclosed flow problems – a cavity with a moving lid, usually denoted as the lid-driven cavity (LDC) problem. We consider both two- and three-dimensional case, denote them as LDC-2D and LDC-3D, respectively.

LDC-2D

The two-dimensional LDC problem is illustrated in Figure 6.2. The computational domain $\Omega = [0, 1]^2$ represents a square-shaped cavity filled with fluid with three sides defined as no-slip walls and the top boundary is a lid that slides from left to right and drives the flow in the cavity. That is, Dirichlet boundary conditions are defined on the whole boundary, $\partial\Omega_D = \partial\Omega$, where a constant nonzero velocity $\boldsymbol{u} = (1, 0)$ is prescribed on the top boundary and homogeneous Dirichlet conditions are set on the rest of the boundaries.

There are several ways to define the nonzero boundary condition on the top boundary leading to different computational models. If the nonzero horizontal velocity is prescribed also at the top two corners of the domain, we talk about a leaky cavity. If, on the contrary, the velocity at the corners is set to zero, we refer to the model as a watertight or non-leaky cavity. In both cases, there are singularities in pressure due to the discontinuity

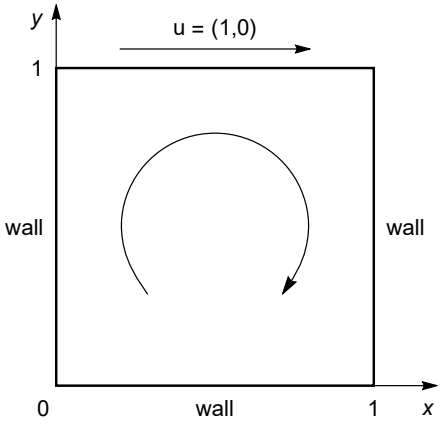


Figure 6.2: Schematic of the lid-driven cavity problem in 2D.

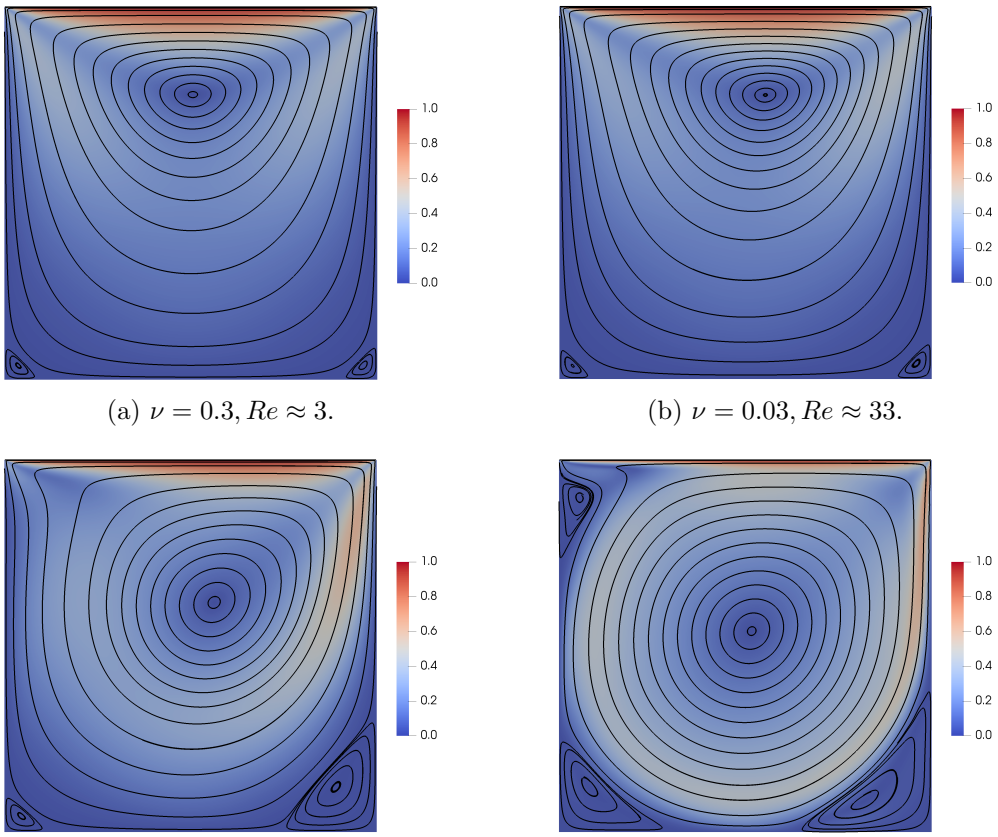
in the boundary condition. Sometimes, a regularized boundary condition is considered to avoid the singularities. We consider the watertight variant of the LDC problem.

The fact that Dirichlet boundary conditions for velocity are imposed on the whole boundary leads to non-unique pressure solution which is determined up to an arbitrary constant. In computations, the value of pressure is often fixed at one point of the domain to get a unique solution. We choose to set zero pressure at the bottom left corner.

The characteristic length scale and the reference velocity are chosen as $L = 1$ and $U = 1$. Thus, the Reynolds number is simply a reciprocal of the viscosity in this case, $Re = \nu^{-1}$. The character of the flow changes depending on the Reynolds number. For Reynolds numbers lower than some critical value (approximately 8000 for the LDC-2D problem), the problem has a stable steady-state solution. A time-periodic solution with waves running around the cavity walls occurs at the critical Reynolds number and for Re higher than 10000, the steady-state solution is unstable, see, e.g., [40]. Thus, it would not make sense to try to compute a steady solution for $Re \geq 8000$. We have chosen four values of viscosity for comparison, $\nu = 0.3, 0.03, 0.003, 0.0003$, which correspond to Reynolds numbers approximately 3, 33, 333 and 3333. Figure 6.3 displays steady-state velocity solutions for all considered viscosity values.

We perform our experiments for IgA discretizations of degree k varying from 1 to 4 for pressure (i.e., 2 to 5 for velocity) with C^0 to C^{k-1} continuity. For all discretizations, we consider four uniform meshes with 16×16, 32×32, 64×64 and 128×128 elements and we denote these meshes as M1, M2, M3 and M4, respectively. As described in Section 3.3 in more detail, the size of the linear systems resulting from different discretizations varies on a given mesh. The number of degrees of freedom (DOFs), the number of nonzeros and their percentage in the matrix for all discretizations and all uniform meshes are summarized in Appendix A, Table A.1.

We are also interested in the performance of the preconditioners on stretched meshes. Typically, meshes that are refined near boundaries are used for the LDC problem to resolve the singularities at the top corners of the domain. Therefore, we consider three stretched meshes obtained from the uniform mesh M1 by recursively refining one row of elements near all boundaries. We perform two, four and six such recursive refinements resulting in meshes denoted as SM1, SM2 and SM3, respectively. See Figure 6.4, where the domain is divided into four parts with meshes M1, SM1, SM2 and SM3 displayed on different parts.



(a) $\nu = 0.3, Re \approx 3.$ (b) $\nu = 0.03, Re \approx 33.$

(c) $\nu = 0.003, Re \approx 333.$ (d) $\nu = 0.0003, Re \approx 3333.$

Figure 6.3: Streamlines and velocity magnitude of the steady-state solutions of the LDC-2D problem for different values of viscosity.

LDC-3D

The three-dimensional LDC problem is analogous to the two-dimensional one. The computational domain is $\Omega = [0, 1]^3$ and we prescribe the velocity $\boldsymbol{u} = (1, 0, 0)$ on the top boundary (lid). Again, we consider the watertight cavity model and set the pressure to zero at the corner at the origin of the coordinate system.

Similarly to the two-dimensional cavity, the length scale L and reference velocity U are equal to one and thus the Reynolds number is $Re = \nu^{-1}$. In the three-dimensional case, the steady solution loses stability for Reynolds number less than 1000 [40, 95]. We present results for viscosity values $\nu = 0.1, 0.05, 0.01, 0.005$, corresponding to $Re = 10, 20, 100, 200$. An example of a steady velocity solution in the 3D cavity with $Re = 100$ is shown in Figure 6.5.

The experiments were performed for IgA discretizations of degree k from 1 to 3 for pressure with C^0 to C^{k-1} continuity on three uniform meshes denoted as M1, M2, M3 with $4 \times 4 \times 4$, $8 \times 8 \times 8$ and $16 \times 16 \times 16$ elements, respectively. The number of DOFs, the number of nonzeros and their percentage in the matrix for all discretizations and all uniform meshes are summarized in Appendix A, Table A.3.

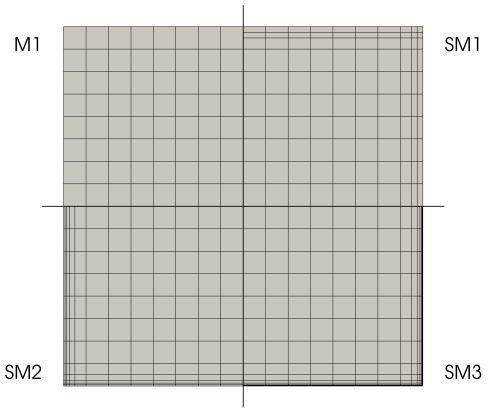


Figure 6.4: The uniform mesh M1 and three stretched meshes used for LDC-2D.

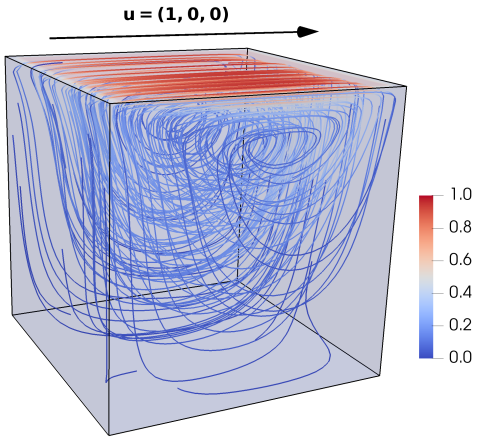


Figure 6.5: Streamlines and velocity magnitude of the steady-state solution of the LDC-3D problem for $\nu = 0.01$ ($Re = 100$).

6.3.2 Backward-facing step

The second test problem is the backward-facing step (BFS) problem, a widely used inflow/outflow benchmark problem. Analogously to the notation above, we denote the two- and three-dimensional case as BFS-2D and BFS-3D, respectively.

BFS-2D

The setting of the two-dimensional BFS problem is illustrated in Figure 6.6. It models fluid flow in a channel with a backward-facing step on the bottom. The fluid enters the channel through the left vertical boundary (inflow) and leaves through the right vertical boundary (outflow). The top and bottom boundaries are defined as no-slip walls. Thus, homogeneous Dirichlet conditions are set on the top and bottom boundary, a nonzero velocity is prescribed on the inflow boundary and the do-nothing condition is set on the outflow boundary. We consider an inflow velocity with zero y-component and a parabolic profile with maximum of one in the x-component.

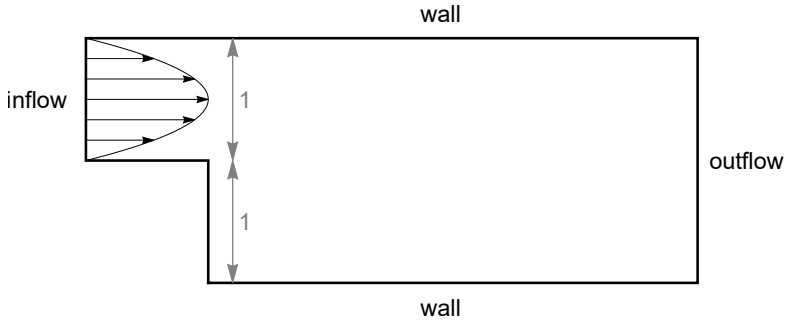
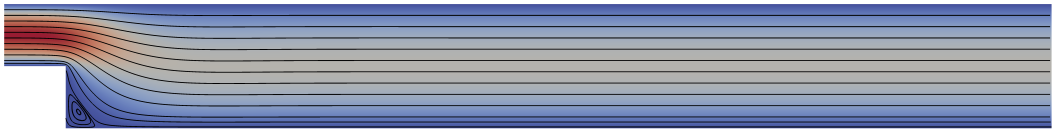


Figure 6.6: Schematic of the backward-facing step problem in 2D.

The choice of the characteristic length L and the reference velocity U varies in the literature. If L is taken to be the height of the channel at the outflow and U as the maximum of the inflow velocity as in [40], i.e., $L = 2$ and $U = 1$ in our case, then the

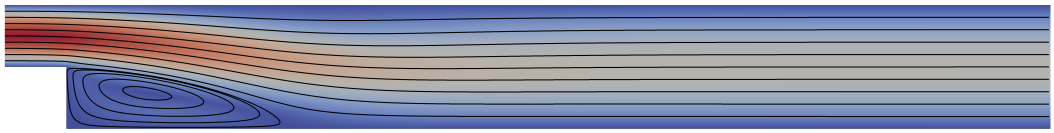
Reynolds number is $Re = 2 \cdot \nu^{-1}$. According to [40], the steady solution of this problem is unstable for $\nu < 0.001$. For the comparison of preconditioners, we consider viscosity values $\nu = 0.2, 0.02, 0.002$ corresponding to $Re = 10, 100, 1000$. Some experiments were also performed for $\nu = 0.01$ ($Re = 200$).

Steady-state velocity solutions for different viscosity values are shown in Figure 6.7. We note that we use $\nu = 0.02$ in most of the experiments, for example when investigating mesh dependence. In such case, we choose the length of the channel behind the step to be 8. However, it is important for the channel to be long enough such that the outflow boundary does not intersect any vortex, otherwise the do-nothing condition at the outflow is not appropriate. Therefore, we double the channel length in the experiments where lower viscosity values are included in the comparison.

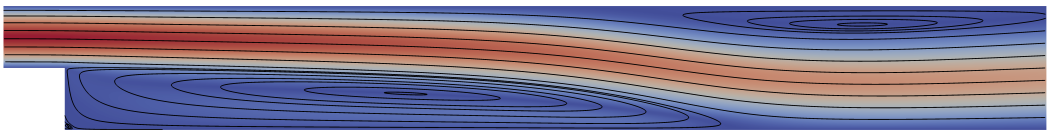


(a) $\nu = 0.2, Re = 10$.

(b) $\nu = 0.02, Re = 100$.



(c) $\nu = 0.01, Re = 200$.



(d) $\nu = 0.002, Re = 1000$.

Figure 6.7: Streamlines and velocity magnitude of the steady-state solutions of the BFS-2D problem for different values of viscosity.

The computational domain can be described as a union of three rectangles, i.e., it can be represented as a multipatch consisting of three rectangular B-spline patches. As already mentioned in Section 3.2, we consider conforming patches where it is possible to identify the basis functions on both sides of the interface in order to connect the individual patches. We discretize the BFS-2D problem using bases of degree k varying from 1 to 4 for pressure with all possible orders of continuity. We compare four uniform meshes denoted as M1 to M4, see Figure 6.8 where the mesh M1 is displayed on the domain with channel length 8. The uniform meshes have a total of 272, 1088, 4352 and 17408 elements, respectively. The number of DOFs, the number of nonzeros and their percentage in the matrix are summarized in Appendix A, Table A.2.

Note that the backward-facing step problem, similarly to the lid-driven cavity problem, has a singularity – the pressure tends to infinity at the non-convex corner. In

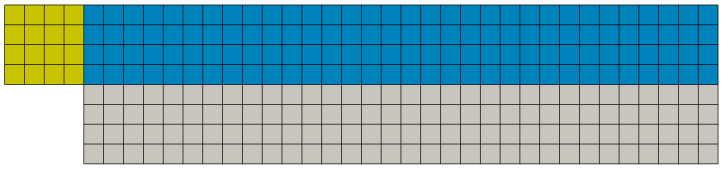


Figure 6.8: Uniform mesh M1 for the BFS-2D problem.

practical computations, local refinement of the mesh in the vicinity of the corner is necessary to resolve the singularity. Therefore, we also work with stretched meshes with such "local" refinement, where one row of elements around the corner is recursively refined two, four and six times, denoted as SM1, SM2 and SM3, respectively.

BFS-3D

The three-dimensional BFS problem is analogous to the two-dimensional one. The computational domain is only extended to the third dimension and periodic boundary conditions are set on the sides of the 3D channel. The same parabolic inflow velocity with maximum of one is considered.

We compare results for three values of viscosity, $\nu = 0.1, 0.05, 0.01$ corresponding to $Re = 20, 40, 200$. The steady-state velocity solution for $Re = 200$ is shown in Figure 6.9.

We consider discretizations of degree $k = 1, 2, 3$ and various continuity on three uniform meshes denoted as M1 to M3. Their number of elements is 144, 1152 and 9216, respectively. The number of DOFs, the number of nonzeros and their percentage in the matrix are summarized in Appendix A, Table A.4.

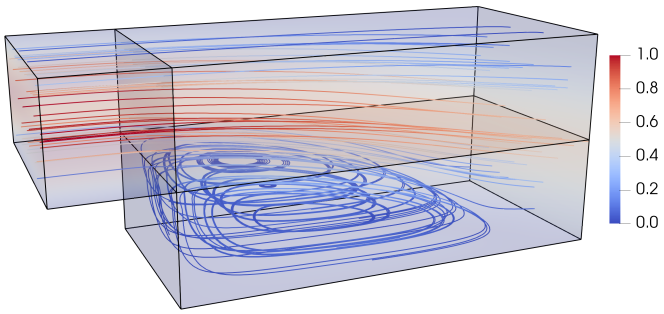


Figure 6.9: Streamlines and velocity magnitude of the steady-state solution of the BFS-3D problem for $\nu = 0.01$ ($Re = 200$).

6.3.3 Water turbine runner wheel

The geometry of the third test problem represents a runner wheel of a water turbine. In this case, the three-dimensional problem is primary and the two-dimensional one originated as its simplification. We denote the two problems as TB-2D and TB-3D, where "TB" stands for "turbine blade".

TB-2D

The two-dimensional problem models flow in a row of 2D blade profiles obtained by unfolding a cylindrical cross-section of the turbine runner wheel. The computational

domain (see Figure 6.10) consists of three B-spline patches and represents a strip between two parts of a blade profile. The upper and lower boundary of the middle (blue) patch form the blade profile and periodic boundary conditions are set at the upper and lower boundaries of the remaining two patches. The left vertical boundary is an inflow boundary, where we prescribe a constant velocity vector with a direction tangent to the camber line of the profile at the leading point. The inflow velocity magnitude is approximately 8.05. The right vertical boundary is an outflow with the do-nothing boundary condition.

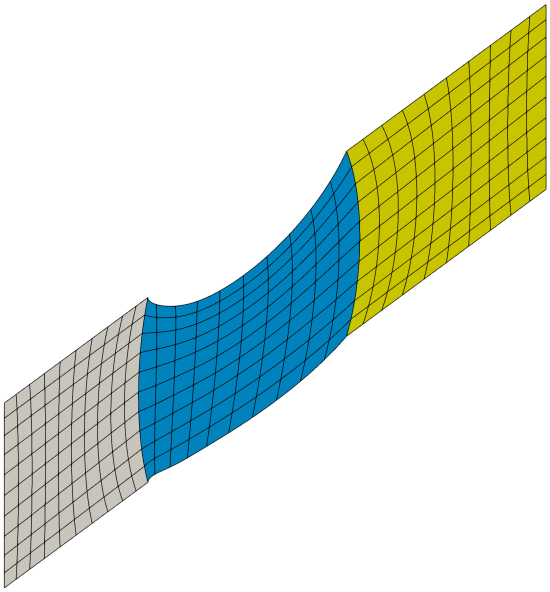


Figure 6.10: Computational domain for the TB-2D problem consisting of three cubic B-spline patches with mesh M1.

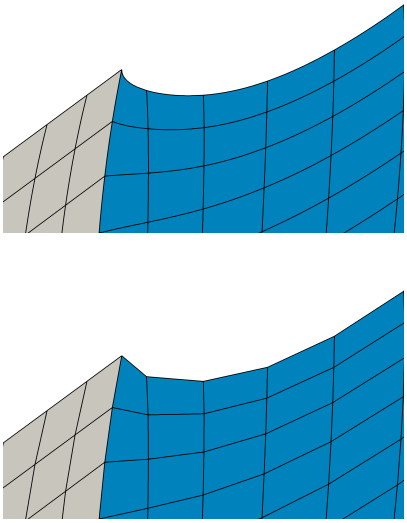


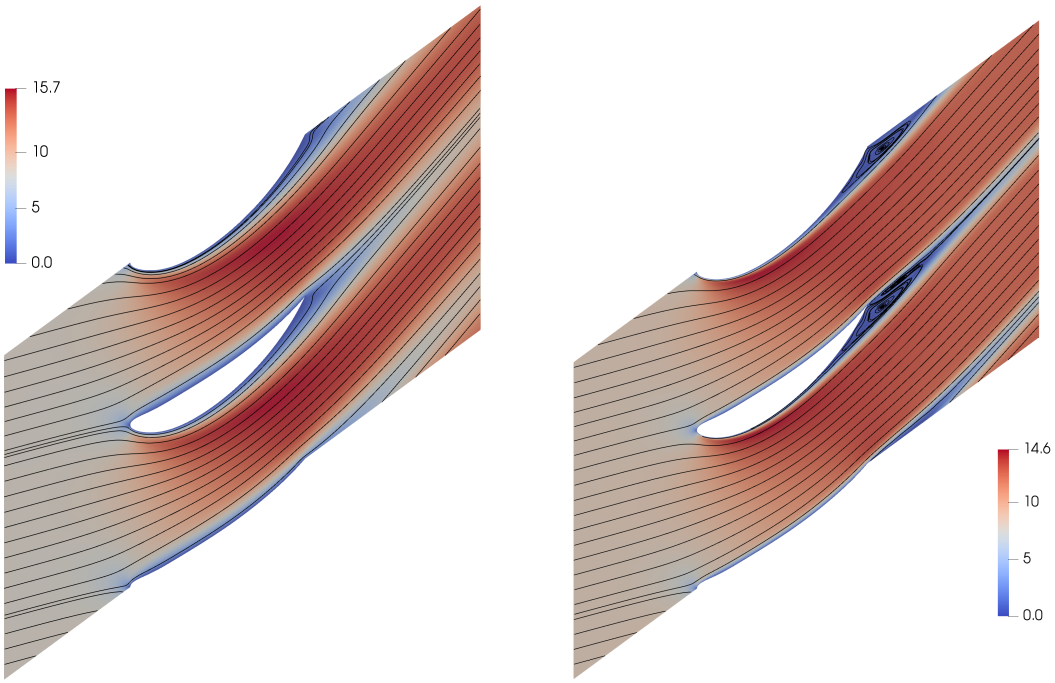
Figure 6.11: Detail of the original cubic geometry (top) and its linear approximation (bottom).

The characteristic length scale L can be chosen as the chord length of the blade profile and the reference velocity U as the inflow velocity magnitude. That is, in our case, the Reynolds number is

$$Re = \frac{UL}{\nu} = \frac{8.05 \cdot 0.37}{\nu}. \tag{6.3}$$

We use viscosity values $\nu = 0.1, 0.01, 0.001$ in the experiments, which corresponds to Reynolds numbers approximately 30, 298 and 2979, respectively. The steady-state velocity solutions for $\nu = 0.01$ and $\nu = 0.001$ are shown in Figure 6.12.

The original geometry displayed in Figure 6.10 is described by B-spline surfaces of degree $k = 3$ with C^2 continuity at the interior knots and their parametrization forms a curvilinear mesh. As already mentioned, we cannot use discretizations of arbitrary degree and continuity for such computational domain in IgA. Besides the discretization 4-3, C^2 on the exact domain, we consider a linear approximation of the domain with discretizations 2-1, C^0 and 4-3, C^0, see Figure 6.11 for illustration. For both domains, we consider four meshes, the one shown in Figure 6.10 denoted as M1, and M2, M3 and M4 that are obtained by one, two and three successive uniform refinements of M1. By uniform refinement we mean inserting a knot in the middle of each knot span in the parametric space. These "uniform" meshes have 300, 1200, 4800 and 19200 elements,



(a) $\nu = 0.01, Re \approx 298$. (b) $\nu = 0.001, Re \approx 2979$.

Figure 6.12: Streamlines and velocity magnitude of the steady-state solutions of the TB-2D problem for two different values of viscosity.

respectively. Again, we summarize the number of DOFs, the number of nonzeros and their percentage in the matrix in Appendix A, Table A.5. Similarly to the test problems described above, we also consider stretched meshes obtained by recursively refining one row of elements along the blade profile two, four and six times and denote them as SM1, SM2 and SM3, respectively.

TB-3D

The last test problem is a flow in a domain representing the volume between two blades of a Kaplan turbine runner wheel. An example of a geometry used for modeling the flow in the turbine is shown in Figure 6.13 (left). It involves the stationary guide vanes (blue) and the rotating runner blades (red). The fluid enters the turbine through the left boundary and leaves through the right boundary. We consider the stationary and rotating part of the turbine separately: a stationary flow is computed in the stationary part first and the obtained velocity is used as a boundary condition for the rotating part. Since both parts are radially periodic, the computational domain can be defined as a strip between two guide vanes or runner blades, respectively, with periodic boundary conditions on the boundaries in front of and behind the vane/blade. The runner wheel computational domain is shown in Figure 6.13 (right), where the top and bottom boundaries correspond to the inflow and outflow, respectively, and the blades are displayed in red color.

Note that we consider this example in order to test the preconditioners for a problem with a real-world geometry, but our settings and parameters are far from the real-world problem from industrial practice. First, we do not consider any rotation of the runner

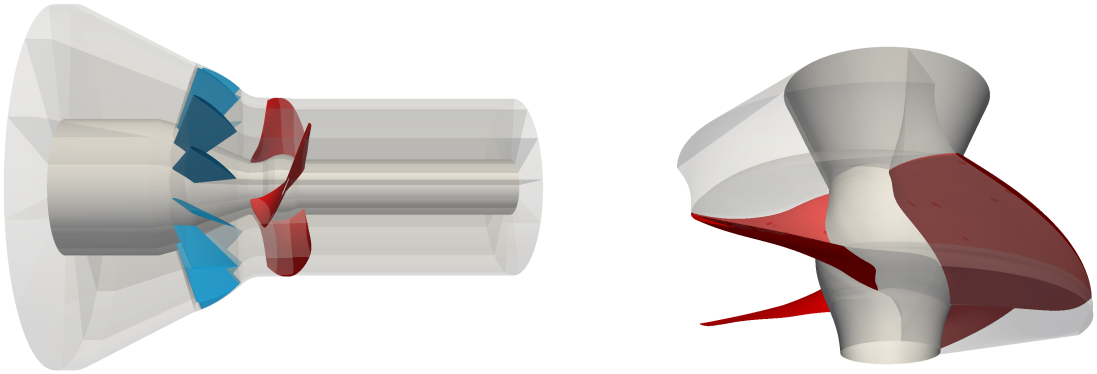


Figure 6.13: The Kaplan turbine geometry: stationary and rotating part (left), the computational domain for the rotating part (right).

wheel and not even the periodic boundary conditions. Since the periodic sides are not parallel and velocity is a vector quantity, the corresponding velocity vectors on the periodic sides are not identical but rotated relative to each other. Thus, the periodic boundary conditions introduce a coupling between velocity components. This results in matrix $\mathbf{F}$ that is not block diagonal, which makes the solution more expensive, since the linear systems with this matrix cannot be split into three smaller systems (each for one velocity component). Therefore we define solid walls (homogeneous Dirichlet conditions) instead of the periodic sides. Further, we consider coarse meshes and very high viscosity values in comparison to the ones used in practice.

Figure 6.14 shows the computational domain consisting of three B-spline patches of degree $k = 3$ with the coarsest considered mesh denoted as M1. We consider another two meshes, M2 and M3, obtained from M1 by one and two successive uniform refinements in the sense described for TB-2D. These meshes have 400, 3200 and 25600 elements, respectively, and the numbers of DOFs and nonzeros and their percentage in the matrix are summarized in Appendix A, Table A.6. For this problem, we perform the experiments only for the discretization based on the original geometry representation 4-3, C^2.

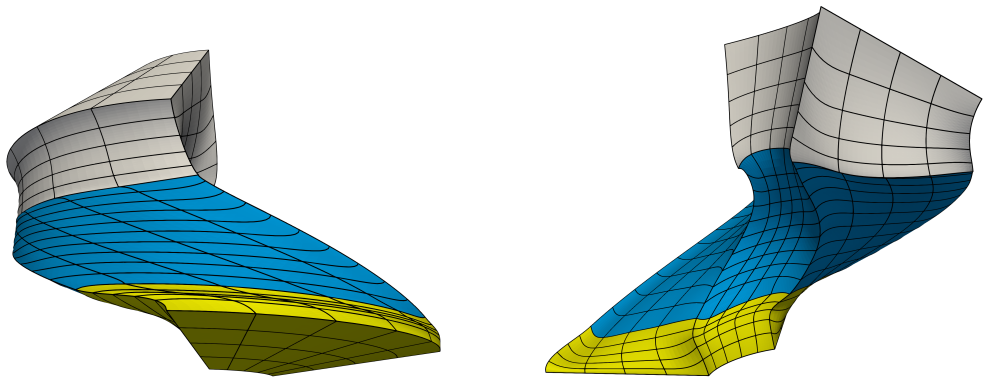


Figure 6.14: Computational domain for the TB-3D problem with mesh M1 (two different views).

We consider two viscosity values, $\nu = 0.1$ and $\nu = 0.05$ for this test problem. The inflow velocity is taken as the solution computed in the stationary part of the turbine with the corresponding viscosity. The maximum of its magnitude is approximately 9 for

$\nu = 0.1$ and 8.7 for $\nu = 0.05$. The steady-state velocity solution for $\nu = 0.05$ is shown in Figure 6.15.

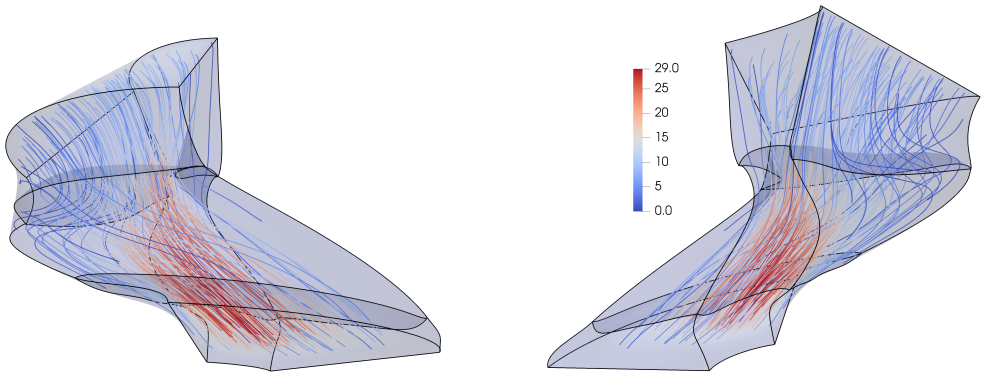


Figure 6.15: Streamlines and velocity magnitude of the steady-state solution of the TB-3D problem for $\nu = 0.05$ (two different views).

6.4 Approximation of mass matrices

The velocity or pressure mass matrix arise in most of the block preconditioners used in this chapter. In many cases, the inverse velocity mass matrix $\mathbf{M}_u^{-1}$ appears in a product with other matrices (see the Schur complement approximations (5.35), (5.52) of PCD and LSC and also the MSIMPLER preconditioner). In these expressions, the mass matrix is usually replaced by a diagonal matrix to make the inverse easy to compute and preserve sparsity of the resulting matrix.

The introduction of scaling matrices into the BFBt preconditioner to obtain the LSC preconditioner has proven that such diagonal scaling can have a significant effect on the performance of the preconditioner (see the comparisons in [35]). The diagonal mass matrix approximation is typically chosen to be equal to the diagonal of $\mathbf{M}_u$ itself, i.e., $\widehat{\mathbf{M}}_u = \mathrm{diag}(\mathbf{M}_u)$. It can be shown for some common standard finite elements that the mass matrix and its diagonal are spectrally equivalent, see, e.g., [40, Chapter 4]. We believe that particular choice of the diagonal scaling can play an important role and deserves a bit of attention.

When it comes to diagonal approximation of the mass matrix, probably the first method that comes to mind is the so-called mass lumping. It is widely used in some areas of computational mathematics, for example in explicit time-stepping schemes. A simple and probably the most popular mass lumping technique is the row-sum method. Consider a given mass matrix $\mathbf{M} \in \mathbb{R}^{N\times N}$ and denote the row-sum lumped mass matrix as $\widehat{\mathbf{M}}_L$. As the name of the method suggests, the diagonal entries of the matrix $\widehat{\mathbf{M}}_L$ are obtained by summing up the elements in the rows of $\mathbf{M}$, that is

$$\widehat{M}_{L,ii} = \sum_{j=1}^{N} M_{ij}, \quad i = 1, \ldots, N. \tag{6.4}$$

This method is not suitable for some higher-order standard finite elements, since it can lead to negative values in $\widehat{\mathbf{M}}_L$. However, positivity of all entries of $\widehat{\mathbf{M}}_L$ is guaranteed in

IgA thanks to the pointwise positivity of the basis functions. Moreover, it holds

$$\widehat{M}_{L,ii} = \sum_{j=1}^{N} M_{ij} = \sum_{j=1}^{N} \int_{\Omega} Q_i Q_j = \int_{\Omega} Q_i \left(\sum_{j=1}^{N} Q_j \right) = \int_{\Omega} Q_i, \tag{6.5}$$

where Q_i, Q_j are the B-spline (or also NURBS) basis functions. This follows from the partition of unity property of the isogeometric basis. Thus, the lumped mass matrix can be obtained directly by integrating the N basis functions.

It is a known issue that when the lumped mass matrix is used in explicit methods with higher-order IgA discretizations, the scheme becomes at most second-order accurate, see, e.g., [23, 24, 30]. To the best of our knowledge, finding a higher-order accurate mass lumping method for IgA is still an open problem. However, this is not a problem in our case, since we use the lumped mass matrix only in the preconditioning operator, which is meant to be an approximation of the original problem.

In this section, we compare the block preconditioners with the two mentioned variants of diagonal mass matrix approximation and choose the superior variant for the experiments in the rest of this chapter. The model problem for this comparison is the BFS-2D problem with $Re = 200$ and $\Delta t = 0.01$ in the time-dependent case.

6.4.1 $\widehat{\mathbf{M}}_u$ in LSC and MSIMPLER

A diagonal approximation $\widehat{\mathbf{M}}_u$ of the velocity mass matrix $\mathbf{M}_u$ appears several times in both preconditioners. We compare the GMRES iteration counts with the two choices of $\widehat{\mathbf{M}}_u$, the diagonal of the consistent (full) mass matrix $\widehat{\mathbf{M}}_u = \mathrm{diag}(\mathbf{M}_u)$ and the row-sum lumped velocity mass matrix $\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$.

Table 6.1 shows results for the steady-state problem. Generally, LSC and MSIMPLER give very similar iteration counts in all cases. Comparing the two mass matrix approximations, there is not a significant difference in convergence for discretizations with the maximum continuity. The same can be said about the coarsest mesh M1 in general. However, with $\widehat{\mathbf{M}}_u = \mathrm{diag}(\mathbf{M}_u)$, the iteration counts increase quite significantly with mesh refinement for discretizations with lower continuity, especially for the C^0 cases. On the other hand, this does not happen with $\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$, where the iteration counts are more or less independent of the discretization degree and continuity for all considered meshes. For the 2-1, C^0 discretization, the convergence seems even independent of the mesh refinement. If we view this discretization as an analogy of the standard Q2-Q1 finite element, these results indicate that mass lumping might improve the properties of LSC and MSIMPLER also for standard low-order FEM discretizations.

Results for the time-dependent problem are displayed in Table 6.2. In this case, the convergence is almost the same for both choices of the mass matrix approximation. The variants with $\widehat{\mathbf{M}}_{u,L}$ usually require one or two iterations less.

Based on the results presented in this section, we choose to use mass lumping for LSC and MSIMPLER in all further experiments.

6.4.2 $\widehat{\mathbf{M}}_u$ in PCD

A diagonal approximation of $\widehat{\mathbf{M}}_u$ is used only in the modified version of the PCD preconditioner proposed by Elman and Tuminaro [41], where the pressure-Poisson matrix $\mathbf{A}_p$ is not assembled, but defined as $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$ instead. We will comment on using

$\text{diag}(\mathbf{M}_u)$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	33	33	40	65
5-4, C^0	28	33	56	113
5-4, C^3	28	24	28	36

(a) LSC with $\widehat{\mathbf{M}}_u = \text{diag}(\mathbf{M}_u)$.

$\widehat{\mathbf{M}}_{u,L}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	29	25	24	29
5-4, C^0	29	29	33	40
5-4, C^3	28	25	30	38

(b) LSC with $\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$.

$\text{diag}(\mathbf{M}_u)$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	31	31	39	64
5-4, C^0	26	32	56	114
5-4, C^3	29	25	27	36

(c) MSIMPLER with $\widehat{\mathbf{M}}_u = \text{diag}(\mathbf{M}_u)$.

$\widehat{\mathbf{M}}_{u,L}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	28	24	23	27
5-4, C^0	29	28	31	38
5-4, C^3	28	25	28	37

(d) MSIMPLER with $\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$.

Table 6.1: Iteration counts of LSC and MSIMPLER preconditioner with two variants of mass matrix approximation for the steady-state BFS-2D problem with $\nu = 0.01$. (Results for all discretizations in Appendix B, Table B.1.)

the two variants of $\widehat{\mathbf{M}}_u$ in PCD in Section 6.5 which is devoted to experiments with various boundary conditions for PCD.

Note that we do not consider any explicit approximation of the pressure mass matrix $\mathbf{M}_p$ in the PCD Schur complement approximations.

6.4.3 $\widehat{\mathbf{M}}_p$ in AL and MAL

In augmented Lagrangian approaches, an approximation $\widehat{\mathbf{M}}_p$ of the pressure mass matrix appears in the inverse approximate Schur complement $\widehat{\mathbf{S}}_\gamma^{-1}$ (5.68). Moreover, we consider $\mathbf{W} = \widehat{\mathbf{M}}_p$, thus, the mass matrix approximation is present also in the augmented block $\mathbf{F}_\gamma$.

We compare the AL and MAL preconditioners with the approximation $\widehat{\mathbf{M}}_p = \text{diag}(\mathbf{M}_p)$ and the row-sum lumped variant $\widehat{\mathbf{M}}_p = \widehat{\mathbf{M}}_{p,L}$ for the BFS-2D problem on mesh M1. We have chosen three discretizations for the presentation of the results, one with the lowest degree (2-1, C^0), one with higher degree and C^0 continuity (4-3, C^0) and the last one with higher degree and maximum continuity (4-3, C^2). Since the convergence of these preconditioners depends on a parameter γ, we consider several values of γ in this comparison to see if the dependence on γ differs for different choices of $\widehat{\mathbf{M}}_p$.

Table 6.3 shows iteration counts of GMRES preconditioned with AL and MAL with the two choices of mass matrix approximation for the steady-state problem. The iteration count ">300" means that the GMRES method reached the maximum number of iterations that was set to 300. Results for the time-dependent problem are shown in Table 6.4.

Generally, lumping does not seem to bring any advantage when used in the AL-based preconditioners. On the contrary, the iteration counts are higher with lumping in many cases or, at least, it seems that larger γ is needed with $\widehat{\mathbf{M}}_{p,L}$ to reach similar convergence as for $\text{diag}(\mathbf{M}_p)$, which is not desirable. Thus, we do not use lumping for AL and MAL in the rest of the experiments.

$\text{diag}(\mathbf{M}_u)$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	6	6	5	5
5-4, C^0	16	15	14	14
5-4, C^3	8	7	6	7

(a) LSC with $\widehat{\mathbf{M}}_u = \text{diag}(\mathbf{M}_u)$.

$\widehat{\mathbf{M}}_{u,L}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	5	5	4	4
5-4, C^0	15	13	12	12
5-4, C^3	7	6	5	6

(b) LSC with $\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$.

$\text{diag}(\mathbf{M}_u)$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	7	6	5	5
5-4, C^0	19	17	16	17
5-4, C^3	10	8	7	8

(c) MSIMPLER with $\widehat{\mathbf{M}}_u = \text{diag}(\mathbf{M}_u)$.

$\widehat{\mathbf{M}}_{u,L}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	6	5	5	5
5-4, C^0	18	16	15	15
5-4, C^3	9	8	6	7

(d) MSIMPLER with $\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$.

Table 6.2: Iteration counts of LSC and MSIMPLER preconditioner with two variants of mass matrix approximation for the time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$. (Results for all discretizations in Appendix B, Table B.2.)

6.5 Boundary conditions for PCD

We have mentioned several possible choices of boundary conditions (BCs) for the discrete operators on the pressure space $\mathbf{A}_p$ and $\mathbf{F}_p$ that can be found in the literature. In this section, we compare different choices and choose the best variant to be used in further experiments. Specifically, we consider the following choices and introduce their notation:

1. original PCD [67]: $\widehat{\mathbf{S}} = \widehat{\mathbf{S}}^{\text{orig}}_{\text{PCD}}$ defined in (5.40)
   - $\mathbf{A}_p$ and $\mathbf{F}_p$ defined with Dirichlet BCs on $\partial\Omega_{\text{in}}$ and Neumann BCs elsewhere,
   - $\mathbf{A}_p$ assembled using (5.36),
   - denote as $\text{PCD}^{\text{orig}}_{\text{bc1}}$,
2. Elman, Tuminaro [41]: $\widehat{\mathbf{S}} = \widehat{\mathbf{S}}^{\text{mod}}_{\text{PCD}}$ defined in (5.42)
   - $\mathbf{F}_p$ defined with Robin BCs on $\partial\Omega_{\text{in}}$ and Neumann BCs elsewhere,
   - $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$,
   - denote as $\text{PCD}^{\text{mod}}_{\text{bc2}}$,
3. Elman, Tuminaro [41]: $\widehat{\mathbf{S}} = \widehat{\mathbf{S}}^{\text{mod}}_{\text{PCD}}$ defined in (5.42)
   - $\mathbf{F}_p$ defined with Robin BCs on $\partial\Omega_{\text{in}}$, Dirichlet BCs on $\partial\Omega_{\text{out}}$ and Neumann BCs on $\partial\Omega_{\text{char}}$,
   - $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$,
   - denote as $\text{PCD}^{\text{mod}}_{\text{bc3}}$,
4. Blechta [9]: $\widehat{\mathbf{S}} = \widehat{\mathbf{S}}^{R}_{\text{PCD}}$ defined in (5.45)
   - $\mathbf{F}_p$ defined with Robin BCs on $\partial\Omega_{\text{in}}$, Dirichlet on $\partial\Omega_{\text{out}}$ and Neumann on $\partial\Omega_{\text{char}}$,

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	107	210	190
0.1	29	36	51
1	8	6	12
2	6	5	8
5	5	4	5
10	4	3	4
50	3	2	3
100	3	2	2

(a) AL with $\widehat{\mathbf{M}}_p = \mathrm{diag}(\mathbf{M}_p)$.

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	135	>300	258
0.1	50	153	136
1	12	24	28
2	9	13	17
5	6	7	10
10	5	5	7
50	3	3	4
100	3	3	3

(b) AL with $\widehat{\mathbf{M}}_p = \widehat{\mathbf{M}}_{p,L}$.

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	111	218	195
0.1	28	40	50
1	51	119	64
2	75	174	93
5	104	267	138
10	114	>300	169
50	117	>300	187
100	112	>300	180

(c) MAL with $\widehat{\mathbf{M}}_p = \mathrm{diag}(\mathbf{M}_p)$.

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	151	>300	>300
0.1	49	153	137
1	30	40	31
2	47	62	39
5	78	111	66
10	102	162	95
50	117	>300	168
100	116	>300	181

(d) MAL with $\widehat{\mathbf{M}}_p = \widehat{\mathbf{M}}_{p,L}$.

Table 6.3: Iteration counts of the AL-based preconditioners with two variants of mass matrix approximation for the steady-state BFS-2D problem with $\nu = 0.01$.

- $\mathbf{A}_p$ assembled using (5.36), defined with Dirichlet BCs on $\partial\Omega_{\text{out}}$ and Neumann BCs elsewhere,
- denote as $\text{PCD}^{\text{mod}}_{\text{bc4}}$.

In addition, we experiment with two new variants of the Schur complement approximation. Recall that the elements of $\mathbf{F}_p$ are given by the integral formula (5.37). Using the formula (5.36) for the elements of $\mathbf{A}_p$, the (steady-state) pressure convection–diffusion matrix can be written as

$$\mathbf{F}_p = \mathbf{N}_p(\boldsymbol{u}_h^k) + \nu\mathbf{A}_p, \tag{6.6}$$

where $\mathbf{N}_p(\boldsymbol{u}_h^k)$ is a discretization of the linearized convective term on the pressure space. Of course, it holds analogously in the time-dependent case using (5.38) instead of (5.37). Inspired by [41], we use $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$ also in (6.6) for consistency. Adding the boundary term corresponding to Robin boundary condition on $\partial\Omega_{\text{in}}$ as in (5.44) leads to a modification of $\text{PCD}^{\text{mod}}_{\text{bc2}}$. Furthermore, we consider a variant with no boundary modification at all, relying only on the boundary conditions defined implicitly through $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$. In such case, the only new matrix to be assembled is $\mathbf{N}_p(\boldsymbol{u}_h^k)$ and even the knowledge of the individual parts of the boundary is not needed. To summarize, we propose the following variants:

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	29	54	39
0.1	33	50	43
1	35	32	31
2	30	24	24
5	22	17	17
10	17	13	13
50	10	8	8
100	8	6	6

(a) AL with $\widehat{\mathbf{M}}_p = \mathrm{diag}(\mathbf{M}_p)$.

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	28	47	40
0.1	31	46	40
1	33	40	39
2	30	32	32
5	29	30	30
10	22	23	23
50	13	13	13
100	10	10	10

(b) AL with $\widehat{\mathbf{M}}_p = \widehat{\mathbf{M}}_{p,L}$.

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	36	66	49
0.1	41	55	49
1	38	35	34
2	32	28	26
5	24	26	20
10	20	28	20
50	22	40	24
100	24	51	32

(c) MAL with $\widehat{\mathbf{M}}_p = \mathrm{diag}(\mathbf{M}_p)$.

γ	2-1, C^0	4-3, C^0	4-3, C^2
0.01	35	57	49
0.1	38	55	49
1	37	42	41
2	32	34	35
5	31	32	32
10	25	26	25
50	20	25	19
100	21	27	20

(d) MAL with $\widehat{\mathbf{M}}_p = \widehat{\mathbf{M}}_{p,L}$.

Table 6.4: Iteration counts of the AL-based preconditioners with two variants of mass matrix approximation for the time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$.

1. $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc5}}$:
   - $\widehat{\mathbf{S}} = \widehat{\mathbf{S}}^{\mathrm{mod}}_{\mathrm{PCD}}$,
   - $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$,
   - $\mathbf{F}_p = \mathbf{N}_p(\boldsymbol{u}_h^k) + \nu\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$ + Robin BCs on $\partial\Omega_{\mathrm{in}}$,
2. $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc6}}$:
   - $\widehat{\mathbf{S}} = \widehat{\mathbf{S}}^{\mathrm{orig}}_{\mathrm{PCD}}$,
   - $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$,
   - $\mathbf{F}_p = \mathbf{N}_p(\boldsymbol{u}_h^k) + \nu\mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$.

The mass matrix approximation is $\widehat{\mathbf{M}}_u = \mathrm{diag}(\mathbf{M}_u)$, unless stated otherwise. Similarly to Section 6.4, we use BFS-2D with $Re = 200$ (and $\Delta t = 0.01$ in the time-dependent case) as the test problem.

6.5.1 Scaling of Dirichlet conditions

Recall that the Dirichlet boundary conditions for $\mathbf{F}_p$ (and $\mathbf{A}_p$, if it is assembled using (5.36)) are applied by modifying certain rows and columns of the matrix such that they

contain only diagonal entries. It was also mentioned in Section 5.3.2 that the value of these entries is arbitrary. Consider a constant diagonal value and denote it as α. A straightforward choice would be $\alpha = 1$, which is considered, e.g., by Blechta [9]. However, according to Elman and Tuminaro [41], the performance of PCD is sensitive to the scale of the diagonal entries of $\mathbf{F}_p$ (note that they do not deal with boundary conditions for $\mathbf{A}_p$, since they use $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$). They suggest taking the value equal to the average of the diagonal values of $\mathbf{F}_p$ not corresponding to the inflow and outflow boundaries. We denote this choice as $\alpha = \alpha_{\text{avg}}$.

If Dirichlet boundary conditions are defined also for the matrix $\mathbf{A}_p$, we treat it in the same way as $\mathbf{F}_p$. That is, if $\alpha = \alpha_{\text{avg}}$ is used for $\mathbf{F}_p$, the value of diagonal entries in the modified rows of $\mathbf{A}_p$ is obtained analogously as the average of appropriate diagonal values of $\mathbf{A}_p$.

We compare results with the two mentioned choices of α for two selected choices of boundary conditions, $\text{PCD}^{\text{orig}}_{\text{bc1}}$ and $\text{PCD}^{\text{mod}}_{\text{bc4}}$, where Dirichlet conditions are set for both $\mathbf{A}_p$ and $\mathbf{F}_p$. We present results for the same set of discretizations as in Section 6.4.3. The results are shown in Table 6.5 for the steady-state problem and in Table 6.6 for the time-dependent problem. In both cases, the iteration counts increase with mesh refinement for $\alpha = 1$, especially for the 4-3, C^0 discretization. When $\alpha = \alpha_{\text{avg}}$ is used, the mesh dependence is eliminated (the iteration counts even decrease with mesh refinement in the steady case) and the convergence seems generally more robust with respect to the discretization. Thus, we set $\alpha = \alpha_{\text{avg}}$ in all experiments.

$\text{PCD}^{\text{orig}}_{\text{bc1}}$	**M1**	**M2**	**M3**
2-1, C^0	45	42	43
4-3, C^0	40	52	75
4-3, C^2	42	40	45
$\text{PCD}^{\text{mod}}_{\text{bc4}}$	**M1**	**M2**	**M3**
2-1, C^0	45	45	55
4-3, C^0	48	66	69
4-3, C^2	42	44	58

(a) $\alpha = 1$.

$\text{PCD}^{\text{orig}}_{\text{bc1}}$	**M1**	**M2**	**M3**
2-1, C^0	47	39	31
4-3, C^0	32	30	28
4-3, C^2	44	35	29
$\text{PCD}^{\text{mod}}_{\text{bc4}}$	**M1**	**M2**	**M3**
2-1, C^0	39	30	23
4-3, C^0	25	23	22
4-3, C^2	31	23	22

(b) $\alpha = \alpha_{\text{avg}}$.

Table 6.5: Iteration counts of $\text{PCD}^{\text{orig}}_{\text{bc1}}$ and $\text{PCD}^{\text{mod}}_{\text{bc4}}$ with two choices of diagonal value α in the rows modified due to Dirichlet boundary conditions for the steady-state BFS-2D problem with $\nu = 0.01$.

For illustration of convergence with the two values of α, we show the evolution of the relative residual norm during the GMRES iterations until convergence in Figure 6.16 for $\text{PCD}^{\text{mod}}_{\text{bc4}}$. The graphs are shown for the steady-state and time-dependent BFS-2D problem with 2-1, C^0 discretization. With $\alpha = 1$, there is a phase of a staircase-like convergence which expands with mesh refinement. This behavior is seen also for other discretizations. Moreover, GMRES with $\text{PCD}^{\text{orig}}_{\text{bc1}}$ almost stagnates at the beginning for steady-state problems and this stagnation phase again expands with mesh refinement.

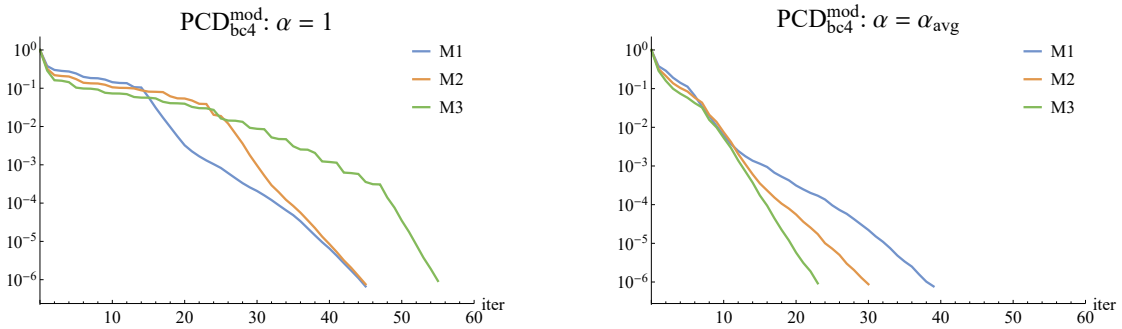
$\text{PCD}^{\text{orig}}_{\text{bc1}}$	**M1**	**M2**	**M3**
2-1, C^0	19	26	42
4-3, C^0	33	47	67
4-3, C^2	25	33	49
$\text{PCD}^{\text{mod}}_{\text{bc4}}$	**M1**	**M2**	**M3**
2-1, C^0	7	9	12
4-3, C^0	13	15	16
4-3, C^2	8	11	12

(a) $\alpha = 1$.

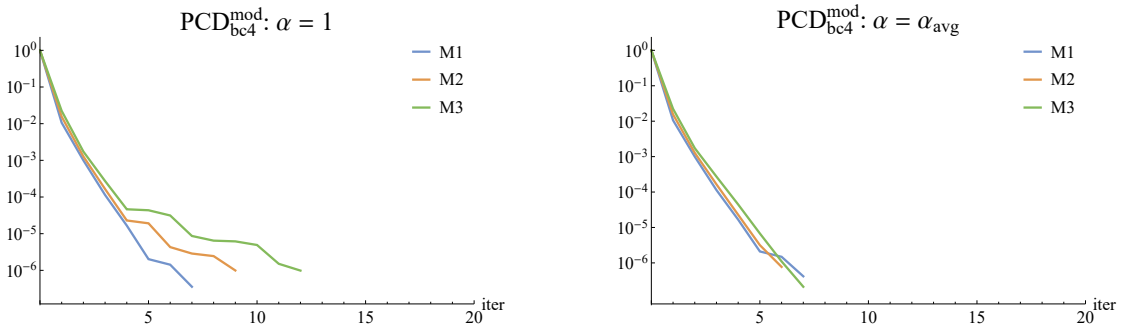
$\text{PCD}^{\text{orig}}_{\text{bc1}}$	**M1**	**M2**	**M3**
2-1, C^0	19	27	37
4-3, C^0	32	42	52
4-3, C^2	25	31	39
$\text{PCD}^{\text{mod}}_{\text{bc4}}$	**M1**	**M2**	**M3**
2-1, C^0	7	6	7
4-3, C^0	8	7	7
4-3, C^2	8	8	7

(b) $\alpha = \alpha_{\text{avg}}$.

Table 6.6: Iteration counts of $\text{PCD}^{\text{orig}}_{\text{bc1}}$ and $\text{PCD}^{\text{mod}}_{\text{bc4}}$ with two choices of diagonal value α in the rows modified due to Dirichlet boundary conditions for the time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$.



(a) The steady-state BFS-2D problem with $\nu = 0.01$.



(b) The time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$.

Figure 6.16: Convergence of $\text{PCD}^{\text{mod}}_{\text{bc4}}$ with Dirichlet diagonal values $\alpha = 1$ (left) and $\alpha = \alpha_{\text{avg}}$ (right) for the 2-1, C^0 discretization.

6.5.2 Comparison of different choices of BCs

In this section, we compare all variants of boundary conditions for PCD mentioned above. Tables 6.7 and 6.8 contain GMRES iteration counts for various discretizations and mesh refinements of the steady-state and time-dependent test problem, respectively. According to these results, $\text{PCD}^{\text{mod}}_{\text{bc4}}$ is clearly the best option. In the steady case, the original version $\text{PCD}^{\text{orig}}_{\text{bc1}}$ is also a good choice, it is robust with respect to mesh refinement

as well as the discretization degree and continuity. However, it shows mesh dependence in the time-dependent case. In general, the convergence of all variants with $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$ is strongly dependent on the discretization degree and especially its continuity.

In the steady case, the proposed variant with no boundary modification $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc6}}$ is obviously not applicable, since its convergence is slow and not robust with respect to mesh refinement. All other variants, including $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc5}}$, are more or less comparable for the low degree discretization 2-1, C^0. It indicates that $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc5}}$ could be successfully used also for the standard low-degree finite elements. Moreover, it seems that, unlike the related variants $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc2}}$ and $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc3}}$, its convergence does not slow down with mesh refinement and the iteration counts are generally lower.

In the time-dependent case, $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc4}}$ is the only variant which is robust with respect to both mesh refinement and discretization and it requires significantly less iterations than the other variants. The iteration counts of $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc1}}$ increase with mesh refinement for all discretizations. The variants $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc2}}$ and $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc3}}$ known from the literature seem mesh independent for the low-degree discretization 2-1, C^0. There is only a mild dependence for discretizations of the maximum continuity for the given degree, but the mesh dependence gets stronger with decreasing continuity. On the contrary, the iteration counts of $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc5}}$ and $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc6}}$ decrease with mesh refinement similarly to the steady case. The convergence of these two variants is mostly very similar. It can be concluded that the linear systems arising from time-dependent problems can be solved quite effectively without any boundary modification with $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc6}}$, if the maximum continuity discretization is used. The relative residual norm evolution for $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc1}}$ and $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc6}}$ for different uniform meshes is shown in Figure 6.17.

Figure 6.18 shows comparison of the relative residual norm evolution of all PCD BCs variants for the steady-state and time-dependent problems. Again, the graphs are presented for three selected discretizations 2-1, C^0, 5-4, C^0 and 5-4, C^3 and the mesh M4.

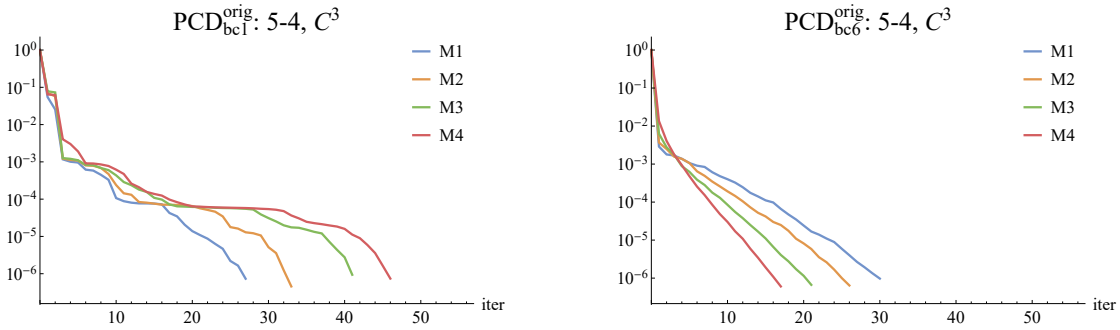


Figure 6.17: Mesh dependence of the original PCD, $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc1}}$, and the proposed variant with no boundary modification $\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc6}}$ for the time-dependent BFS-2D problem with 5-4, C^3 discretization.

Computational time

Finally, we present the wall-clock time of the computation for the individual PCD variants, i.e., the computational time of the whole solution of the given linear system. The total time is divided into two parts denoted as T_{setup} and T_{solve}. T_{setup} is the setup time of the preconditioner which involves factorization of the sub-blocks needed for direct solution of the inner linear systems (in the case of ideal version of PCD, which is con-

$\text{PCD}^{\text{orig}}_{\text{bc1}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	47	39	31	29
4-3, C^0	32	30	28	27
4-3, C^2	44	35	29	27
5-4, C^0	31	30	29	27
5-4, C^3	42	34	29	27

$\text{PCD}^{\text{mod}}_{\text{bc2}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	38	35	34	38
4-3, C^0	262	>300	>300	>300
4-3, C^2	66	69	75	79
5-4, C^0	>300	>300	>300	>300
5-4, C^3	98	99	109	114

$\text{PCD}^{\text{mod}}_{\text{bc3}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	44	39	37	38
4-3, C^0	265	>300	>300	>300
4-3, C^2	68	70	72	75
5-4, C^0	>300	>300	>300	>300
5-4, C^3	100	102	103	106

$\text{PCD}^{\text{mod}}_{\text{bc4}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	39	30	23	22
4-3, C^0	25	23	22	21
4-3, C^2	31	23	22	21
5-4, C^0	24	23	22	21
5-4, C^3	27	22	21	21

$\text{PCD}^{\text{mod}}_{\text{bc5}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	40	36	34	34
4-3, C^0	213	238	198	136
4-3, C^2	50	51	50	49
5-4, C^0	>300	>300	>300	>300
5-4, C^3	68	63	60	57

$\text{PCD}^{\text{orig}}_{\text{bc6}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	106	>300	>300	>300
4-3, C^0	>300	>300	>300	>300
4-3, C^2	109	155	>300	>300
5-4, C^0	>300	>300	>300	>300
5-4, C^3	124	155	>300	>300

Table 6.7: Comparison of all considered variants of boundary conditions for PCD, steady-state BFS-2D problem with $\nu = 0.01$. (Results for all discretizations in Appendix B, Table B.3.)

sidered here). T_{solve} is the time of the actual GMRES iterations. Note that the time of $\mathbf{A}_p$ and $\mathbf{F}_p$ assembly is not included. Times for the steady-state and time-dependent problem with the mesh M4 and selected discretizations are plotted in Figures 6.19 and 6.20, respectively. T_{solve} is displayed only for the cases where GMRES converged to the required tolerance.

Again, $\text{PCD}^{\text{mod}}_{\text{bc4}}$ appears as the most efficient variant of all. Only in a few cases, $\text{PCD}^{\text{orig}}_{\text{bc1}}$ took less time than $\text{PCD}^{\text{mod}}_{\text{bc4}}$, but their differences are minimal. Note that $\text{PCD}^{\text{mod}}_{\text{bc5}}$ and $\text{PCD}^{\text{orig}}_{\text{bc6}}$ took the second and third lowest time in the time-dependent case for the discretizations 2-1, C^0 and 5-4, C^3.

6.5.3 Mass matrix approximation in $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$

As we have seen in the previous section, the convergence of the PCD variants with $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$ depends on the discretization degree and continuity. In the case of LSC and MSIMPLER preconditioners, the situation was similar and using the lumped mass matrix approximation $\widehat{\mathbf{M}}_{u,L}$ instead of $\text{diag}(\mathbf{M}_u)$ helped with this issue.

In Table 6.9, we present a comparison of iteration counts for the relevant variants of

$\text{PCD}^{\text{orig}}_{\text{bc1}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	19	27	37	48
4-3, C^0	32	42	52	54
4-3, C^2	25	31	39	45
5-4, C^0	37	48	53	53
5-4, C^3	27	33	41	46

$\text{PCD}^{\text{mod}}_{\text{bc2}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	12	12	11	11
4-3, C^0	98	112	118	131
4-3, C^2	24	23	23	26
5-4, C^0	232	>300	>300	>300
5-4, C^3	33	32	33	37

$\text{PCD}^{\text{mod}}_{\text{bc3}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	17	19	19	16
4-3, C^0	122	149	158	160
4-3, C^2	32	35	37	39
5-4, C^0	284	>300	>300	>300
5-4, C^3	42	44	48	53

$\text{PCD}^{\text{mod}}_{\text{bc4}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	7	6	7	7
4-3, C^0	8	7	7	7
4-3, C^2	8	8	7	7
5-4, C^0	8	7	7	7
5-4, C^3	9	8	7	7

$\text{PCD}^{\text{mod}}_{\text{bc5}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	12	12	11	11
4-3, C^0	83	71	53	36
4-3, C^2	23	21	16	14
5-4, C^0	182	187	175	161
5-4, C^3	31	26	21	17

$\text{PCD}^{\text{orig}}_{\text{bc6}}$	**M1**	**M2**	**M3**	**M4**
2-1, C^0	12	12	11	11
4-3, C^0	82	70	53	38
4-3, C^2	23	20	17	15
5-4, C^0	181	184	172	>300
5-4, C^3	30	26	21	17

Table 6.8: Comparison of all considered variants of boundary conditions for PCD, time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$. (Results for all discretizations in Appendix B, Table B.4.)

PCD with $\widehat{\mathbf{M}}_u = \text{diag}(\mathbf{M}_u)$ and $\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$ on the mesh M1. Unlike LSC and MSIMPLER, the convergence with the lumped mass matrix approximation is even slower than with the diagonal of $\mathbf{M}_u$. Therefore, we include results with the exact mass matrix inverse, i.e. $\mathbf{A}_p = \mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T$, into the comparison to see if a more accurate approximation of the mass matrix inverse could potentially lead to better results. Indeed, the convergence is almost independent of the degree and continuity for all PCD variants in the time-dependent case and for $\text{PCD}^{\text{mod}}_{\text{bc2}}$, $\text{PCD}^{\text{mod}}_{\text{bc3}}$ and $\text{PCD}^{\text{mod}}_{\text{bc5}}$ also in the steady case. However, we were not able to find a suitable approximation of $\mathbf{M}_u^{-1}$ that would lead to degree-independent convergence. This could be a subject for future work.

6.5.4 Summary

Based on the results presented in this section, we recommend the following approach to the PCD boundary conditions for solving linear systems arising from IgA discretizations: the variant $\text{PCD}^{\text{mod}}_{\text{bc4}}$ combined with appropriate scaling of diagonal elements in $\mathbf{A}_p$ and $\mathbf{F}_p$ in the rows modified due to Dirichlet boundary conditions. For the sake of clarity, we summarize this approach again:

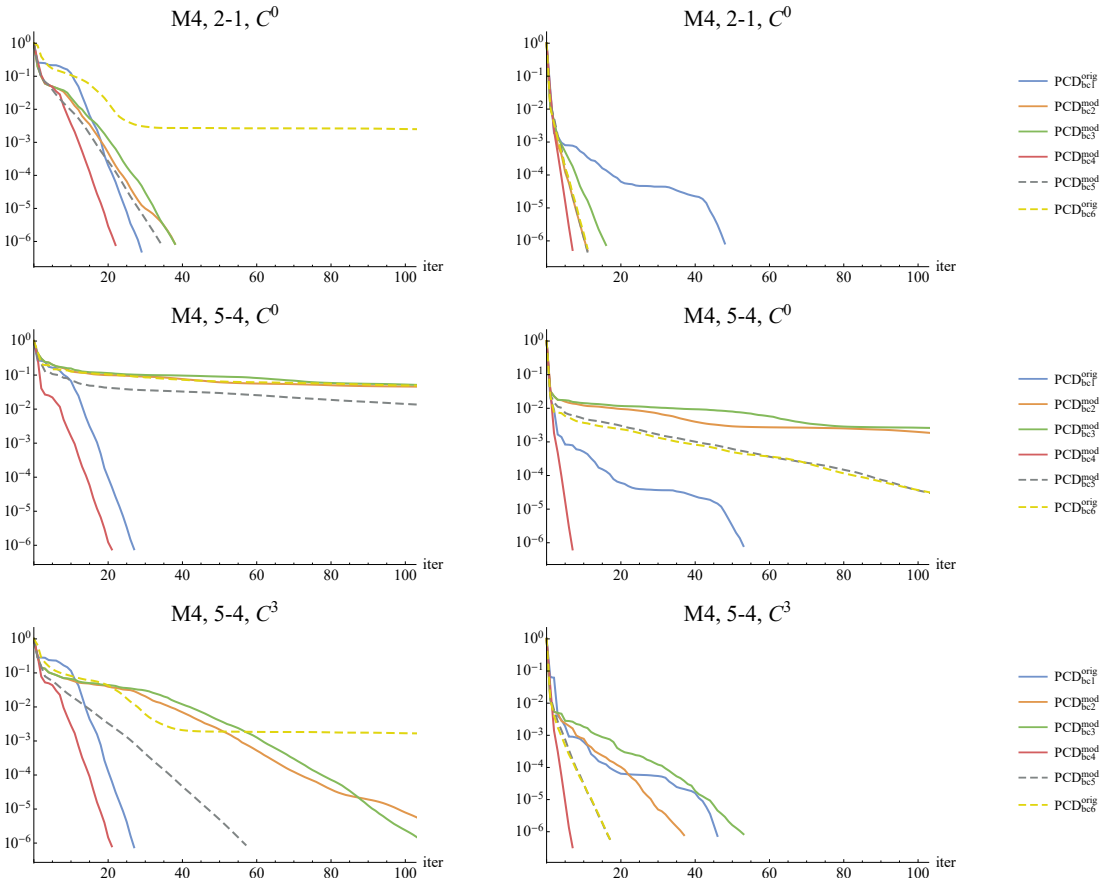


Figure 6.18: Comparison of different PCD BCs for three selected discretizations on the mesh M4, steady-state problem (left) and time-dependent problem (right).

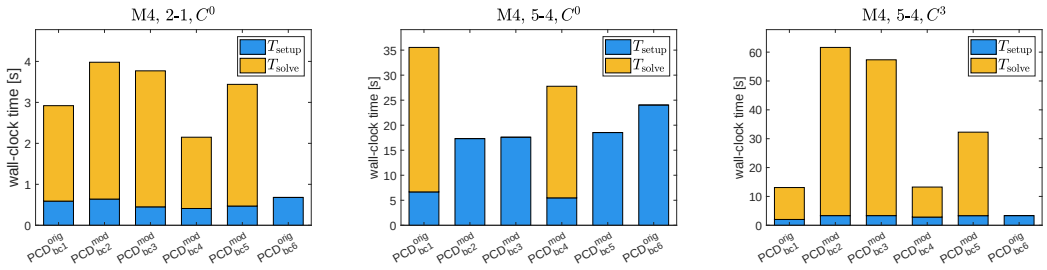


Figure 6.19: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for PCD with various BCs, steady-state BFS-2D problem with $\nu = 0.01$, mesh M4.

- $\mathbf{A}_p$ assembled using (5.36),
- $\mathbf{F}_p$ defined with Robin BCs on $\partial\Omega_{\mathrm{in}}$, see (5.44),
- Dirichlet BCs defined on $\partial\Omega_{\mathrm{out}}$ for both $\mathbf{A}_p$ and $\mathbf{F}_\mathbf{p}$, i.e., both matrices are modified such that the rows and columns corresponding to the pressure DOFs on $\partial\Omega_{\mathrm{out}}$ contain only diagonal entries,
- the value of these diagonal entries is determined as average of the diagonal entries

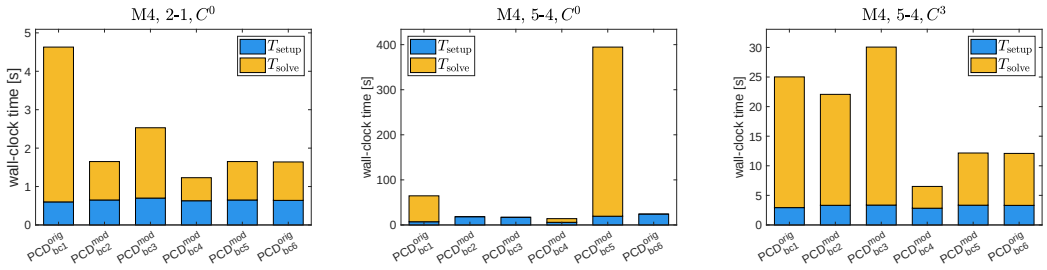


Figure 6.20: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for PCD with various BCs, time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$, mesh M4.

Mesh M1		**steady**			**unsteady**		
		2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3
$\widehat{\mathbf{M}}_u = \text{diag}(\mathbf{M}_u)$	$\text{PCD}^{\text{mod}}_{\text{bc2}}$	38	>300	98	12	232	33
	$\text{PCD}^{\text{mod}}_{\text{bc3}}$	44	>300	100	17	284	42
	$\text{PCD}^{\text{mod}}_{\text{bc5}}$	40	>300	68	12	182	31
	$\text{PCD}^{\text{orig}}_{\text{bc6}}$	106	>300	124	12	181	30
$\widehat{\mathbf{M}}_u = \widehat{\mathbf{M}}_{u,L}$	$\text{PCD}^{\text{mod}}_{\text{bc2}}$	39	>300	101	12	274	38
	$\text{PCD}^{\text{mod}}_{\text{bc3}}$	45	>300	107	14	284	46
	$\text{PCD}^{\text{mod}}_{\text{bc5}}$	41	>300	97	12	220	36
	$\text{PCD}^{\text{orig}}_{\text{bc6}}$	95	>300	201	12	219	36
$\widehat{\mathbf{M}}_u = \mathbf{M}_u$	$\text{PCD}^{\text{mod}}_{\text{bc2}}$	35	28	28	4	5	5
	$\text{PCD}^{\text{mod}}_{\text{bc3}}$	38	43	38	4	6	7
	$\text{PCD}^{\text{mod}}_{\text{bc5}}$	35	24	26	4	5	5
	$\text{PCD}^{\text{orig}}_{\text{bc6}}$	88	>300	83	4	6	6

Table 6.9: Comparison of different choices of $\widehat{\mathbf{M}}_u$ in PCD variants with $\mathbf{A}_p = \mathbf{B}\widehat{\mathbf{M}}_u^{-1}\mathbf{B}^T$ for BFS-2D with $\nu = 0.01$.

of the respective matrix not corresponding to the inflow and outflow boundaries,

- denote the resulting matrices as $\mathbf{A}_p^{D\text{out}}$ and $\mathbf{F}_p^{R\text{in},D\text{out}}$,
- the Schur complement approximation is given by $\widehat{\mathbf{S}} = -\mathbf{M}_p \left(\mathbf{F}_p^{R\text{in},D\text{out}}\right)^{-1} \mathbf{A}_p^{D\text{out}}$.

This approach is a combination of one of the PCD variants formulated in [9] and the scaling mentioned in [41], where it was considered for the diagonal entries corresponding to Dirichlet BCs in $\mathbf{F}_p$. To our knowledge, using an analogous scaling also for the matrix $\mathbf{A}_p$ is new.

According to our results, it seems that the variant from [9] without the scaling ($\text{PCD}^{\text{mod}}_{\text{bc4}}$ with $\alpha = 1$) shows some mesh dependence for all discretizations. The variants from [41] seem to perform relatively well for the low-degree discretization 2-1, C^0, but the convergence slows down for higher-degree IgA discretizations, especially for the C^0 continuity, which suggests that the performance may also worsen for higher-degree

standard finite elements. Only the combination of $\mathrm{PCD}^{\mathrm{mod}}_{\mathrm{bc4}}$ with the scaling ($\alpha = \alpha_{\mathrm{avg}}$) leads to a method which is robust with respect to the discretization as well as mesh refinement. We will use this combination in all further experiments, where we denote it shortly as "PCD".

Our experiments also indicate that if a suitable approximation of $\mathbf{M}_u^{-1}$ was found, the variants with $\mathbf{A}_p = \mathbf{B}\mathbf{M}_u^{-1}\mathbf{B}^T$ could also give a fast and robust convergence, moreover, without any boundary modification in the time-dependent case ($\mathrm{PCD}^{\mathrm{orig}}_{\mathrm{bc6}}$). However, we note that this choice of $\mathbf{A}_p$ is denser and thus leads to higher memory requirements than if $\mathbf{A}_p$ is assembled as a discrete Poisson operator.

6.6 Comparison of ideal versions of block preconditioners

The main comparison of ideal versions of all considered block preconditioners for all test problems is presented in this section. For each test problem, we limit ourselves to three discretizations: 2-1, C^0 that can serve as an analogue of the standard low-degree finite elements, and the discretizations of the highest degree k considered for the given test problem with C^0 and C^{k-1} continuity. As already mentioned, we refer to Appendix B for complete results involving the discretizations omitted here. If a parameter was not considered for a given problem, the corresponding position in the table is filled with a dash (—).

First, we comment on the choice of the parameter γ in the AL-based preconditioners for the individual test problems. The rest of the section is divided into three parts: the first two are devoted to the steady-state and time-dependent two-dimensional test problems and the last one deals with the three-dimensional test problems. We focus on the convergence behavior of the block preconditioners depending on different aspects: uniform mesh refinement and viscosity value in both 2D and 3D and mesh stretching in 2D.

6.6.1 Parameter γ for AL and MAL

From our experience, the convergence of AL depends on γ in a straightforward way – the larger the value of γ, the faster the convergence. For a large enough γ, GMRES preconditioned with AL usually converges in just a few iterations. However, due to the ill-conditioning of the augmented matrix $\mathbf{F}_\gamma$ for large γ and thus an increasing influence of rounding errors, the obtained solution becomes useless for γ too large. Therefore, we choose a moderate value of the parameter such that the convergence is already fast and the solution error is acceptable at the same time. (We assess the "error" by comparing the obtained solution with a direct solution of the non-augmented linear system.) Usually, a larger γ is needed for time-dependent problems to achieve a comparably fast convergence as in the steady case.

The dependence of MAL convergence on γ is not so straightforward. There is usually some optimal value for which the iteration count is minimal and that can differ depending on the problem at hand and its parameters, discretization, etc. We investigated the optimal choice for IgA discretizations of the 2D backward facing step problem in [31]. It appears that the optimal value of γ is rather small (at least for the steady-state problem) and it does not differ much for different IgA discretizations. Here we do not go into much detail regarding the optimal choice of γ for MAL.

Based on a set of experiments for each 2D test problem, we have chosen the values of γ for AL and MAL shown in Table 6.10 and we use the same value for the corresponding 3D test problems.

γ	**AL**		**MAL**	
	st.	unst.	st.	unst.
LDC	2	10	0.02	0.2
BFS	2	10	0.1	10
TB	2	10	0.1	0.1

Table 6.10: Values of γ chosen for individual test problems (the same value used for 2D and 3D).

6.6.2 Steady-state 2D test problems

Uniform mesh refinement

A comparison of iteration counts for the steady-state LDC-2D, BFS-2D and TB-2D problems for various uniformly refined meshes M1 to M4 and a fixed viscosity is presented in Table 6.11. Note that only three meshes, M1 to M3, were considered for the AL-based preconditioners because of high computational and especially memory requirements of their ideal version.

Generally, we observe the expected convergence properties for all discretizations of the test problems. LSC and MSIMPLER preconditioners give very similar iteration counts and it seems that their convergence depends on the mesh refinement roughly as $O(h^{-1/2})$. The convergence of PCD is independent of h, except for some deterioration for TB-2D with coarse meshes, where the iteration counts decrease with mesh refinement at first. In the case of lid-driven cavity problem, there seems to be a jump from approximately 50 iterations to around 30 iterations at some point when the mesh is refined. This is caused by a stagnation phase that occurs in a later iteration for finer meshes, see Figure 6.21 for plots of convergence curves. From the iteration counts point of view, the convergence of SIMPLE is significantly slower than the other preconditioners. Its mesh dependence is of order $O(h^{-1/2})$, except for the high-degree discretizations with low continuity, where the iteration counts seem to be almost independent of h. The AL preconditioner requires a very low number of iterations for all problems and is robust with respect to the mesh refinement and discretization. The iteration counts of MAL are significantly higher than for AL and they seem to increase with the discretization degree and continuity. Of course, this can be caused by a non-optimal choice of the parameter γ. At least it seems that it is possible to choose γ such that the convergence of MAL is independent (or almost independent) of h.

Interesting is the behavior of SIMPLER. For high-degree C^0 discretizations, the iteration counts increase approximately as $O(h^{-1})$, whereas for the other discretizations, they are almost independent of the refinement at first and seem to tend to the order $O(h^{-1/2})$ or even decrease for the tested meshes considered for TB-2D. This is very similar to the behavior of LSC and MSIMPLER without mass lumping investigated in Section 6.4.1. Thus, a modification of the diagonal approximation $\mathbf{D} = \mathrm{diag}(\mathbf{F})$ might be needed. However, lumping is not suitable in this case (which our experiments confirmed). Figures 6.22 and 6.23 show the convergence curves of SIMPLER for BFS-2D and TB-2D,

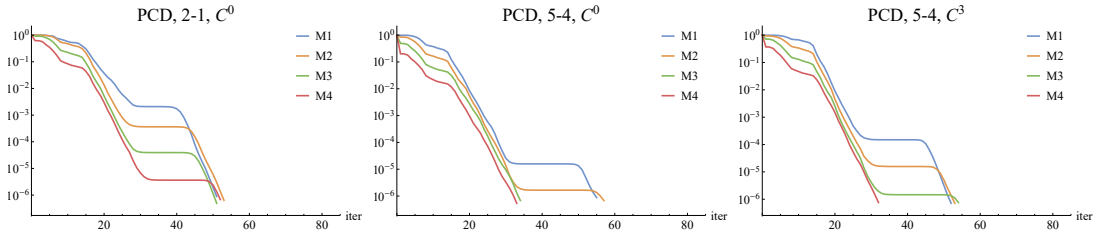


Figure 6.21: PCD convergence curves (relative residual norm) for the steady-state LDC-2D problem with $\nu = 0.003$ and selected discretizations, comparison of uniform meshes.

respectively. Apparently, the GMRES method stagnates at the beginning of the iteration process. This is typical for the SIMPLER preconditioner, as mentioned for example in [107]. The stagnation phase expands with mesh refinement and this expansion is by far most significant in the 5-4, C^0 case for BFS-2D and 4-3, C^0 for TB-2D.

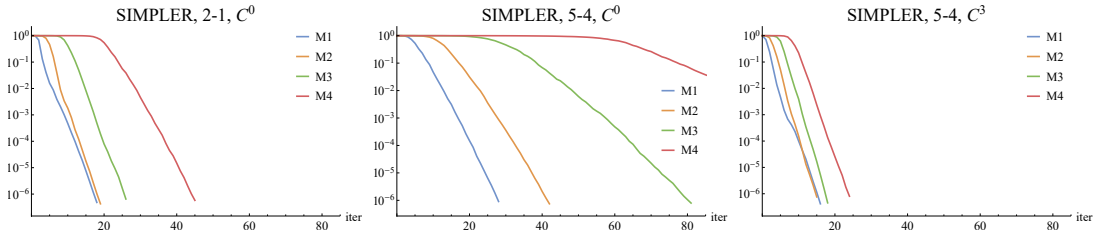


Figure 6.22: SIMPLER convergence curves (relative residual norm) for the steady-state BFS-2D problem with $\nu = 0.02$ and selected discretizations, comparison of uniform meshes.

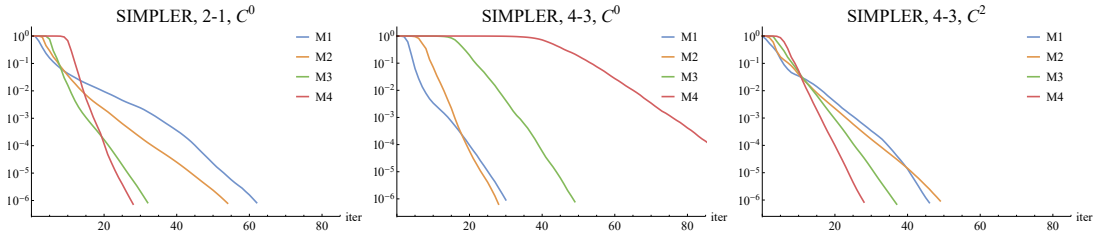


Figure 6.23: SIMPLER convergence curves (relative residual norm) for the steady-state TB-2D problem with $\nu = 0.01$ and selected discretizations, comparison of uniform meshes.

Also notice that the iteration counts of PCD depend quite significantly on the discretization for TB-2D with coarse meshes M1 and M2. Surprisingly, the fastest convergence is obtained for the 4-3, C^0 discretization. The iteration counts for the remaining discretizations decrease gradually with mesh refinement. The convergence curves are plotted in Figure 6.24. We do not have a theoretical explanation for this behavior, nevertheless, note that the discretizations 2-1, C^0 and 4-3, C^0 correspond to a linearized geometry. Thus, the computational domain of the underlying problem differs from that with the 4-3, C^2 discretization and the finer the mesh, the closer the two domains are to each other.

We present the computational time for all preconditioners for the BFS-2D problem with the mesh M3 in Figure 6.25. Note that the performance of our codes is not optimized

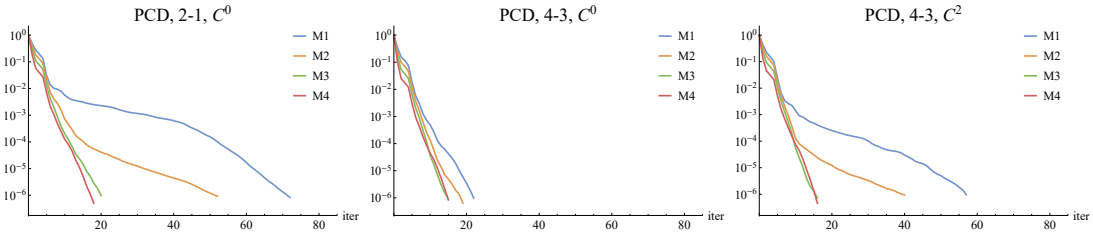


Figure 6.24: PCD convergence curves (relative residual norm) for the steady-state TB-2D problem with $\nu = 0.01$ and selected discretizations, comparison of uniform meshes.

and the preconditioners can probably be implemented more efficiently. Yet, these graphs are useful for a rough comparison of the individual methods. Obviously, the setup of the ideal versions of AL and also MAL is significantly more expensive than of the rest of the preconditioners, which is due to factorization of the augmented matrix $\mathbf{F}_\gamma$ or its diagonal blocks. Thus, efficient solution of these subproblems is really essential for efficiency of these methods. On the other hand, the solution with the SIMPLE preconditioner takes long due to many iterations needed for convergence leading to high T_{solve}. PCD seems as the most efficient of the tested methods, but let us remind that the assembly of $\mathbf{A}_p$ and $\mathbf{F}_p$ is not included in the displayed computational time. More comparisons of computational time for all test problems with the finest uniform mesh M4 are included in Appendix B.

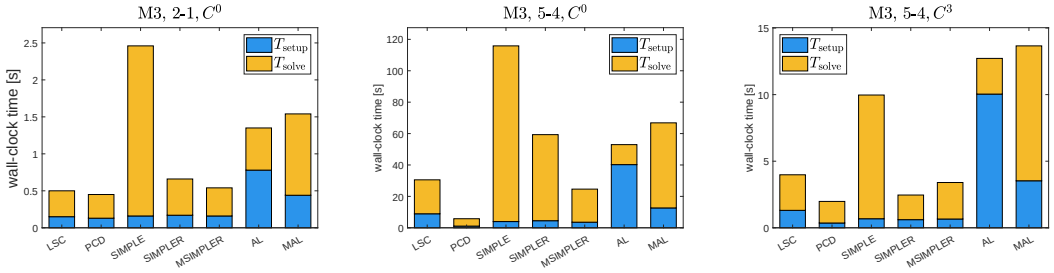


Figure 6.25: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the steady-state BFS-2D problem with $\nu = 0.02$ on the mesh M3.

Mesh stretching

We compare the iteration counts for the steady state 2D problems with stretched meshes in Table 6.12. Let us denote the maximum aspect ratio of the elements in a given mesh (in the parametric space) as $r_{\max}$. Then $r_{\max} = 1$ for the uniform mesh and $r_{\max} = 4, 16$ and 64 for the stretched meshes SM1, SM2 and SM3, respectively.

The convergence slows down with increasing maximum aspect ratio for LSC, SIMPLE and MSIMPLER, and for both inflow/outflow problems also for SIMPLER. The strongest dependence (approximately $O(r_{\max}^{1/3})$ to $O(r_{\max}^{1/2})$) is observed for LSC and MSIMPLER for the LDC-2D problem. On the other hand, SIMPLER gives convergence almost independent of the aspect ratio when used for the LDC-2D problem. The iteration counts of PCD and AL are independent of the aspect ratio. The results for MAL indicate, that γ can be chosen such that the convergence also does not depend on $r_{\max}$.

	LDC-2D			BFS-2D			TB-2D		
LSC	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	26	43	27	18	25	19	43	22	31
M2	30	44	33	18	27	21	29	19	24
M3	32	47	40	21	31	28	21	24	22
M4	41	53	53	29	38	38	20	30	27
PCD	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	51	55	52	23	22	20	72	22	57
M2	53	57	53	21	20	19	52	19	40
M3	51	34	54	20	19	19	20	15	16
M4	52	33	32	19	18	18	18	15	16
SIMPLE	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	80	>300	88	69	271	91	85	>300	147
M2	118	>300	97	97	275	85	108	272	139
M3	152	>300	126	124	246	108	131	210	135
M4	206	>300	172	164	247	146	169	268	173
SIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	24	34	22	18	28	16	62	30	46
M2	25	45	22	19	42	15	54	28	49
M3	26	80	23	26	81	18	32	49	37
M4	41	146	31	45	151	24	28	106	28
MSIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	24	42	25	17	25	18	49	28	33
M2	27	42	31	17	27	20	36	24	34
M3	29	45	38	20	31	27	27	27	31
M4	39	52	51	27	37	36	23	35	30
AL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	6	8	6	5	7	6	4	5
M2	5	5	4	6	4	5	5	4	4
M3	5	5	3	6	4	4	5	4	4
MAL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	29	252	128	24	85	77	43	56	56
M2	32	243	125	21	61	44	43	61	43
M3	36	129	121	20	41	29	44	61	50

Table 6.11: Comparison of block preconditioners for the steady-state two-dimensional problems with uniform meshes. LDC-2D with $\nu = 0.003$ ($Re \approx 333$), BFS-2D with $\nu = 0.02$ ($Re = 100$), TB-2D with $\nu = 0.01$ ($Re \approx 298$). (Results for all discretizations of LDC-2D and BFS-2D in Appendix B, Tables B.5 and B.11, respectively.)

	LDC-2D			BFS-2D			TB-2D		
LSC	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	26	43	27	18	25	19	43	22	31
SM1	38	51	42	20	42	24	52	37	39
SM2	62	80	71	24	66	34	68	58	59
SM3	88	101	100	30	72	48	80	79	73
PCD	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	51	55	52	23	22	20	72	22	57
SM1	54	33	55	22	21	20	73	22	57
SM2	53	31	31	22	21	19	73	22	58
SM3	31	25	28	22	21	19	73	22	58
SIMPLE	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	80	>300	88	69	271	91	85	>300	147
SM1	90	>300	80	89	>300	101	147	>300	201
SM2	106	>300	88	104	>300	128	192	>300	244
SM3	112	295	96	119	>300	146	225	>300	281
SIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	24	34	22	18	28	16	62	30	46
SM1	24	32	20	20	51	22	69	39	58
SM2	23	37	20	30	78	31	78	51	65
SM3	22	36	20	40	94	37	81	59	67
MSIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	24	42	25	17	25	18	49	28	33
SM1	36	49	40	19	41	23	61	45	49
SM2	59	80	71	25	66	35	78	71	72
SM3	86	106	99	31	73	52	90	95	89
AL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	6	8	6	5	7	6	4	5
SM1	5	5	3	6	5	6	6	4	5
SM2	3	3	3	6	5	6	6	4	5
MAL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	29	252	128	24	85	77	43	56	56
SM1	30	116	117	23	51	39	43	60	52
SM2	31	75	46	23	34	32	43	60	53

Table 6.12: Comparison of block preconditioners for the steady-state two-dimensional problems with stretched meshes. LDC-2D with $\nu = 0.003$ ($Re \approx 333$), BFS-2D with $\nu = 0.02$ ($Re = 100$), TB-2D with $\nu = 0.01$ ($Re \approx 298$). (Results for all discretizations of LDC-2D and BFS-2D in Appendix B, Tables B.6 and B.12, respectively.)

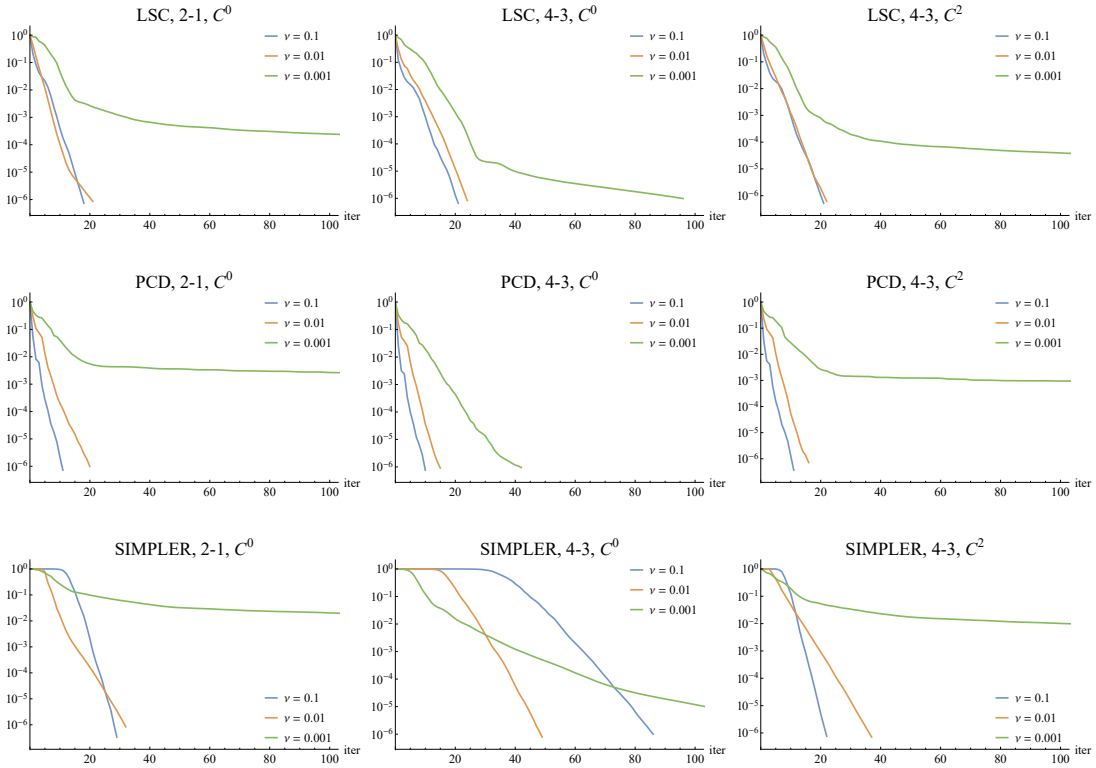


Figure 6.26: LSC, PCD and SIMPLER convergence curves (relative residual norm) for the steady-state TB-2D problem with the mesh M3 and selected discretizations, comparison of different viscosity values.

Viscosity dependence

The comparison for various viscosity values on a fixed mesh (M3 for each test problem) is presented in Table 6.13. The only preconditioner that gives ν-independent convergence is AL. From the results, it is not clear how to quantify the dependence on ν for the rest of the preconditioners. Interestingly, it seems to be somewhat weaker for the C^0 high-degree discretizations in many cases. For more insight, we show the convergence curves for LSC, PCD and SIMPLER for the TB-2D problem in Figure 6.26. For LSC and PCD (it is true also for MSIMPLER), it can be seen that the convergence does not differ much for different discretizations if the viscosity is large enough. However, for the smallest viscosity, the convergence slows down dramatically or almost stagnates from some iteration on. For the 4-3, C^0 discretization, this happens at a later iteration compared to the other two discretizations. For SIMPLER, it seems that the initial stagnation phase tends to shorten with decreasing viscosity at first, but then the convergence slows down.

If we look at the solution of the Navier–Stokes problem at the Picard iteration, where the linear systems were obtained, we find the likely reason for this behavior. Figure 6.27 shows the pressure for the TB-2D problem with the selected discretizations. The solutions obtained with discretizations 2-1, C^0 and 4-3, C^2 are polluted by spurious oscillations. Apparently, the viscosity is too low for these discretizations on the considered mesh, and thus, the solution becomes unstable. The performance of the all tested

	LDC-2D			BFS-2D			TB-2D		
LSC	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	14	16	16	25	30	30	18	21	21
ν_2	18	22	22	24	34	31	21	24	22
ν_3	32	47	40	68	49	64	>300	96	>300
ν_4	184	142	137	—	—	—	—	—	—
PCD	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	11	10	10	22	21	21	11	10	11
ν_2	18	17	17	20	20	19	20	15	16
ν_3	51	34	54	80	44	58	>300	42	>300
ν_4	262	137	213	—	—	—	—	—	—
SIMPLE	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	67	118	54	90	143	84	100	147	91
ν_2	72	127	59	165	297	141	131	210	135
ν_3	152	>300	126	>300	>300	>300	>300	>300	>300
ν_4	>300	>300	>300	—	—	—	—	—	—
SIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	23	63	13	38	95	28	29	86	22
ν_2	25	66	15	31	106	21	32	49	37
ν_3	26	80	23	88	91	73	>300	142	>300
ν_4	149	95	130	—	—	—	—	—	—
MSIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	18	23	23	26	33	32	22	26	25
ν_2	20	25	25	23	34	30	27	27	31
ν_3	29	45	38	69	48	65	>300	147	>300
ν_4	173	137	134	—	—	—	—	—	—
AL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	5	27	18	9	—	7	5	3	4
ν_2	4	12	8	6	—	5	5	4	4
ν_3	5	5	3	5	—	4	5	4	4
ν_4	5	3	3	—	—	—	—	—	—
MAL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	18	178	99	30	—	58	27	28	25
ν_2	18	169	90	27	—	29	44	61	50
ν_3	36	129	121	50	—	79	153	>300	200
ν_4	147	>300	265	—	—	—	—	—	—

Table 6.13: Comparison of block preconditioners for the steady-state two-dimensional problems with various viscosity values on the mesh M3. (Results for all discretizations of LDC-2D and BFS-2D in Appendix B, Tables B.7 and B.13, respectively.)
LDC-2D: $\nu_1 = 0.3$, $\nu_2 = 0.03$, $\nu_3 = 0.003$, $\nu_4 = 0.0003$; BFS-2D: $\nu_1 = 0.2$, $\nu_2 = 0.02$, $\nu_3 = 0.002$; TB-2D: $\nu_1 = 0.1$, $\nu_2 = 0.01$, $\nu_3 = 0.001$.

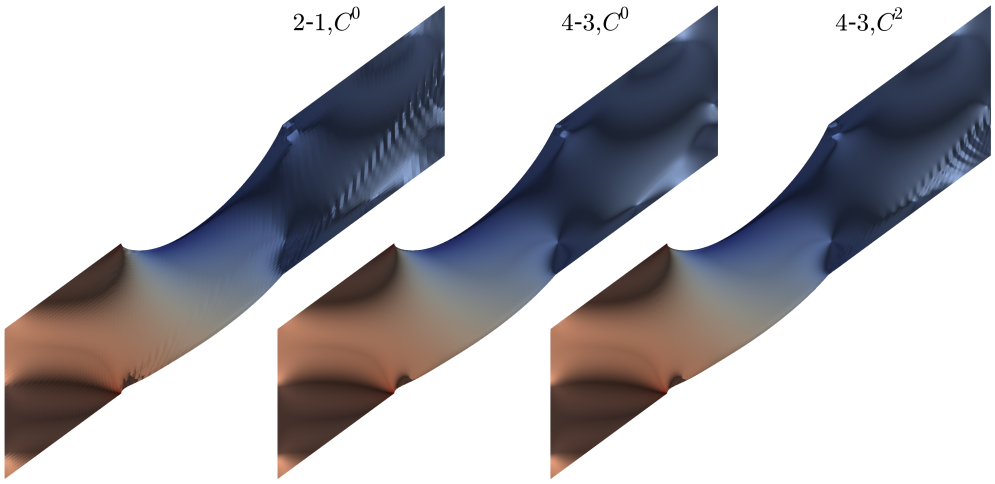


Figure 6.27: Pressure solution for the TB-2D problem with $\nu = 0.001$ for different discretizations on the mesh M3.

preconditioners except AL is obviously sensitive to such instability.

Summary

We have observed the expected behavior in the majority of cases for all considered IgA discretizations. One of the exceptions is SIMPLER, which seems not suitable for higher-degree C^0 discretizations due to relatively strong mesh dependence of its convergence. This observation indicates, that it may not perform well also for the standard higher-degree finite elements that are in some aspects similar to the C^0 IgA discretizations.

From the iteration count point of view, AL seems as an optimal preconditioner. GMRES with AL preconditioning converges in a few iterations independently of the problem parameters and discretization. However, the setup of its ideal version is several times more expensive compared to the other preconditioners and finding a fast approximate solver for the augmented block $\mathbf{F}_\gamma$ is not a trivial task. A disadvantage of the modified AL approach (MAL) is its sensitivity to the parameter γ, which is generally not very well predictable. It seems that convergence independent of the problem parameters can be achieved for MAL, but γ really needs to be optimized and the convergence is significantly slower than for AL even for γ close to the optimal value.

Based on our results, we recommend using the variant of PCD described in 6.5.4 for isogeometric discretizations of the steady-state Navier–Stokes equations. Its performance is robust with respect to the uniform mesh refinement as well as the aspect ratio of the elements. Moreover, PCD seems to be the most efficient of the tested preconditioners from the computational time point of view.

Eigenvalues of the preconditioned matrix

Although we are aware that the spectrum of the preconditioned matrix does not determine the convergence of GMRES, it might be interesting to look at the eigenvalue plots. We consider the steady-state BFS-2D problem with $\nu = 0.01$, three selected discretizations 2-1, C^0, 4-3, C^0 and 4-3, C^2 and three meshes. These meshes are denoted as "ref0", "ref1" and "ref2". Note that "ref0" and "ref1" are coarser than the meshes considered

in other experiments and "ref2" corresponds to M1.

For illustration, we display the eigenvalues for two preconditioners, PCD and SIMPLE, and refer to Appendix B, Figures B.5 to B.9 for eigenvalue plots for the other preconditioners. We have chosen these two preconditioners for presentation, because the convergence of PCD is robust with respect to the mesh refinement as well as the discretization and, on the other hand, SIMPLE gives convergence dependent on both.

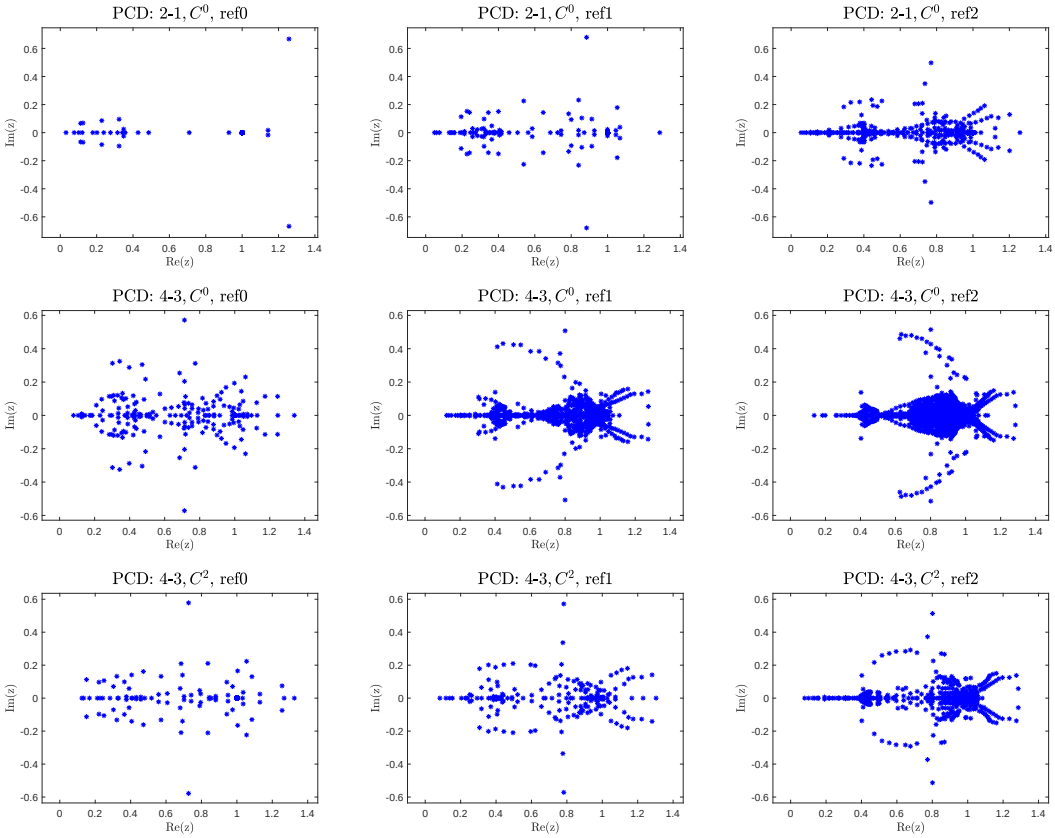


Figure 6.28: Eigenvalues of the preconditioned matrix (PCD), steady-state BFS-2D.

6.6.3 Time-dependent 2D test problems

Uniform mesh refinement

Table 6.14 shows the comparison of iteration counts for the time-dependent 2D problems with uniform meshes. Obviously, the convergence is generally faster than for the steady-state problems, which can be expected due to the presence of the velocity mass matrix in the block **F**.

The performance of all preconditioners except SIMPLE is essentially independent of the mesh parameter (the convergence of MAL could be improved by tuning the parameter γ) for all discretizations of the LDC-2D and BFS-2D problems. Their iteration counts are usually comparable for the discretizations of maximum continuity and slightly increase with decreasing continuity for discretizations of a given degree. This does not happen only for AL and PCD. The iteration counts of SIMPLE decrease with mesh refinement for these two test problems.

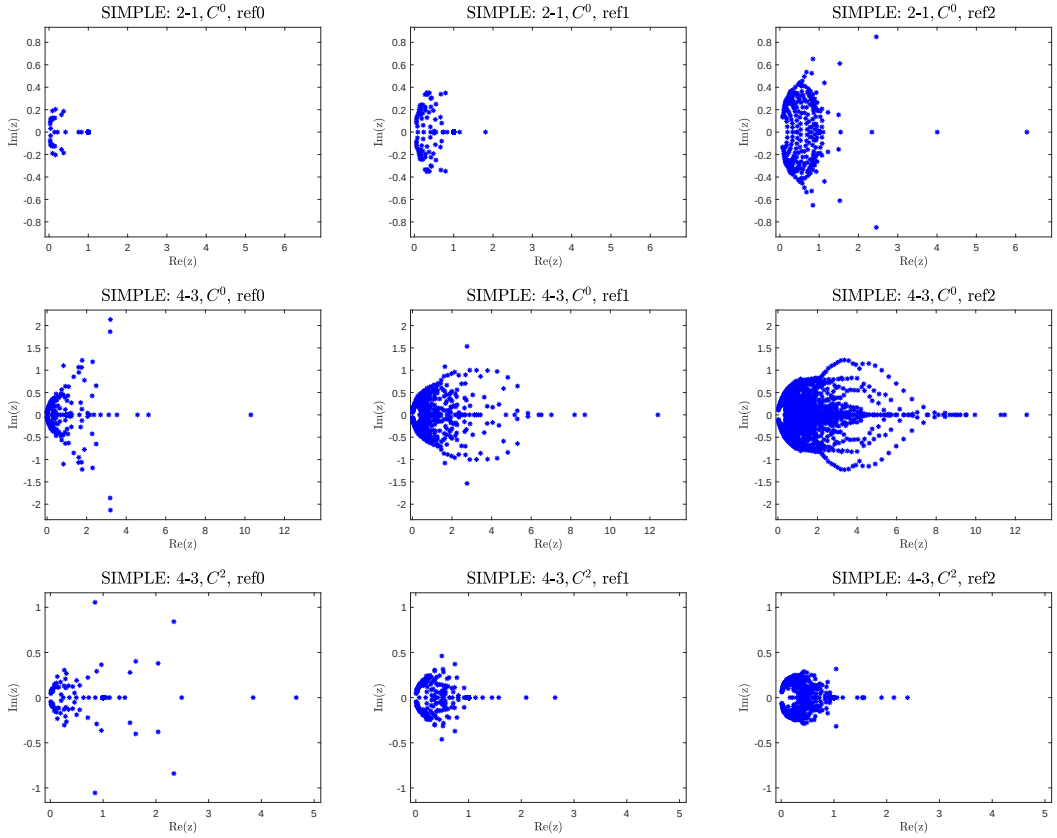


Figure 6.29: Eigenvalues of the preconditioned matrix (SIMPLE), steady-state BFS-2D.

The situation is a bit different for the TB-2D problem. The iteration counts are generally higher than for the two academic problems (except for AL) and mesh dependence is observed for LSC and SIMPLE-type preconditioners for some discretizations, most significantly when SIMPLER is used for the 4-3, C^0 discretization. As can be expected, the convergence accelerates with decreasing time step size. However, the mesh dependence is still evident.

We present computational times for the time-dependent BFS-2D problem with the mesh M3 in Figure 6.30. Again, PCD comes out as the most effective, especially for higher-degree discretizations. For the high-degree discretizations with maximum continuity, SIMPLER and MSIMPLER seem also as a good choice. More computational time comparisons are presented in Appendix B.

Mesh stretching and viscosity dependence

See Table 6.15 for comparison of the time-dependent 2D test problems with stretched meshes. In this case, we observe similar behavior as for the steady-state problems.

Results for various values of viscosity are contained in Table 6.16. Again, the iteration counts for all preconditioners except SIMPLE and MAL seem almost independent of ν for the LDC-2D and BFS-2D test problems. In the case of TB-2D, the behavior is similar to the steady-state problem, probably for the same reason.

	LDC-2D			BFS-2D			TB-2D		
LSC	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	14	6	5	14	7	26	16	18
M2	5	13	5	5	12	6	20	16	17
M3	4	13	5	4	12	6	15	20	18
M4	5	13	6	4	13	6	17	25	22
PCD	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	8	11	8	7	8	8	44	16	31
M2	8	12	9	6	7	7	31	15	20
M3	10	13	10	7	7	6	15	13	13
M4	11	13	11	7	7	7	14	12	13
SIMPLE	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	11	183	28	8	140	26	58	239	88
M2	10	162	25	7	112	20	80	217	98
M3	7	118	19	5	74	15	101	172	107
M4	12	105	15	4	61	10	137	224	143
SIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	20	7	6	19	9	34	26	31
M2	5	19	7	5	17	7	35	22	29
M3	4	19	7	4	16	7	27	36	30
M4	4	19	6	4	17	7	22	83	25
MSIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	17	7	6	18	9	30	23	24
M2	5	15	7	5	16	7	27	20	27
M3	5	15	6	4	15	6	24	23	27
M4	5	16	7	4	16	7	19	29	25
AL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	3	4	17	12	12	5	4	4
M2	5	3	3	17	11	12	4	3	4
M3	4	3	3	17	9	12	4	3	3
MAL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	28	32	32	20	30	20	40	55	54
M2	30	39	18	25	42	25	39	53	43
M3	34	54	21	31	54	31	42	56	43

Table 6.14: Comparison of block preconditioners for the time-dependent two-dimensional problems with uniform meshes. LDC-2D with $\nu = 0.003$ ($Re \approx 333$), BFS-2D with $\nu = 0.02$ ($Re = 100$), TB-2D with $\nu = 0.01$ ($Re \approx 298$), time step $\Delta t = 0.01$ for all problems. (Results for all discretizations of LDC-2D and BFS-2D in Appendix B, Tables B.8 and B.14, respectively.)

	LDC-2D			BFS-2D			TB-2D		
LSC	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	14	6	5	14	7	26	16	18
SM1	4	12	5	5	23	8	30	28	24
SM2	6	16	7	5	36	9	41	42	40
SM3	9	18	11	8	40	13	53	60	52
PCD	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	8	11	8	7	8	8	44	16	31
SM1	9	11	9	6	6	7	43	16	33
SM2	10	11	9	6	6	6	43	17	33
SM3	10	10	9	6	6	6	43	17	33
SIMPLE	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	11	183	28	8	140	26	58	239	88
SM1	10	165	22	8	137	23	101	290	116
SM2	17	160	20	12	141	24	138	>300	154
SM3	27	172	25	22	159	30	163	>300	181
SIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	20	7	6	19	9	34	26	31
SM1	4	16	5	6	30	9	43	33	41
SM2	5	21	5	8	40	11	51	44	49
SM3	6	24	5	19	51	23	55	51	52
MSIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	17	7	6	18	9	30	23	24
SM1	5	14	5	5	28	9	37	34	30
SM2	7	22	9	6	44	11	49	52	51
SM3	12	35	17	10	48	16	61	73	66
AL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	5	3	4	17	12	12	5	4	4
SM1	4	3	3	18	12	12	5	4	4
SM2	4	2	3	18	12	12	5	4	4
MAL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
M1	28	32	32	20	30	20	40	55	54
SM1	30	40	16	30	59	30	42	57	51
SM2	22	55	17	41	78	42	43	58	51

Table 6.15: Comparison of block preconditioners for the time-dependent two-dimensional problems with stretched meshes. LDC-2D with $\nu = 0.003$ ($Re \approx 333$), BFS-2D with $\nu = 0.02$ ($Re = 100$), TB-2D with $\nu = 0.01$ ($Re \approx 298$), time step $\Delta t = 0.01$ for all problems. (Results for all discretizations of LDC-2D and BFS-2D in Appendix B, Tables B.9 and B.15, respectively.)

	LDC-2D			BFS-2D			TB-2D		
LSC	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	7	11	9	5	13	7	15	18	17
ν_2	5	13	6	4	12	6	15	20	18
ν_3	4	13	5	5	14	6	294	40	148
ν_4	10	28	8	—	—	—	—	—	—
PCD	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	9	9	9	8	8	7	10	9	9
ν_2	10	11	10	7	7	6	15	13	13
ν_3	10	13	10	6	7	9	>300	24	>300
ν_4	12	15	11	—	—	—	—	—	—
SIMPLE	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	45	98	35	8	72	14	87	132	78
ν_2	16	97	14	5	74	15	101	172	107
ν_3	7	118	19	7	234	22	>300	>300	>300
ν_4	12	>300	38	—	—	—	—	—	—
SIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	11	42	7	5	19	7	23	75	17
ν_2	6	19	6	4	16	7	27	36	30
ν_3	4	19	7	6	20	8	>300	108	>300
ν_4	8	39	11	—	—	—	—	—	—
MSIMPLER	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	10	18	13	5	17	8	18	23	21
ν_2	6	17	8	4	15	6	24	23	27
ν_3	5	15	6	5	16	7	>300	111	238
ν_4	9	32	9	—	—	—	—	—	—
AL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	4	7	7	13	8	11	4	3	3
ν_2	4	3	3	17	9	12	4	3	3
ν_3	4	3	3	17	10	12	4	3	3
ν_4	4	3	3	—	—	—	—	—	—
MAL	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	5-4, C^0	5-4, C^3	2-1, C^0	4-3, C^0	4-3, C^2
ν_1	19	95	56	20	35	22	32	29	27
ν_2	20	61	31	31	54	31	42	56	43
ν_3	34	54	21	32	86	38	107	268	131
ν_4	39	145	39	—	—	—	—	—	—

Table 6.16: Comparison of block preconditioners for the time-dependent two-dimensional problems with various viscosity values on the mesh M3. (Results for all discretizations of LDC-2D and BFS-2D in Appendix B, Tables B.10 and B.16, respectively.)
LDC-2D: $\nu_1 = 0.3$, $\nu_2 = 0.03$, $\nu_3 = 0.003$, $\nu_4 = 0.0003$; BFS-2D: $\nu_1 = 0.2$, $\nu_2 = 0.02$, $\nu_3 = 0.002$; TB-2D: $\nu_1 = 0.1$, $\nu_2 = 0.01$, $\nu_3 = 0.001$.

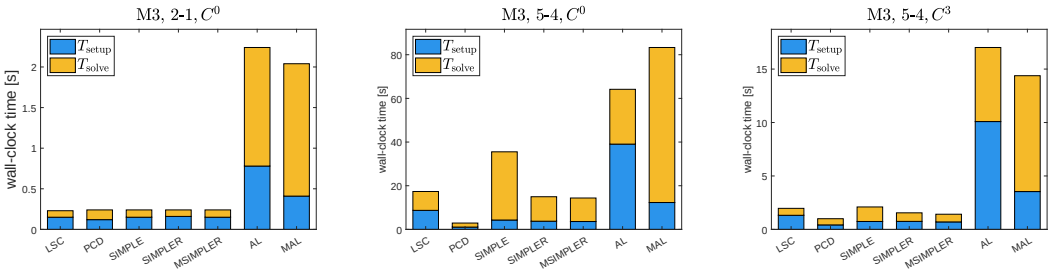


Figure 6.30: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the time-dependent BFS-2D problem with $\nu = 0.02$ and $\Delta t = 0.01$ on the mesh M3.

Summary

In the time-dependent case, the same conclusions regarding AL can be drawn as for the steady case.

Thanks to the robustness with respect to the problem parameters for time-dependent problems, there are more preconditioners that can be recommended in this case, including LSC, PCD, SIMPLER and MSIMPLER, especially for the low-degree discretization 2-1, C^0 and higher-degree discretizations of maximum continuity. For the inflow/outflow problems, PCD is the only one of them which is also robust with respect to the continuity (for fine enough meshes in the case of TB-2D) and outperforms the other preconditioners. It is also the only one which is robust with respect to the aspect ratio for all discretizations.

6.6.4 3D test problems

In this section, we present results for the 3D test problems and very briefly comment on both steady-state and time-dependent problems together. We do not divide the section into parts devoted to individual aspects, since there is nothing fundamentally different from what was stated above for the 2D test problems.

A comparison of iteration counts for the steady-state problems with uniform meshes is in Table 6.17 and Table 6.18 compares various viscosity values. The corresponding results for the time-dependent problems are summarized in Tables 6.19 and 6.20. Some entries of the tables are missing, either because the corresponding parameter or discretization were not considered or the computation failed due to lack of computer memory. Again, we refer to Appendix B for results including all considered discretizations.

The convergence of LSC and SIMPLE-type preconditioners may seem independent of the uniform mesh refinement for the steady-state problems in some cases, but that is probably because the meshes considered for the 3D problems are too coarse and the mesh dependence is not manifested yet.

The computational times for the steady-state and time-dependent BFS-3D problem are shown in Figure 6.31 and 6.32, respectively. We have chosen the mesh M2 for this comparison, since the results for BFS-3D on the finest mesh M3 are missing for the discretization 4-3, C^0. The computational time comparison for LDC-3D on M3 is presented in Appendix B, Figures B.10 and B.11.

	LDC-3D			BFS-3D			TB-3D
LSC	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	10	22	12	22	28	23	23
M2	13	23	15	27	22	26	25
M3	16	23	19	26	—	27	35
PCD	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	43	62	52	31	28	28	23
M2	49	34	58	35	25	29	22
M3	56	33	31	30	—	24	20
SIMPLE	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	18	190	53	51	>300	229	174
M2	29	235	55	93	>300	185	175
M3	49	227	55	133	—	137	218
SIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	8	20	11	23	28	27	24
M2	11	20	12	28	21	24	28
M3	15	29	12	28	—	22	34
MSIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	9	23	12	22	29	24	28
M2	11	21	13	26	22	25	31
M3	14	21	17	24	—	25	40
AL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	9	45	31	8	29	24	44
M2	9	30	16	8	—	18	—
MAL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	56	>300	>300	36	181	102	253
M2	68	>300	230	40	—	104	—

Table 6.17: Comparison of block preconditioners for the steady-state three-dimensional problems with uniform meshes. LDC-3D and BFS-3D with $\nu = 0.01$, TB-3D with $\nu = 0.1$. (Results for all discretizations of LDC-3D and BFS-3D in Appendix B, Tables B.17 and B.21, respectively.)

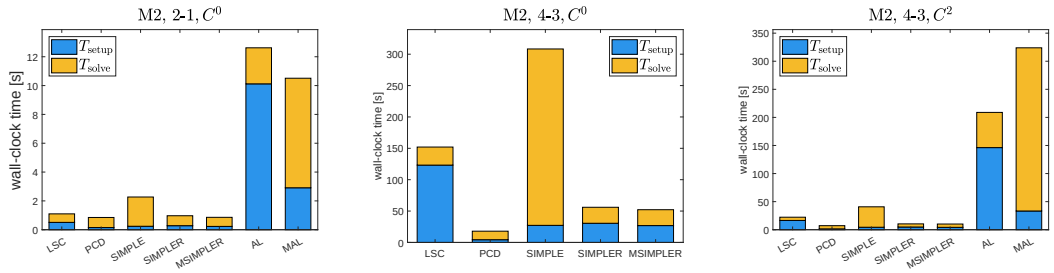


Figure 6.31: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the steady-state BFS-3D problem with $\nu = 0.01$ on the mesh M2.

	LDC-3D			BFS-3D			TB-3D
LSC	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	9	14	10	12	17	14	35
$\nu = 0.05$	9	15	11	12	18	14	36
$\nu = 0.01$	13	23	15	27	22	26	—
$\nu = 0.005$	17	28	18	—	—	—	—
PCD	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	36	21	19	17	17	17	20
$\nu = 0.05$	40	24	22	18	19	17	22
$\nu = 0.01$	49	34	58	35	25	29	—
$\nu = 0.005$	59	40	67	—	—	—	—
SIMPLE	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	18	135	29	28	131	43	218
$\nu = 0.05$	19	142	32	34	179	60	>300
$\nu = 0.01$	29	235	55	93	>300	185	—
$\nu = 0.005$	40	>300	84	—	—	—	—
SIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	8	18	8	13	22	12	34
$\nu = 0.05$	9	18	8	13	20	12	38
$\nu = 0.01$	11	20	12	28	21	24	—
$\nu = 0.005$	15	22	16	—	—	—	—
MSIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	8	16	10	12	17	14	40
$\nu = 0.05$	8	16	10	11	17	13	43
$\nu = 0.01$	11	21	13	26	22	25	—
$\nu = 0.005$	15	26	16	—	—	—	—
AL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	17	79	36	8	—	22	—
$\nu = 0.05$	8	60	28	7	—	17	—
$\nu = 0.01$	9	30	16	8	—	18	—
$\nu = 0.005$	8	23	14	—	—		—
MAL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	75	>300	141	29	—	92	—
$\nu = 0.05$	76	>300	156	29	—	89	—
$\nu = 0.01$	68	>300	230	40	—	104	—
$\nu = 0.005$	64	>300	>300	—	—		—

Table 6.18: Comparison of block preconditioners for the steady-state three-dimensional problems with various viscosity values on the mesh M2 for LDC-3D and BFS-3D, mesh M3 for TB-3D. (Results for all discretizations of LDC-3D and BFS-3D in Appendix B, Tables B.18 and B.22, respectively.)

	LDC-3D			BFS-3D			TB-3D
LSC	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	5	13	8	6	13	10	18
M2	5	12	7	5	11	8	21
M3	5	11	6	5	—	6	29
PCD	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	9	10	8	9	11	14	17
M2	9	12	9	9	10	11	17
M3	10	13	10	8	—	11	17
SIMPLE	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	16	220	50	9	175	45	137
M2	17	237	48	9	205	36	152
M3	14	184	40	8	—	30	197
SIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	6	18	9	7	17	12	20
M2	5	16	9	7	15	9	24
M3	4	15	7	6	—	8	30
MSIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	6	16	9	7	18	11	24
M2	6	15	8	6	15	9	27
M3	5	13	7	6	—	7	35
AL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	7	11	9	14	18	18	11
M2	6	8	7	16	—	17	—
MAL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
M1	13	58	32	16	21	20	179
M2	17	64	32	18	—	19	—

Table 6.19: Comparison of block preconditioners for the time-dependent three-dimensional problems with uniform meshes and $\Delta t = 0.01$. LDC-3D and BFS-3D with $\nu = 0.01$, TB-3D with $\nu = 0.1$. (Results for all discretizations of LDC-3D and BFS-3D in Appendix B, Tables B.19 and B.23, respectively.)

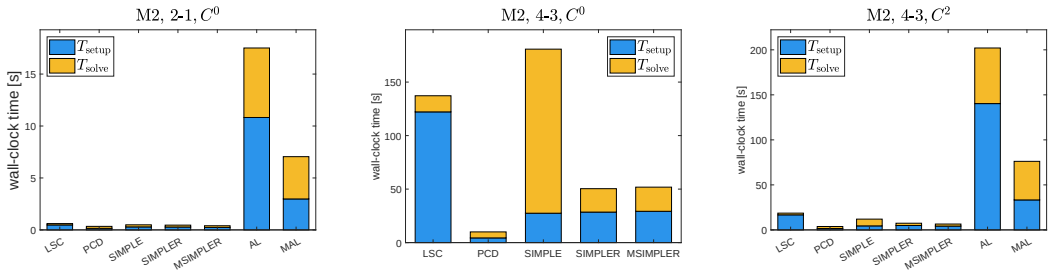


Figure 6.32: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the time-dependent BFS-3D problem with $\nu = 0.01$ and $\Delta t = 0.01$ on the mesh M2.

	LDC-3D			BFS-3D			TB-3D
LSC	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	4	11	6	5	10	6	29
$\nu = 0.05$	4	11	6	5	10	6	30
$\nu = 0.01$	5	12	7	5	11	8	—
$\nu = 0.005$	5	12	8	—	—	—	—
PCD	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	10	13	11	8	8	9	17
$\nu = 0.05$	10	13	10	8	9	10	17
$\nu = 0.01$	9	12	9	9	10	11	—
$\nu = 0.005$	9	11	8	—	—	—	—
SIMPLE	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	11	145	37	9	125	32	197
$\nu = 0.05$	14	160	41	9	148	34	279
$\nu = 0.01$	17	237	48	9	205	36	—
$\nu = 0.005$	17	273	50	—	—	—	—
SIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	4	16	7	6	13	7	30
$\nu = 0.05$	4	16	8	6	13	8	32
$\nu = 0.01$	5	16	9	7	15	9	—
$\nu = 0.005$	5	17	9	—	—	—	—
MSIMPLER	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	5	14	7	5	13	7	35
$\nu = 0.05$	5	14	7	5	13	8	37
$\nu = 0.01$	6	15	8	6	15	9	—
$\nu = 0.005$	6	15	8	—	—	—	—
AL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	7	22	11	16	—	17	—
$\nu = 0.05$	7	16	9	16	—	17	—
$\nu = 0.01$	6	8	7	16	—	17	—
$\nu = 0.005$	6	7	7	—	—	—	—
MAL	2-1, C^0	4-3, C^0	4-3, C^2	2-1, C^0	4-3, C^0	4-3, C^2	4-3, C^2
$\nu = 0.1$	21	147	56	18	—	19	—
$\nu = 0.05$	20	117	48	18	—	19	—
$\nu = 0.01$	17	64	32	18	—	19	—
$\nu = 0.005$	20	50	29	—	—	—	—

Table 6.20: Comparison of block preconditioners for the time-dependent three-dimensional problems with $\Delta t = 0.01$ and various viscosity values on the mesh M2 for LDC-3D and BFS-3D, mesh M3 for TB-3D. (Results for all discretizations of LDC-3D and BFS-3D in Appendix B, Tables B.20 and B.24, respectively.)

Chapter 7

Conclusions

This thesis focused on an important part of incompressible fluid flow simulation – an efficient solution of the saddle-point linear systems that arise from discretization of the incompressible Navier–Stokes equations. We are interested in a specific discretization approach, isogeometric analysis. As mentioned in the introduction of this work, it is believed that IgA can provide solutions of comparable quality to the widely used discretization methods (FEM, FVM), but with significantly less degrees of freedom and thus, hopefully, with less effort.

Solution of the linear systems represents one of the main bottlenecks of the numerical flow simulation. We consider the state-of-the-art techniques based on iterative solution with preconditioned Krylov subspace methods, using specialized block preconditioners. These methods are usually developed for and applied to linear systems resulting from finite element or finite volume discretizations, but, to the best of our knowledge, they have not yet been tested for isogeometric discretizations of the Navier–Stokes equations.

We presented results of extensive numerical experiments comparing the performance of selected preconditioners for several test problems. Namely, the following preconditioners were involved in the comparison: LSC, PCD, AL, modified AL (MAL) and several SIMPLE-type preconditioners (SIMPLE, SIMPLER, MSIMPLER). Two well known benchmark problems were chosen as test problems: the lid-driven cavity flow (as a representative of enclosed flow) and the backward facing step problem, both in two and three dimensions. Additionally, a problem originating from industrial practice was considered, also in 2D and 3D.

The first set of experiments was devoted to the question of mass matrix approximation. The mass matrix (either for velocity or pressure) appears in most of the preconditioners and it is usually approximated by its main diagonal. However, our experiments show that this choice is not ideal for some IgA discretizations. For LSC and MSIMPLER, it leads to a relatively strong mesh dependence of the convergence for discretizations of low continuity. Thanks to the properties of B-spline basis functions, the isogeometric mass matrices are suitable for row-sum lumping, which yields a different diagonal approximation of the mass matrix. It turns out that using the lumped mass matrix instead of the diagonal approximation leads to improved behavior of LSC and MSIMPLER for the low-continuity discretizations, including the low-degree discretization 2-1, C^0 with linear basis for pressure and quadratic basis for velocity. The convergence for discretizations of maximum continuity remains almost unchanged.

Another topic we addressed in the numerical section was the choice of boundary conditions for the discrete operators that have to be defined in order to construct the

PCD preconditioner. We considered several variants known from the literature and, inspired by some of them, we proposed additional two. In some of the variants (including the newly proposed ones), the mass matrix approximation also plays a role. They are not universally applicable to IgA discretizations, because their convergence gets worse with increasing discretization degree and more so for low-continuity bases. Lumping does not help in this case, however, it seems that some "better" mass matrix approximation might result in discretization-independent convergence. Without it, these variants are useful only for the low-degree discretization 2-1, C^0. An advantage of one of the proposed variants is that it does not require the knowledge of individual parts of the boundary (inflow, outflow, solid wall). It does not work for steady-state problems, but it performs well for time-dependent problems with the 2-1, C^0 discretization and also higher-degree discretizations of maximum continuity.

Based on our experiments, we have identified suitable choices for the treatment of PCD boundary conditions and described the construction of the preconditioner in detail. It is a combination of ideas from the literature and it leads to convergence robust with respect to the mesh refinement as well as the discretization degree and continuity.

We used the mentioned modifications of LSC, MSIMPLER and PCD in the main comparison of all preconditioners. Our observations suggest that LSC, PCD, MSIMPLER and the AL-based preconditioners behave as expected for all IgA discretizations. SIMPLE and SIMPLER seem to be not suitable for higher-degree low-continuity discretizations. Overall, the considered variant of the PCD preconditioner appears as the best choice due to its robustness with respect to uniform mesh refinement, mesh aspect ratio and discretization and also due to its computational costs.

It is important to emphasize that we investigated only the ideal versions of the preconditioners in this work. It means that all subsystems were solved with a direct solver, which is, of course, not practical. For an efficient preconditioner, it is necessary to employ suitable approximate solvers for the subsystems. The quality of these inner solvers can have a major impact on the performance of the block preconditioners. The subsystems are typically of Poisson and convection–diffusion type. Thus, investigation of efficient solvers for such problems should be the next research direction. Usually, multigrid is the method of choice. However, development of multigrid methods suitable for IgA is still an evolving field.

Another challenging task for future research is the approximate solution of the augmented $(1, 1)$ block of the AL preconditioner. If we were able to solve this subsystem efficiently, AL would be a very good preconditioner. To our knowledge, this is not completely resolved even for general standard finite element discretizations.

Chapter 8

Bibliography

[1] Y. Bazilevs, L. B. da Veiga, J. Cottrell, T. Hughes, and G. Sangalli, *Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes*, Mathematical Models and Methods in Applied Sciences, 16 (2006), pp. 1031–1090.

[2] Y. Bazilevs and T. Hughes, *Weak imposition of Dirichlet boundary conditions in fluid mechanics*, Computers & Fluids, 36 (2007), pp. 12–26.

[3] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[4] ——, *Some uses of the field of values in numerical analysis*, Bollettino dell'Unione Matematica Italiana, 14 (2021), pp. 159–177.

[5] M. Benzi, G. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.

[6] M. Benzi and M. A. Olshanskii, *An augmented Lagrangian-based approach to the Oseen problem*, SIAM J. Sci. Comput., 28 (2006), pp. 2095–2113.

[7] ——, *Field-of-values convergence analysis of augmented lagrangian preconditioners for the linearized Navier–Stokes problem*, SIAM J. Numer. Anal., 49 (2011), p. 770–788.

[8] M. Benzi, M. A. Olshanskii, and Z. Wang, *Modified augmented Lagrangian preconditioners for the incompressible Navier–Stokes equations*, Int. J. Numer. Meth. Fluids, 66 (2011), pp. 486–508.

[9] J. Blechta, *Towards efficient numerical computation of flows of non-Newtonian fluids*, PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2019.

[10] M. Bollhöfer, O. Schenk, R. Janalik, S. Hamm, and K. Gullapalli, *State-of-the-art sparse direct solvers*, in Parallel Algorithms in Computational Science and Engineering, A. Grama and A. Sameh, eds., Modeling and Simulation in Science, Engineering and Technology, Birkhäuser, 2020, pp. 3–33.

[11] M. Braack and P. B. Mucha, *Directional do-nothing condition for the Navier–Stokes equations*, Journal of Computational Mathematics, 32 (2014), pp. 507–521.

[12] A. Bressan and G. Sangalli, *Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique*, IMA Journal of Numerical Analysis, 33 (2012), pp. 629–651.

[13] A. Buffa, C. de Falco, and G. Sangalli, *Isogeometric analysis: Stable elements for the 2D stokes equation*, Int. J. Num. Meth. Fluids, 65 (2011), pp. 1407 – 1422.

[14] A. Buffa, H. Harbrecht, A. Kunoth, and G. Sangalli, *BPX-preconditioning for isogeometric analysis*, Comput. Methods Appl. Mech. Engrg., 265 (2013), pp. 63–70.

[15] F. Calabrò, G. Sangalli, and M. Tani, *Fast formation of isogeometric Galerkin matrices by weighted quadrature*, Comput. Methods Appl. Mech. Engrg., 316 (2017), pp. 606–622.

[16] L. Cesari, *Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive*, Atti Accad. Nazionale Lincei R. Classe Sci. Fis. Mat. Nat., 25 (1937), pp. 422–428.

[17] K. Chen, *Matrix preconditioning techniques and applications*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2005.

[18] B. Cockburn, *Discontinuous Galerkin methods for convection-dominated problems*, Lecture Notes in Computer Science and Engineering, 9 (1999).

[19] ——, *Discontinuous Galerkin methods*, Journal of Applied Mathematics and Mechanics, 83 (2003).

[20] N. Collier, L. Dalcin, D. Pardo, and V. M. Calo, *The cost of continuity: Perfomance of iterative solvers on isogeometric finite elements*, SIAM J. Sci. Comput., 35 (2013), pp. 767–784.

[21] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, and V. M. Calo, *The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers*, Comput. Methods Appl. Mech. Engrg., 213-216 (2012), pp. 353–361.

[22] A. M. A. Côrtes, L. Dalcin, A. Sarmiento, N. Collier, and V. M. Calo, *A scalable block-preconditioning strategy for divergence-conforming B-spline discretizations of the Stokes problem*, Comput. Methods Appl. Mech. Engrg., 316 (2017), pp. 839–858.

[23] J. Cottrell, T. Hughes, and Y. Bazilevs, *Isogeometric analysis: Toward integration of CAD and FEA*, John Wiley & Sons, Ltd, (2009).

[24] J. Cottrell, R. Reali, Y. Bazilevs, and T. Hughes, *Isogeometric analysis of structural vibrations*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 5257–5296.

[25] C. de Boor, *A practical guide to splines*, vol. 27 of Applied Mathematical Sciences, Springer, New York, 2001. Revised Edition.

[26] A. P. de la Riva, C. Rodrigo, and F. J. Gaspar, *A robust multigrid solver for isogeometric analysis based on multiplicative Schwarz smoothers*, SIAM J. Sci. Comput., 41 (2019), pp. S321–S345.

[27] T. Dokken, T. Lyche, and K. F. Pettersen, *Polynomial splines over locally refined box-partitions*, Computer Aided Geometric Design, 30 (2013), pp. 331–356.

[28] M. Donatelli, C. Garoni, C. Manni, S. Serra-Capizzano, and H. Speleers, *Symbol-based multigrid methods for Galerkin B-spline isogeometric analysis*, SIAM J. Numer. Anal., 55 (2017), pp. 31–62.

[29] J. P. V. Doormaal and G. D. Raithby, *Enhancements of the SIMPLE method for predicting incompressible fluid flows*, Numerical Heat Transfer, 7 (1984), pp. 147–163.

[30] S. Duczek and H. Gravenkamp, *Mass lumping techniques in the spectral element method: On the equivalence of the row-sum, nodal quadrature, and diagonal scaling methods*, Comput. Methods Appl. Mech. Engrg., 353 (2019), p. 516–569.

[31] J. Egermaier and H. Horníková, *On the parameter in augmented Lagrangian preconditioning for isogeometric discretizations of the Navier–Stokes equations*, Applications of mathematics, (2022). https://doi.org/10.21136/AM.2022.0130-21.

[32] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), p. 345–357.

[33] H. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[34] H. Elman and G. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1645–1661.

[35] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *Block preconditioners based on approximate commutators*, SIAM J. Sci. Comput., 27 (2006), pp. 1651–1668.

[36] ——, *A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations*, Journal of Computational Physics, 227 (2008), p. 1790–1808.

[37] H. Elman, V. E. Howle, J. Shadid, D. Silvester, and R. Tuminaro, *Least squares preconditioners for stabilized discretizations of the Navier–Stokes equations*, SIAM J. Sci. Comput., 30 (2007), pp. 290–311.

[38] H. Elman and D. Silvester, *Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46.

[39] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2005.

[40] ——, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2nd ed., 2014.

[41] H. Elman and R. Tuminaro, *Boundary conditions in approximate commutator preconditioners for the Navier–Stokes equations*, Electronic transactions on numerical analysis, 35 (2009), pp. 257–280.

[42] D. J. Evans, *The use of pre-conditioning in iterative methods for solving linear systems with symmetric positive definite matrices*, IMA Journal of Applied Mathematics, 4 (1968), p. 295–314.

[43] ——, *The use of preconditioning in direct methods of solving positive definite linear systems*, Int. J. Computer Math., 69 (1998), pp. 101–121.

[44] P. E. Farrell, L. Mitchell, and F. Wechsung, *An augmented Lagrangian preconditioner for the 3D stationary incompressible Navier–Stokes equations at high Reynolds number*, SIAM J. Sci. Comput., 41 (2019), pp. A3073–A3096.

[45] M. Feistauer, *Mathematical methods in fluid dynamics*, Longman Scientific & Technical, Harlow, 1993.

[46] B. Fischer, A. Ramage, D. Silvester, and A. Wathen, *Minimum residual methods for augmented systems*, BIT, 38 (1998), pp. 527–543.

[47] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, G. A. Watson, ed., vol. 506 of Lecture Notes in Mathematics, Berlin, 1976, Springer, pp. 73–89.

[48] M. Fortin and R. Glowinski, *Augmented Lagrangian methods: Applications to the numerical solution of boundary-value problems*, vol. 15 of Stud. Math. Appl., North-Holland, 1983.

[49] T.-P. Fries and H. G. Matthies, *A review of Petrov–Galerkin stabilization approaches and extension to meshfree methods*, Technical report: Informatikbericht 2004-01, Technical University Braunschweig, (2004).

[50] G. P. Galdi, *An introduction to the mathematical theory of the Navier–Stokes equations. Steady-state problems*, Springer Monographs in Mathematics, Springer, 2011. Second Edition.

[51] D. Garcia, D. Pardo, L. Dalcin, M. Paszynski, N. Collier, and V. M. Calo, *The value of continuity: Refined isogeometric analysis and fast direct solvers*, Comput. Methods Appl. Mech. Engrg., 316 (2017), pp. 586–605.

[52] C. Gianelli, B. Jüttler, and H. Speleers, *THB-splines: The truncated basis for hierarchical splines*, Computer Aided Geometric Design, 29 (2012), pp. 485–498.

[53] G. Golub and C. Greif, *On solving block-structured indefinite linear systems*, SIAM J. Sci. Comput., 24 (2003), p. 2076–2092.

[54] G. H. Golub and C. F. V. Loan, *Matrix computations*, The Johns Hopkins University Press, Baltimore, 4th ed., 2013.

[55] G. Guennebaud, B. Jacob, et al., *Eigen v3*. http://eigen.tuxfamily.org, 2010.

[56] X. He and C. Vuik, *Comparison of some preconditioners for the incompressible Navier–Stokes equations*, Numer. Math. Theor. Meth. Appl., 9 (2016), pp. 239–261.

[57] X. He, C. Vuik, and C. Klaij, *Block-preconditioners for the incompressible Navier–Stokes equations discretized by a finite volume method*, J. Numer. Math., 25 (2017), pp. 89–105.

[58] M. R. Hestenes, *The conjugate-gradient method for solving linear systems*, in Proceedings of the Sixth Symposium in Applied Mathematics 1953, New York, 1956, McGraw-Hill, pp. 83–102.

[59] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[60] C. Hofer and S. Takacs, *A parallel multigrid solver for multi-patch isogeometric analysis*, Advanced Finite Element Methods with Applications. FEM 2017. Lecture Notes in Computational Science and Engineering, 128 (2019), pp. 205 – 219.

[61] C. Hofreither and S. Takacs, *Robust multigrid for isogeometric analysis based on stable splittings of spline spaces*, SIAM J. Numer. Anal., 55 (2017), pp. 2004–2024.

[62] H. Horníková, C. Vuik, and J. Egermaier, *A comparison of block preconditioners for isogeometric analysis discretizations of the incompressible Navier–Stokes equations*, Int. J. Numer. Meth. Fluids, 93 (2021), pp. 1788–1815.

[63] T. Hughes, J. Cottrel, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 4135–4195.

[64] R. I. Issa, *Solution of the implicitly discretised fluid flow equations by operator-splitting*, J. Comp. Phys., 62 (1986), pp. 40–65.

[65] C. G. J. Jacobi, *Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden linearen Gleichungen*, Astronomische Nachrichten, 22 (1845), pp. 297–306.

[66] V. John, *Finite element methods for incompressible flow problems*, vol. 51 of Springer Series in Computational Mathematics, Springer, 2016.

[67] D. Kay and D. Loghin, *A Green's function preconditioner for the steady-state Navier–Stokes equations*, Oxford University Computing Laboratory, (1999). Report 99/06.

[68] D. Kay, D. Loghin, and A. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.

[69] C. M. Klaij and C. Vuik, *SIMPLE-type preconditioners for cell-centered, colocated finite volume discretization of incompressible Reynolds-averaged Navier–Stokes equations*, Int. J. Numer. Meth. Fluids, 17 (2013), pp. 830–849.

[70] U. Langer, A. Mantzaflaris, S. E. Moore, and I. Toulopoulos, *Multipatch discontinuous Galerkin isogeometric analysis*, Lecture Notes in Computational Science and Engineering, 107 (2014), pp. 1–32.

[71] J. Liesen and Z. S. s, *Krylov subspace methods: principles and analysis*, Numerical Methods and Scientific Computation, Oxford University Press, Oxford, 2013.

[72] D. Loghin, *Analysis of preconditioned Picard iterations for the Navier–Stokes equations*, (2001).

[73] D. Loghin and A. Wathen, *Analysis of preconditioners for saddle-point problems*, SIAM J. Sci. Comput., 25 (2004), pp. 2029–2049.

[74] A. Mantzaflaris, B. Jüttler, B. Khoromskij, and U. Langer, *Low rank tensor methods in Galerkin-based isogeometric analysis*, Comput. Methods Appl. Mech. Engrg., 316 (2017), pp. 1062–1085.

[75] A. Mantzaflaris and others (see website), *G+Smo (Geometry plus Simulation modules) v0.8.1*. http://github.com/gismo, 2018.

[76] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[77] J. Nédélec, *A new family of mixed finite elements in* $\mathbb{R}^3$, Numerische Mathematik, 50 (1986), pp. 57–81.

[78] P. N. Nielsen, A. R. Gersborg, J. Gravesen, and N. L. Pedersen, *Discretizations in isogeometric analysis of Navier–Stokes flow*, Computer Methods in Applied Mechanics and Engineering, 200 (2011), pp. 3242–3253.

[79] M. A. Olshanskii and Y. V. Vassilevski, *Pressure Schur complement preconditioners for the discrete Oseen problem*, SIAM J. Sci. Comput., 29 (2007), pp. 2686–2704.

[80] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[81] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York, 1980.

[82] S. V. Patankar and D. B. Spalding, *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, Int. J. Heat and Mass Transfer, 15 (1972), pp. 1787–1805.

[83] J. W. Pearson and J. Pestana, *Preconditioners for Krylov subspace methods: An overview*, GAMM-Mitteilungen, 43 (2020).

[84] L. Piegl and W. Tiller, *The NURBS book*, Springer, 1997.

[85] P. Raviart and J. Thomas, *A mixed finite element method for 2nd order elliptic problems*, in Mathematical Aspects of the Finite Element Mehtod, I. Galligani and E. Magenes, eds., vol. 606 of Lecture Notes in Mathematics, Springer, 1977, pp. 292–315.

[86] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[87] ——, *Iterative methods for sparse linear systems, second edition*, SIAM, (2003).

[88] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[89] Y. Saad and H. A. van der Vorst, *Iterative solution of linear systems in the 20th century*, J. Comput. Appl. Math., 123 (2000), pp. 1–33.

[90] G. Sangalli and M. Tani, *Isogeometric preconditioners based on fast solvers for the Sylvester equation*, SIAM J. Sci. Comput., 38 (2016), pp. A3644–A3671.

[91] O. Schenk and K. Gärtner, *Solving unsymmetric sparse systems of linear equations with PARDISO*, J. of Future Generation Computer Systems, 20 (2004), pp. 475–487.

[92] T. W. Sederberg, J. Zheng, A. Bakenov, and A. H. Nasri, *T-splines and T-NURCCs*, ACM Transactions on Graphics, 22 (2003), pp. 477–484.

[93] A. Segal, *Finite element methods for the incompressible Navier–Stokes equations*, J.M. Burgerscentrum, Delft University of Technology, (2017).

[94] A. Segal, M. ur Rehman, and C. Vuik, *Preconditioners for incompressible Navier–Stokes solvers*, Numer. Math. Theor. Meth. Appl., 3 (2010), pp. 245–275.

[95] P. Shankar and M. Deshpande, *Fluid mechanics in the driven cavity*, Annu. Rev. Fluid Mech., 32 (2000), pp. 93–136.

[96] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow*, Journal of Computational and Applied Mathematics, 128 (2001), pp. 261–279.

[97] V. Simoncini and D. B. Szyld, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.

[98] C. Taylor and P. Hood, *A numerical solution of the Navier–Stokes equations using the finite element technique*, Int. J. Comput. & Fluids, 1 (1973), pp. 73–100.

[99] R. Tielen, M. Möller, D. Göddeke, and C. Vuik, *A p-multigrid method enhanced with an ILUT smoother and its comparison to h-multigrid methods within isogeometric analysis. Accepted for publication in Computer Methods in Applied Mechanics and Engineering*, 2019.

[100] A. Toselli and O. B. Widlund, *Domain decomposition methods - algorithms and theory*, vol. 34 of Springer Series in Computational Mathematics, Springer, 2004.

[101] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, 2000.

[102] A. M. Turing, *Rounding-off errors in matrix processes*, Q. J. Mech. Appl. Math., 1 (1948), pp. 287–308.

[103] E. Turnerová, *Isogeometric analysis for incompressible turbulent flow*, Ph.D. thesis, University of West Bohemia, Pilsen, Czech republic, (2019).

[104] M. ur Rehman, *Fast iterative methods for incompressible Navier–Stokes equations*, Ph.D. thesis, Delft University of Technology, the Netherlands, (2010).

[105] M. ur Rehman, C. Vuik, and A. Segal, *A comparison of preconditioners for incompressible Navier–Stokes solvers*, Int. J. Numer. Meth. Fluids, 57 (2008), pp. 1731–1751.

[106] ——, *Preconditioners for the steady incompressible Navier–Stokes problem*, IAENG International Journal of Applied Mathematics, 38 (2008).

[107] ——, *SIMPLE-type preconditioners for the Oseen problem*, Int. J. Numer. Meth. Fluids, 61 (2008), pp. 432–452.

[108] H. Uzawa, *Iterative methods for concave programming*, in Studies in linear and nonlinear programming, K. J. Arrow, L. Hurwicz, and H. Uzawa, eds., Stanford University Press, Stanford, CA, 1958.

[109] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*, SIAM J. Sci. Stat. Comput., 12 (1992), p. 631–644.

[110] H. A. van der Vorst and C. Vuik, *GMRESR: a family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[111] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: The finite volume method*, Pearson Education Ltd, 2007.

[112] C. Vuik and A. Saghir, *The Krylov accelerated SIMPLE(R) method for incompressible flow*, Delft University of Technology, Technical Report 02-01, (2002).

[113] C. Vuik, A. Saghir, and G. P. Boerstoel, *The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces*, Int. J. Numer. Meth. Fluids, 33 (2000), pp. 1027–1040.

[114] A. J. Wathen, *Preconditioning*, Acta Numerica, 24 (2015), pp. 329–376.

[115] P. Wesseling, *Principles of computational fluid dynamics*, vol. 29 of Springer Series in Computational Mathematics, Springer, 2001.

[116] D. M. Young and K. C. Jea, *Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), p. 159–194.

List of publications

Articles in journals

- H. Horníková, C. Vuik, J. Egermaier J. *A comparison of block preconditioners for isogeometric analysis discretizations of the incompressible Navier–Stokes equations.* International Journal for Numerical Methods in Fluids, 93 (2021), pp. 1788–1815.

- B. Bastl, M. Brandner, J. Egermaier, H. Horníková, K. Michálková, E. Turnerová. *Numerical simulation of lid-driven cavity flow by isogeometric analysis.* Acta polytechnica, 61 (2021), pp. 33–48.

Conference papers

- B. Bastl, M. Brandner, J. Egermaier, H. Horníková, K. Michálková, E. Turnerová. *Gradient-free and gradient-based methods for shape optimization of water turbine blade.* In: J. Chleboun, P. Kůs, P. Přikryl, M. Rozložník, K. Segeth, J. Šístek, and T. Vejchodský (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Hejnice, June 24-29, 2018. Institute of Mathematics CAS, Prague, 2019. pp. 15–26.

- B. Bastl, M. Brandner, J. Egermaier, H. Horníková, K. Michálková, J. Šourek, E. Turnerová. *Comparison of coupled and decoupled solvers for incompressible Navier–Stokes equations solved by isogeometric analysis.* In: H. van Brummelen, A. Corsini, A. Perotto, G. Rozza (eds.) Numerical Methods for Flows, volume 132 of Lecture Notes in Computational Science and Engineering. Springer, Cham, 2020.

- H. Horníková, C. Vuik. *Preconditioning for linear systems arising from IgA discretized incompressible Navier–Stokes equations.* In: H. van Brummelen, C. Vuik, M. Möller, C. Verhoosel, B. Simeon, B. Jüttler (eds.): Isogeometric Analysis and Applications 2018, volume 133 of Lecture Notes in Computational Science and Engineering. Springer, Cham, 2021.

- J. Egermaier, H. Horníková. *On the parameter in augmented Lagrangian preconditioning for isogeometric discretizations of the Navier–Stokes equations.* Applications of Mathematics (2022), available online: https://link.springer.com/article/10.21136/AM.2022.0130-21.

Other publications

- B. Bastl, M. Brandner, H. Horníková, J. Šourek. *Report and prototype software for incompressible flow solver*, Research report, Horizon 2020, No. 678727 (MOTOR), 2017.

- B. Bastl, M. Brandner, H. Horníková, E. Turnerová, *Report: Incompressible turbulent flow solver*, Research report, Horizon 2020, No. 678727 (MOTOR), 2018.

Appendix A

Test problems: DOFs and nonzeros

M1	DOFs	nnz	nnz [%]
2-1, C^0	2 210	52 022	1.07
3-2, C^0	5 506	219 986	0.73
3-2, C^1	2 371	129 716	2.31
4-3, C^0	10 338	622 150	0.58
4-3, C^1	5 763	461 320	1.39
4-3, C^2	2 538	247 270	3.84
5-4, C^0	16 706	1 409 042	0.50
5-4, C^1	10 691	1 160 372	1.02
5-4, C^2	6 026	799 922	2.20
5-4, C^3	2 711	408 692	5.56

(a) Mesh with 16×16 elements (M1).

M2	DOFs	nnz	nnz [%]
2-1, C^0	9 026	220 598	0.27
3-2, C^0	22 274	917 778	0.18
3-2, C^1	9 347	537 780	0.62
4-3, C^0	41 666	2 572 742	0.15
4-3, C^1	22 787	1 898 120	0.37
4-3, C^2	9 674	1 004 390	1.07
5-4, C^0	67 202	5 793 746	0.13
5-4, C^1	42 371	4 753 076	0.26
5-4, C^2	23 306	3 249 266	0.60
5-4, C^3	10 007	1 628 276	1.63

(b) Mesh with 32×32 elements (M2).

M3	DOFs	nnz	nnz [%]
2-1, C^0	36 482	907 958	0.068
3-2, C^0	89 602	3 747 986	0.057
3-2, C^1	37 123	2 189 492	0.159
4-3, C^0	167 298	10 461 382	0.037
4-3, C^1	90 627	7 699 336	0.094
4-3, C^2	37 770	4 048 486	0.284
5-4, C^0	269 570	23 493 458	0.032
5-4, C^1	168 707	19 237 556	0.068
5-4, C^2	91 658	13 096 946	0.156
5-4, C^3	38 423	6 500 468	0.440

(c) Mesh with 64×64 elements (M3).

M4	DOFs	nnz	nnz [%]
2-1, C^0	146 690	3 683 510	0.017
3-2, C^0	359 426	15 146 898	0.012
3-2, C^1	147 971	8 835 252	0.040
4-3, C^0	670 466	42 188 486	0.009
4-3, C^1	361 475	31 012 232	0.024
4-3, C^2	149 258	16 256 102	0.073
5-4, C^0	1 079 810	94 614 098	0.008
5-4, C^1	673 283	77 402 804	0.017
5-4, C^2	363 530	52 588 274	0.040
5-4, C^3	150 551	25 976 948	0.115

(d) Mesh with 128×128 elements (M4).

Table A.1: LDC-2D number of degrees of freedom (DOFs), number of nonzero elements (nnz) in the sparse matrix and their percentage for all uniform meshes.

M1	DOFs	nnz	nnz [%]
2-1, C^0	2 349	54 100	0.98
3-2, C^0	5 857	230 176	0.67
3-2, C^1	2 758	142 750	1.88
4-3, C^0	10 997	653 108	0.54
4-3, C^1	6 506	497 858	1.18
4-3, C^2	3 185	285 908	2.82
5-4, C^0	17 769	1 482 256	0.47
5-4, C^1	11 886	1 242 478	0.88
5-4, C^2	7 173	889 648	1.73
5-4, C^3	3 630	493 966	3.75

(a) Uniform mesh with 272 elements (M1).

M2	DOFs	nnz	nnz [%]
2-1, C^0	9 593	232 132	0.25
3-2, C^0	23 681	968 216	0.17
3-2, C^1	10 402	581 438	0.54
4-3, C^0	44 297	2 718 020	0.14
4-3, C^1	24 970	2 032 538	0.33
4-3, C^2	11 229	1 113 508	0.88
5-4, C^0	71 441	6 126 664	0.12
5-4, C^1	46 066	5 069 902	0.24
5-4, C^2	26 277	3 532 072	0.51
5-4, C^3	12 074	1 848 334	1.27

(b) Uniform mesh with 1088 elements (M2).

M3	DOFs	nnz	nnz [%]
2-1, C^0	38 769	960 292	0.064
3-2, C^0	95 233	3 968 584	0.044
3-2, C^1	40 378	2 346 622	0.144
4-3, C^0	177 809	11 084 516	0.035
4-3, C^1	97 802	8 212 490	0.086
4-3, C^2	42 005	4 394 180	0.249
5-4, C^0	286 497	24 903 928	0.030
5-4, C^1	181 338	20 480 014	0.062
5-4, C^2	100 389	14 075 224	0.140
5-4, C^3	43 650	7 142 158	0.375

(c) Uniform mesh with 4352 elements (M3).

M4	DOFs	nnz	nnz [%]
2-1, C^0	155 873	3 904 996	0.016
3-2, C^0	381 953	16 066 472	0.011
3-2, C^1	159 082	9 428 222	0.037
4-3, C^0	712 481	44 764 196	0.009
4-3, C^1	387 082	3 3014 762	0.022
4-3, C^2	162 309	17 457 412	0.066
5-4, C^0	1 147 457	100 412 248	0.008
5-4, C^1	719 530	82 321 294	0.016
5-4, C^2	392 229	56 194 744	0.037
5-4, C^3	165 554	28 070 158	0.102

(d) Uniform mesh with 17408 elements (M4).

Table A.2: BFS-2D number of degrees of freedom (DOFs), number of nonzero elements (nnz) in the sparse matrix and their percentage for all uniform meshes.

M1	DOFs	nnz	nnz [%]
2-1, C^0	1 153	65 931	4.96
3-2, C^0	4 721	667 221	2.99
3-2, C^1	1 751	326 454	10.65
4-3, C^0	12 321	3 449 787	2.27
4-3, C^1	6 183	2 324 616	6.08
4-3, C^2	2 529	1 045 491	16.35

(a) Mesh with $4 \times 4 \times 4$ elements (M1).

M2	DOFs	nnz	nnz [%]
2-1, C^0	10 853	802 995	0.68
3-2, C^0	41 413	7 105 689	0.41
3-2, C^1	13 287	3 337 974	1.89
4-3, C^0	104 997	34 369 491	0.31
4-3, C^1	47 303	22 396 152	1.00
4-3, C^2	16 069	9 284 907	3.59

(b) Mesh with $8 \times 8 \times 8$ elements (M2).

M3	DOFs	nnz	nnz [%]
2-1, C^0	94 285	7 794 627	0.09
3-2, C^0	347 405	65 092 353	0.05
3-2, C^1	104135	30 087 798	0.28
4-3, C^0	867 789	305 480 643	0.04
4-3, C^1	371 079	196 176 408	0.14
4-3, C^2	114 669	78 478 683	0.60

(c) Mesh with $16 \times 16 \times 16$ elements (M3).

Table A.3: LDC-3D number of degrees of freedom (DOFs), number of nonzero elements (nnz) in the sparse matrix and their percentage for all uniform meshes.

M1	DOFs	nnz	nnz [%]
2-1, C^0	3 180	231 960	2.29
3-2, C^0	11 968	2 043 672	1.43
3-2, C^1	5 081	1 150 287	4.46
4-3, C^0	30 132	9 849 432	1.08
4-3, C^1	16 365	7 078 539	2.64
4-3, C^2	7 554	3 666 870	6.43

(a) Mesh with 144 elements (M1).

M2	DOFs	nnz	nnz [%]
2-1, C^0	27 112	2 235 408	0.30
3-2, C^0	99 120	18 566 784	0.19
3-2, C^1	34 335	9 369 639	0.79
4-3, C^0	246 648	86 874 384	0.14
4-3, C^1	116 087	58 513 809	0.43
4-3, C^2	42 634	26 354 304	1.45

(b) Mesh with 1152 elements (M2).

M3	DOFs	nnz	nnz [%]
2-1, C^0	223 632	19 457 184	0.04
3-2, C^0	806 560	157 642 944	0.02
3-2, C^1	251 795	75 548 247	0.12
4-3, C^0	—	—	—
4-3, C^1	873 219	475 407 813	0.06
4-3, C^2	282 042	198 592 848	0.25

(c) Mesh with 9216 elements (M3).

Table A.4: BFS-3D number of degrees of freedom (DOFs), number of nonzero elements (nnz) in the sparse matrix and their percentage for all uniform meshes.

M1	DOFs	nnz	nnz [%]
2-1, C^0	2 677	65 386	0.92
4-3, C^0	12 277	758 682	0.50
4-3, C^2	3 313	322 794	2.94

(a) Uniform mesh with 300 elements (M1).

M2	DOFs	nnz	nnz [%]
2-1, C^0	10 757	267 706	0.23
4-3, C^0	49 157	3 075 482	0.12
4-3, C^2	11 993	1 243 274	0.86

(b) Uniform mesh with 1200 elements (M2).

M3	DOFs	nnz	nnz [%]
2-1, C^0	43 117	1 082 746	0.058
4-3, C^0	196 717	12 381 882	0.032
4-3, C^2	45 553	4 877 034	0.235

(c) Uniform mesh with 4800 elements (M3).

M4	DOFs	nnz	nnz [%]
2-1, C^0	17 2637	4 354 426	0.015
4-3, C^0	787 037	49 685 882	0.008
4-3, C^2	177 473	19 315 754	0.061

(d) Uniform mesh with 19200 elements (M4).

Table A.5: TB-2D number of degrees of freedom (DOFs), number of nonzero elements (nnz) in the sparse matrix and their percentage for all "uniform" meshes.

	DOFs	nnz	nnz [%]
M1	15 174	7 580 160	3.29
M2	98 827	62 688 306	0.64
M3	711 813	510 130 998	0.10

Table A.6: TB-3D number of degrees of freedom (DOFs), number of nonzero elements (nnz) in the sparse matrix and their percentage for all meshes, 4-3, C^2 discretization.

Appendix B

Complete results

B.1 Mass matrix approximation

LSC		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
$\mathrm{diag}(\mathbf{M}_u)$	M1	33	25	31	21	25	27	28	24	26	28
	M2	33	31	28	26	22	25	33	24	25	24
	M3	40	53	28	48	29	27	56	36	28	28
	M4	65	104	32	95	45	37	113	65	37	36
$\widehat{\mathbf{M}}_{u,L}$	M1	29	21	31	22	24	29	29	25	25	28
	M2	25	17	28	22	20	26	29	23	23	25
	M3	24	22	29	26	25	29	33	28	28	30
	M4	29	32	34	35	35	36	40	39	38	38
MSIMPLER		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
$\mathrm{diag}(\mathbf{M}_u)$	M1	31	26	31	20	28	27	26	24	28	29
	M2	31	31	28	24	23	25	32	22	24	25
	M3	39	52	27	47	28	26	56	35	27	27
	M4	64	103	31	96	44	36	114	65	37	36
$\widehat{\mathbf{M}}_{u,L}$	M1	28	22	30	21	27	28	29	25	28	28
	M2	24	18	27	21	21	26	28	22	23	25
	M3	23	20	27	24	24	28	31	27	27	28
	M4	27	30	32	33	33	35	38	37	36	37

Table B.1: Iteration counts of LSC and MSIMPLER preconditioner with two variants of mass matrix approximation for the steady-state BFS-2D problem with $\nu = 0.01$.

LSC		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
$\mathrm{diag}(\mathbf{M}_u)$	M1	6	9	7	12	10	7	16	14	11	8
	M2	6	9	6	11	9	7	15	12	10	7
	M3	5	9	6	10	8	6	14	11	9	6
	M4	5	8	5	10	8	6	14	11	9	7
$\widehat{\mathbf{M}}_{u,L}$	M1	5	8	6	11	9	7	15	12	11	7
	M2	5	8	6	10	8	6	13	10	9	6
	M3	4	7	5	9	7	5	12	9	8	5
	M4	4	7	5	9	6	5	12	9	8	6
MSIMPLER		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
$\mathrm{diag}(\mathbf{M}_u)$	M1	7	10	8	14	12	9	19	16	14	10
	M2	6	10	7	12	11	8	17	14	11	8
	M3	5	10	7	12	9	7	16	12	11	7
	M4	5	9	6	12	9	7	17	12	10	8
$\widehat{\mathbf{M}}_{u,L}$	M1	6	9	7	13	11	8	18	15	13	9
	M2	5	9	6	12	9	7	16	12	11	8
	M3	5	8	6	11	8	6	15	11	9	6
	M4	5	8	5	11	7	6	15	10	9	7

Table B.2: Iteration counts of LSC and MSIMPLER preconditioner with two variants of mass matrix approximation for the time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$.

B.2 PCD boundary conditions

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
$\mathbf{PCD}^{\text{orig}}_{\text{bc1}}$	M1	47	36	45	32	36	44	31	31	36	42
	M2	39	31	37	30	30	35	30	29	30	34
	M3	31	29	30	28	28	29	29	28	28	29
	M4	29	27	28	27	27	27	27	27	27	27
$\mathbf{PCD}^{\text{mod}}_{\text{bc2}}$	M1	38	95	49	262	142	66	>300	>300	211	98
	M2	35	112	49	>300	167	69	>300	>300	256	99
	M3	34	117	53	>300	194	75	>300	>300	>300	109
	M4	38	128	57	>300	206	79	>300	>300	>300	114
$\mathbf{PCD}^{\text{mod}}_{\text{bc3}}$	M1	44	98	53	265	144	68	>300	>300	213	100
	M2	39	114	50	>300	170	70	>300	>300	260	102
	M3	37	120	52	>300	186	72	>300	>300	>300	103
	M4	38	128	55	>300	192	75	>300	>300	>300	106
$\mathbf{PCD}^{\text{mod}}_{\text{bc4}}$	M1	39	27	35	25	24	31	24	24	23	27
	M2	30	23	27	23	22	23	23	22	22	22
	M3	23	22	22	22	22	22	22	21	21	21
	M4	22	22	22	21	21	21	21	21	21	21
$\mathbf{PCD}^{\text{mod}}_{\text{bc5}}$	M1	40	68	42	213	103	50	>300	>300	154	68
	M2	36	69	42	238	105	51	>300	>300	169	63
	M3	34	59	42	198	89	50	>300	>300	153	60
	M4	34	50	42	136	69	49	>300	>300	116	57
$\mathbf{PCD}^{\text{orig}}_{\text{bc6}}$	M1	106	171	111	>300	221	109	>300	>300	260	124
	M2	>300	>300	>300	>300	>300	155	>300	>300	>300	155
	M3	>300	>300	>300	>300	>300	>300	>300	>300	>300	>300
	M4	>300	>300	>300	>300	>300	>300	>300	>300	>300	>300

Table B.3: Comparison of all considered variants of boundary conditions for PCD, steady-state BFS-2D problem with $\nu = 0.01$.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
$\mathbf{PCD}^{\text{orig}}_{\text{bc1}}$	M1	19	26	21	32	27	25	37	31	30	27
	M2	27	35	27	42	34	31	48	41	37	33
	M3	37	47	36	52	46	39	53	50	45	41
	M4	48	53	46	54	50	45	53	51	50	46
$\mathbf{PCD}^{\text{mod}}_{\text{bc2}}$	M1	12	33	17	98	51	24	232	144	76	33
	M2	12	35	17	112	53	23	>300	174	86	32
	M3	11	36	16	118	57	23	>300	195	97	33
	M4	11	39	16	131	64	26	>300	222	111	37
$\mathbf{PCD}^{\text{mod}}_{\text{bc3}}$	M1	17	44	24	122	65	32	284	173	93	42
	M2	19	50	27	149	71	35	>300	220	108	44
	M3	19	55	29	158	78	37	>300	247	122	48
	M4	16	59	26	160	85	39	>300	267	139	53
$\mathbf{PCD}^{\text{mod}}_{\text{bc4}}$	M1	7	7	8	8	7	8	8	8	8	9
	M2	6	7	8	7	7	8	7	7	7	8
	M3	7	7	6	7	7	7	7	7	6	7
	M4	7	7	7	7	7	7	7	7	7	7
$\mathbf{PCD}^{\text{mod}}_{\text{bc5}}$	M1	12	31	17	83	46	23	182	124	68	31
	M2	12	28	16	71	41	21	187	114	61	26
	M3	11	23	14	53	31	16	175	94	48	21
	M4	11	18	13	36	23	14	161	76	38	17
$\mathbf{PCD}^{\text{orig}}_{\text{bc6}}$	M1	12	31	16	82	46	23	181	124	68	30
	M2	12	28	16	70	41	20	184	113	61	26
	M3	11	23	14	53	31	17	172	94	48	21
	M4	11	19	13	38	24	15	>300	105	62	17

Table B.4: Comparison of all considered variants of boundary conditions for PCD, time-dependent BFS-2D problem with $\nu = 0.01$ and $\Delta t = 0.01$.

B.3 Comparison of ideal versions

B.3.1 Lid-driven cavity 2D

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	26	25	27	33	29	27	43	33	31	27
	M2	30	27	30	34	30	31	44	33	32	33
	M3	32	35	35	39	37	38	47	40	40	40
	M4	41	45	45	49	49	50	53	52	52	53
PCD	M1	51	54	51	54	53	52	55	54	53	52
	M2	53	53	53	55	53	54	57	55	54	53
	M3	51	54	53	33	53	54	34	32	33	54
	M4	52	33	34	33	32	33	33	32	32	32
SPL	M1	80	96	81	168	119	82	>300	233	146	88
	M2	118	124	118	164	125	103	>300	229	147	97
	M3	152	170	148	194	158	133	>300	215	158	126
	M4	206	241	199	277	221	182	>300	247	212	172
SPLR	M1	24	23	22	26	27	22	34	30	29	22
	M2	25	33	24	36	26	22	45	25	26	22
	M3	26	57	25	63	26	24	80	32	26	23
	M4	41	106	35	117	33	29	146	54	31	31
MSPLR	M1	24	24	25	30	28	25	42	32	30	25
	M2	27	26	28	32	28	28	42	32	31	31
	M3	29	33	32	37	36	36	45	39	39	38
	M4	39	44	43	48	47	47	52	51	51	51
AL $\gamma = 2$	M1	5	6	5	5	6	6	6	5	5	8
	M2	5	4	5	4	4	4	5	4	4	4
	M3	5	3	3	4	3	3	5	3	3	3
MAL $\gamma = 0.02$	M1	29	49	37	113	79	64	252	176	143	128
	M2	32	48	37	105	75	63	243	167	139	125
	M3	36	52	43	59	74	63	129	87	72	121

Table B.5: Comparison of block preconditioners for the steady-state LDC-2D problem with $\nu = 0.003$, uniform meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

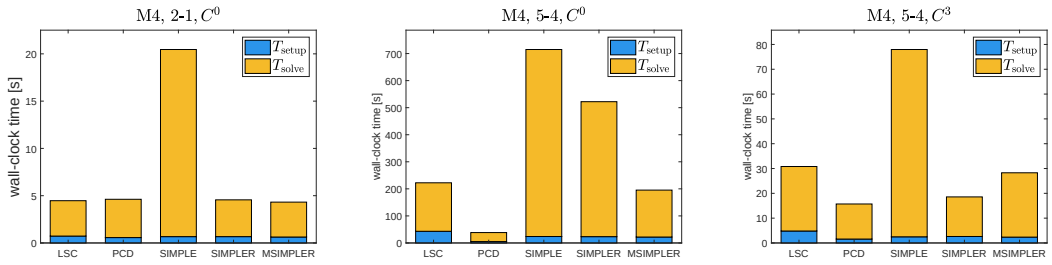


Figure B.1: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the steady-state LDC-2D problem with $\nu = 0.003$ on the mesh M4.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	26	25	27	33	29	27	43	33	31	27
	SM1	38	36	40	42	40	40	51	41	43	42
	SM2	62	60	67	68	67	69	80	70	70	71
	SM3	88	100	93	107	106	95	101	101	105	100
PCD	M1	51	54	51	54	53	52	55	54	53	52
	SM1	54	54	54	52	55	55	33	32	33	55
	SM2	53	31	33	31	31	32	31	30	30	31
	SM3	31	28	29	28	28	28	25	25	25	28
SPL	M1	80	96	81	168	119	82	>300	233	146	88
	SM1	90	99	87	154	105	81	>300	207	126	80
	SM2	106	123	102	188	126	90	>300	213	140	88
	SM3	112	146	109	235	148	96	295	189	127	96
SPLR	M1	24	23	22	26	27	22	34	30	29	22
	SM1	24	24	22	25	24	21	32	24	23	20
	SM2	23	24	22	26	23	21	37	23	24	20
	SM3	22	24	21	26	22	20	36	20	21	20
MSPLR	M1	24	24	25	30	28	25	42	32	30	25
	SM1	36	35	38	40	38	38	49	39	41	40
	SM2	59	59	66	67	65	67	80	69	71	71
	SM3	86	101	91	108	107	94	106	105	107	99
AL $\gamma = 2$	M1	5	6	5	5	6	6	6	5	5	8
	SM1	5	3	5	4	3	3	5	3	3	3
	SM2	3	3	3	3	3	3	3	3	3	3
MAL $\gamma = 0.02$	M1	29	49	37	113	79	64	252	176	143	128
	SM1	30	43	35	95	74	60	116	84	132	117
	SM2	31	22	18	39	29	26	75	56	46	46

Table B.6: Comparison of block preconditioners for the steady-state LDC-2D problem with $\nu = 0.003$, stretched meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	$\nu = 0.3$	14	15	15	16	16	16	16	16	16	16
	$\nu = 0.03$	18	20	20	21	21	21	22	22	22	22
	$\nu = 0.003$	32	35	35	39	37	38	47	40	40	40
	$\nu = 0.0003$	184	116	188	118	142	160	142	141	146	137
PCD	$\nu = 0.3$	11	10	10	10	10	10	10	10	10	10
	$\nu = 0.03$	18	18	18	17	17	18	17	17	17	17
	$\nu = 0.003$	51	54	53	33	53	54	34	32	33	54
	$\nu = 0.0003$	262	186	255	153	190	233	137	155	182	213
SPL	$\nu = 0.3$	67	84	64	99	75	57	118	86	72	54
	$\nu = 0.03$	72	90	70	105	80	62	127	93	77	59
	$\nu = 0.003$	152	170	148	194	158	133	>300	215	158	126
	$\nu = 0.0003$	>300	>300	>300	>300	>300	>300	>300	>300	>300	>300
SPLR	$\nu = 0.3$	23	45	15	52	20	15	63	29	19	13
	$\nu = 0.03$	25	49	18	55	21	17	66	31	20	15
	$\nu = 0.003$	26	57	25	63	26	24	80	32	26	23
	$\nu = 0.0003$	149	87	173	86	135	144	95	101	135	130
MSPLR	$\nu = 0.3$	18	20	19	22	21	21	23	23	23	23
	$\nu = 0.03$	20	22	22	24	24	23	25	25	25	25
	$\nu = 0.003$	29	33	32	37	36	36	45	39	39	38
	$\nu = 0.0003$	173	113	179	114	138	152	137	138	143	134
AL $\gamma = 2$	$\nu = 0.3$	5	9	7	17	13	11	27	23	20	18
	$\nu = 0.03$	4	5	4	7	5	5	12	9	8	8
	$\nu = 0.003$	5	3	3	4	3	3	5	3	3	3
	$\nu = 0.0003$	5	5	5	3	3	5	3	3	3	3
MAL $\gamma = 0.02$	$\nu = 0.3$	18	43	32	95	70	57	178	138	112	99
	$\nu = 0.03$	18	36	28	79	59	50	169	121	101	90
	$\nu = 0.003$	36	52	43	59	74	63	129	87	72	121
	$\nu = 0.0003$	147	245	191	>300	296	231	>300	>300	>300	265

Table B.7: Comparison of block preconditioners for the steady-state LDC-2D problem on the mesh M3, various viscosity values. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	5	8	5	10	8	6	14	10	9	6
	M2	5	7	5	10	7	5	13	9	8	5
	M3	4	7	5	10	7	5	13	9	8	5
	M4	5	7	5	10	7	5	13	9	7	6
PCD	M1	8	9	8	10	9	7	11	9	9	8
	M2	8	10	9	11	10	9	12	11	10	9
	M3	10	11	10	12	11	10	13	12	11	10
	M4	11	12	11	13	12	11	13	12	12	11
SPL	M1	11	31	16	82	48	21	183	124	68	28
	M2	10	23	15	63	40	19	162	108	61	25
	M3	7	17	12	43	28	14	118	73	43	19
	M4	12	17	11	39	24	12	105	63	36	15
SPLR	M1	5	10	6	14	12	7	20	16	13	7
	M2	5	11	6	13	12	7	19	14	12	7
	M3	4	9	6	13	10	6	19	12	10	7
	M4	4	8	5	13	9	5	19	11	9	6
MSPLR	M1	5	9	6	12	10	7	17	12	12	7
	M2	5	10	6	11	9	6	15	10	11	7
	M3	5	8	5	11	8	6	15	10	9	6
	M4	5	8	5	12	8	6	16	11	9	7
AL $\gamma = 10$	M1	5	4	4	4	4	4	3	4	4	4
	M2	5	4	4	3	3	4	3	3	3	3
	M3	4	3	3	3	3	3	3	3	3	3
MAL $\gamma = 0.2$	M1	28	26	25	19	16	24	32	22	19	32
	M2	30	22	27	28	20	16	39	28	20	18
	M3	34	30	22	42	31	20	54	43	33	21

Table B.8: Comparison of block preconditioners for the time-dependent LDC-2D problem with $\nu = 0.003$ and $\Delta t = 0.01$, uniform meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

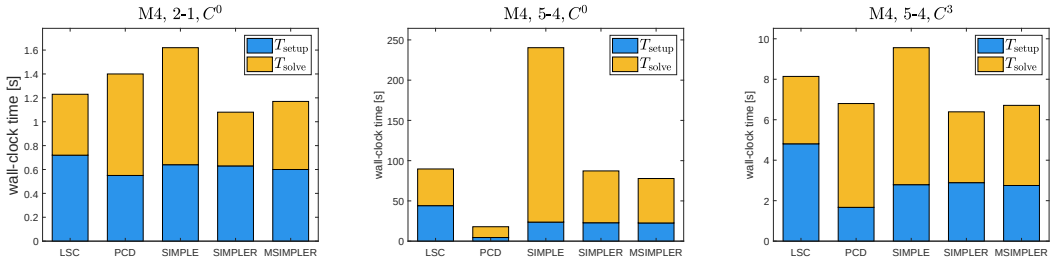


Figure B.2: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the time-dependent LDC-2D problem with $\nu = 0.003$ and $\Delta t = 0.01$ on the mesh M4.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	5	8	5	10	8	6	14	10	9	6
	SM1	4	6	4	8	6	4	12	8	6	5
	SM2	6	8	6	12	7	6	16	8	7	7
	SM3	9	11	10	14	11	10	18	12	13	11
PCD	M1	8	9	8	10	9	7	11	9	9	8
	SM1	9	11	9	11	10	9	11	10	10	9
	SM2	10	10	10	10	10	9	11	10	10	9
	SM3	10	10	10	10	10	9	10	10	9	9
SPL	M1	11	31	16	82	48	21	183	124	68	28
	SM1	10	28	14	72	38	17	165	109	55	22
	SM2	17	34	17	72	38	17	160	101	50	20
	SM3	27	52	27	94	55	24	172	107	60	25
SPLR	M1	5	10	6	14	12	7	20	16	13	7
	SM1	4	8	5	11	9	5	16	12	9	5
	SM2	5	9	5	13	9	5	21	11	9	5
	SM3	6	10	6	14	9	5	24	11	9	5
MSPLR	M1	5	9	6	12	10	7	17	12	12	7
	SM1	5	8	5	10	8	5	14	9	8	5
	SM2	7	11	7	16	9	8	22	11	11	9
	SM3	12	16	15	24	17	16	35	20	21	17
AL $\gamma = 10$	M1	5	4	4	4	4	4	3	4	4	4
	SM1	4	4	4	3	3	4	3	3	3	3
	SM2	4	3	3	3	3	3	2	2	3	3
MAL $\gamma = 0.2$	M1	28	26	25	19	16	24	32	22	19	32
	SM1	30	21	17	30	21	16	40	31	22	16
	SM2	22	30	20	42	31	18	55	42	30	17

Table B.9: Comparison of block preconditioners for the time-dependent LDC-2D problem with $\nu = 0.003$ and $\Delta t = 0.01$, stretched meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	$\nu = 0.3$	7	8	8	9	8	8	11	9	9	9
	$\nu = 0.03$	5	7	5	10	6	6	13	8	7	6
	$\nu = 0.003$	4	7	5	10	7	5	13	9	8	5
	$\nu = 0.0003$	10	20	9	18	13	8	28	21	14	8
PCD	$\nu = 0.3$	9	9	9	9	9	9	9	9	9	9
	$\nu = 0.03$	10	11	10	11	10	10	11	10	10	10
	$\nu = 0.003$	10	11	10	12	11	10	13	12	11	10
	$\nu = 0.0003$	12	16	12	16	15	12	15	15	14	11
SPL	$\nu = 0.3$	45	61	43	76	53	37	98	65	50	35
	$\nu = 0.03$	16	23	15	37	23	13	97	59	33	14
	$\nu = 0.003$	7	17	12	43	28	14	118	73	43	19
	$\nu = 0.0003$	12	48	20	213	91	27	>300	>300	149	38
SPLR	$\nu = 0.3$	11	28	8	32	10	8	42	15	11	7
	$\nu = 0.03$	6	10	5	14	9	5	19	12	9	6
	$\nu = 0.003$	4	9	6	13	10	6	19	12	10	7
	$\nu = 0.0003$	8	25	10	32	28	10	39	30	25	11
MSPLR	$\nu = 0.3$	10	12	11	13	13	12	18	13	13	13
	$\nu = 0.03$	6	9	6	13	8	7	17	11	10	8
	$\nu = 0.003$	5	8	5	11	8	6	15	10	9	6
	$\nu = 0.0003$	9	24	9	29	19	9	32	28	23	9
AL $\gamma = 10$	$\nu = 0.3$	4	4	4	7	5	5	7	7	7	7
	$\nu = 0.03$	4	3	4	3	3	3	3	3	3	3
	$\nu = 0.003$	4	3	3	3	3	3	3	3	3	3
	$\nu = 0.0003$	4	3	3	3	3	3	3	3	3	3
MAL $\gamma = 0.2$	$\nu = 0.3$	19	29	22	56	42	35	95	76	63	56
	$\nu = 0.03$	20	20	17	31	22	18	61	43	35	31
	$\nu = 0.003$	34	30	22	42	31	20	54	43	33	21
	$\nu = 0.0003$	39	76	39	101	60	39	145	103	66	39

Table B.10: Comparison of block preconditioners for the time-dependent LDC-2D problem with $\Delta t = 0.01$ on the mesh M3, various viscosity values. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

B.3.2 Backward-facing step 2D

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	18	14	19	18	16	18	25	21	18	19
	M2	18	17	21	21	19	21	27	22	21	21
	M3	21	23	25	26	26	27	31	29	28	28
	M4	29	32	32	35	35	35	38	38	38	38
PCD	M1	23	21	22	22	20	20	22	21	20	20
	M2	21	21	20	21	20	19	20	20	20	19
	M3	20	20	19	19	19	19	19	19	19	19
	M4	19	19	19	19	18	19	18	18	18	18
SPL	M1	69	80	77	138	108	77	271	214	141	91
	M2	97	98	93	130	105	84	275	203	133	85
	M3	124	134	120	153	130	112	246	178	132	108
	M4	164	187	159	217	178	151	247	196	171	146
SPLR	M1	18	20	16	20	16	16	28	18	18	16
	M2	19	30	17	32	17	16	42	22	19	15
	M3	26	57	20	64	23	19	81	35	22	18
	M4	45	110	28	122	38	28	151	62	37	24
MSPLR	M1	17	14	18	18	16	18	25	20	19	18
	M2	17	16	20	20	18	20	27	22	21	20
	M3	20	22	24	25	25	26	31	28	27	27
	M4	27	31	31	34	34	34	37	37	37	36
AL $\gamma = 2$	M1	6	6	6	5	5	7	5	5	5	7
	M2	6	5	5	5	5	5	4	4	5	5
	M3	6	5	5	4	4	5	4	4	4	4
MAL $\gamma = 0.1$	M1	24	26	28	41	37	42	85	70	67	77
	M2	21	23	20	34	28	27	61	48	44	44
	M3	20	22	19	29	24	21	41	36	35	29

Table B.11: Comparison of block preconditioners for the steady-state BFS-2D problem with $\nu = 0.02$, uniform meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

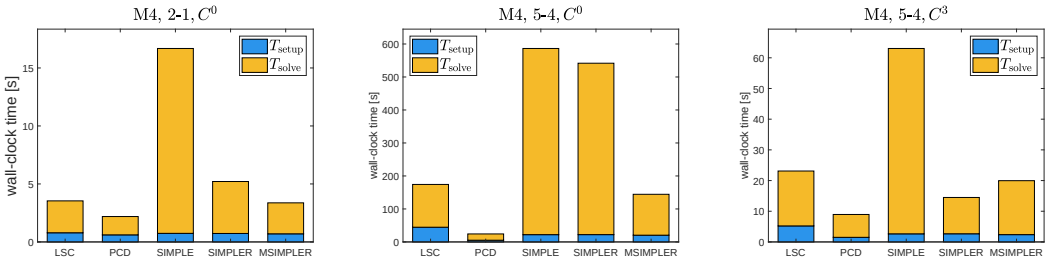


Figure B.3: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the steady-state BFS-2D problem with $\nu = 0.02$ on the mesh M4.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	18	14	19	18	16	18	25	21	18	19
	SM1	20	20	21	31	21	21	42	31	24	24
	SM2	24	28	26	40	28	27	66	45	32	34
	SM3	30	36	31	48	38	35	72	47	41	48
PCD	M1	23	21	22	22	20	20	22	21	20	20
	SM1	22	21	20	21	20	20	21	21	20	20
	SM2	22	21	20	21	20	20	21	21	20	19
	SM3	22	21	20	21	20	20	21	20	20	19
SPL	M1	69	80	77	138	108	77	271	214	141	91
	SM1	89	111	95	170	130	95	>300	244	159	101
	SM2	104	161	116	246	175	122	>300	291	197	128
	SM3	119	191	135	>300	207	139	>300	>300	227	146
SPLR	M1	18	20	16	20	16	16	28	18	18	16
	SM1	20	24	22	33	22	21	51	30	25	22
	SM2	30	32	31	45	31	31	78	41	36	31
	SM3	40	36	39	53	38	36	94	50	43	37
MSPLR	M1	17	14	18	18	16	18	25	20	19	18
	SM1	19	20	20	30	22	21	41	31	25	23
	SM2	25	28	26	40	28	27	66	44	34	35
	SM3	31	37	32	48	39	36	73	48	43	52
AL $\gamma = 2$	M1	6	6	6	5	5	7	5	5	5	7
	SM1	6	6	6	5	5	6	5	5	5	6
	SM2	6	6	6	5	5	6	5	5	5	6
MAL $\gamma = 0.1$	M1	24	26	28	41	37	42	85	70	67	77
	SM1	23	25	26	31	32	31	51	43	54	39
	SM2	23	25	24	28	32	28	34	29	53	32

Table B.12: Comparison of block preconditioners for the steady-state BFS-2D problem with $\nu = 0.02$, stretched meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	$\nu = 0.2$	25	26	27	28	29	29	30	29	30	30
	$\nu = 0.02$	24	25	28	28	28	30	34	31	31	31
	$\nu = 0.002$	68	45	80	40	53	71	49	44	50	64
PCD	$\nu = 0.2$	22	21	22	21	21	22	21	21	21	21
	$\nu = 0.02$	20	20	20	20	20	19	20	19	19	19
	$\nu = 0.002$	80	51	74	45	47	64	44	44	44	58
SPL	$\nu = 0.2$	90	110	91	129	104	87	143	115	101	84
	$\nu = 0.02$	165	179	159	207	174	147	297	217	172	141
	$\nu = 0.002$	>300	>300	>300	>300	>300	>300	>300	>300	>300	>300
SPLR	$\nu = 0.2$	38	73	29	80	34	29	95	49	34	28
	$\nu = 0.02$	31	73	23	81	26	22	106	43	26	21
	$\nu = 0.002$	88	117	83	67	71	75	91	45	73	73
MSPLR	$\nu = 0.2$	26	29	28	31	31	31	33	33	33	32
	$\nu = 0.02$	23	24	27	28	27	28	34	31	30	30
	$\nu = 0.002$	69	52	78	39	63	71	48	43	56	65
AL $\gamma = 2$	$\nu = 0.2$	9	8	8	7	7	7	—	7	7	7
	$\nu = 0.02$	6	5	6	4	4	5	—	4	5	5
	$\nu = 0.002$	5	4	5	4	4	5	—	4	4	4
MAL $\gamma = 0.1$	$\nu = 0.2$	30	42	34	50	44	39	—	65	75	58
	$\nu = 0.02$	27	26	24	29	25	23	—	37	41	29
	$\nu = 0.002$	50	74	62	100	85	71	—	109	96	79

Table B.13: Comparison of block preconditioners for the steady-state BFS-2D problem on the mesh M3, various viscosity values. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	5	8	6	11	9	7	14	11	11	7
	M2	5	7	5	9	7	5	12	9	8	6
	M3	4	7	5	9	6	5	12	9	8	6
	M4	4	7	5	9	6	5	13	9	8	6
PCD	M1	7	7	8	8	7	8	8	7	8	8
	M2	6	7	7	7	6	7	7	6	7	7
	M3	7	7	6	7	7	6	7	7	7	6
	M4	7	7	7	7	7	7	7	7	7	7
SPL	M1	8	23	12	61	36	18	140	97	56	26
	M2	7	18	11	45	29	15	112	76	45	20
	M3	5	11	8	28	18	10	74	48	29	15
	M4	4	9	6	22	14	8	61	37	22	10
SPLR	M1	6	10	7	14	12	8	19	15	13	9
	M2	5	9	7	12	10	7	17	13	11	7
	M3	4	8	6	11	9	6	16	12	10	7
	M4	4	7	5	11	8	6	17	10	9	7
MSPLR	M1	6	9	7	13	11	8	18	14	13	9
	M2	5	8	6	12	9	6	16	11	10	7
	M3	4	8	5	11	7	6	15	10	9	6
	M4	4	8	5	11	7	6	16	11	9	7
AL $\gamma = 10$	M1	17	14	15	13	13	13	12	12	12	12
	M2	17	14	15	13	12	13	11	11	12	12
	M3	17	13	14	12	12	13	9	9	11	12
MAL $\gamma = 10$	M1	20	23	19	28	23	20	30	28	25	20
	M2	25	31	24	35	28	25	42	36	31	25
	M3	31	37	28	47	38	29	54	49	42	31

Table B.14: Comparison of block preconditioners for the time-dependent BFS-2D problem with $\nu = 0.02$ and $\Delta t = 0.01$, uniform meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

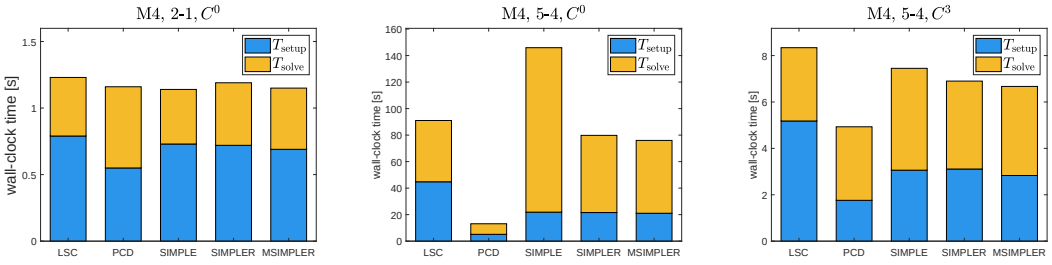


Figure B.4: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the time-dependent BFS-2D problem with $\nu = 0.02$ on the mesh M4.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	M1	5	8	6	11	9	7	14	11	11	7
	SM1	5	9	5	16	9	6	23	16	12	8
	SM2	5	9	6	18	10	7	36	21	14	9
	SM3	8	13	11	21	12	12	40	21	15	13
PCD	M1	7	7	8	8	7	8	8	7	8	8
	SM1	6	7	7	6	6	6	6	6	6	7
	SM2	6	6	6	6	6	6	6	6	6	6
	SM3	6	6	6	6	6	6	6	6	6	6
SPL	M1	8	23	12	61	36	18	140	97	56	26
	SM1	8	21	12	58	33	16	137	90	51	23
	SM2	12	24	13	59	34	17	141	90	51	24
	SM3	22	41	23	77	46	25	159	100	58	30
SPLR	M1	6	10	7	14	12	8	19	15	13	9
	SM1	6	10	7	18	12	8	30	20	15	9
	SM2	8	12	9	21	13	9	40	24	16	11
	SM3	19	24	20	30	24	21	51	30	24	23
MSPLR	M1	6	9	7	13	11	8	18	14	13	9
	SM1	5	11	6	19	12	8	28	20	14	9
	SM2	6	11	7	22	13	9	44	26	17	11
	SM3	10	16	12	26	14	13	48	26	18	16
AL $\gamma = 10$	M1	17	14	15	13	13	13	12	12	12	12
	SM1	18	15	15	13	13	13	12	12	12	12
	SM2	18	15	15	13	13	13	12	12	12	12
MAL $\gamma = 10$	M1	20	23	19	28	23	20	30	28	25	20
	SM1	30	37	31	51	38	31	59	52	42	30
	SM2	41	55	40	68	58	41	78	69	60	42

Table B.15: Comparison of block preconditioners for the time-dependent BFS-2D problem with $\nu = 0.02$ and $\Delta t = 0.01$, stretched meshes. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2	5-4, C^0	5-4, C^1	5-4, C^2	5-4, C^3
LSC	$\nu = 0.2$	5	7	5	9	7	6	13	9	8	7
	$\nu = 0.02$	4	7	5	9	6	5	12	9	8	6
	$\nu = 0.002$	5	8	6	10	9	6	14	11	10	6
	$\nu = 0.0002$	5	8	6	12	10	7	17	13	12	7
PCD	$\nu = 0.2$	8	8	7	8	7	7	8	7	7	7
	$\nu = 0.02$	7	7	6	7	7	6	7	7	7	6
	$\nu = 0.002$	6	7	8	7	7	9	7	7	8	9
	$\nu = 0.0002$	6	8	9	10	7	9	10	8	10	11
SPL	$\nu = 0.2$	8	11	8	27	17	10	72	46	27	14
	$\nu = 0.02$	5	11	8	28	18	10	74	48	29	15
	$\nu = 0.002$	7	21	10	75	36	14	234	132	59	22
	$\nu = 0.0002$	7	25	10	130	44	16	>300	223	77	24
SPLR	$\nu = 0.2$	5	9	5	13	9	6	19	12	10	7
	$\nu = 0.02$	4	8	6	11	9	6	16	12	10	7
	$\nu = 0.002$	6	12	7	13	13	8	20	16	13	8
	$\nu = 0.0002$	6	14	8	16	15	9	34	23	16	9
MSPLR	$\nu = 0.2$	5	9	6	12	9	7	17	12	11	8
	$\nu = 0.02$	4	8	5	11	7	6	15	10	9	6
	$\nu = 0.002$	5	9	6	12	9	7	16	12	12	7
	$\nu = 0.0002$	6	11	7	13	11	7	21	14	14	8
AL $\gamma = 10$	$\nu = 0.2$	13	10	14	9	9	12	8	8	9	11
	$\nu = 0.02$	17	13	14	12	12	13	9	9	11	12
	$\nu = 0.002$	17	13	14	12	12	13	10	10	11	12
	$\nu = 0.0002$	17	13	14	12	12	13	10	10	11	12
MAL $\gamma = 10$	$\nu = 0.2$	20	25	20	30	25	21	35	30	27	22
	$\nu = 0.02$	31	37	28	47	38	29	54	49	42	31
	$\nu = 0.002$	32	45	31	65	49	35	86	70	55	38
	$\nu = 0.0002$	33	48	32	74	53	36	107	81	61	40

Table B.16: Comparison of block preconditioners for the time-dependent BFS-2D problem on the mesh M3, various viscosity values. (The abbreviations "SPL", "SPLR" and "MSPLR" stand for the corresponding SIMPLE-type preconditioners.)

B.3.3 BFS-2D: eigenvalues of preconditioned matrix

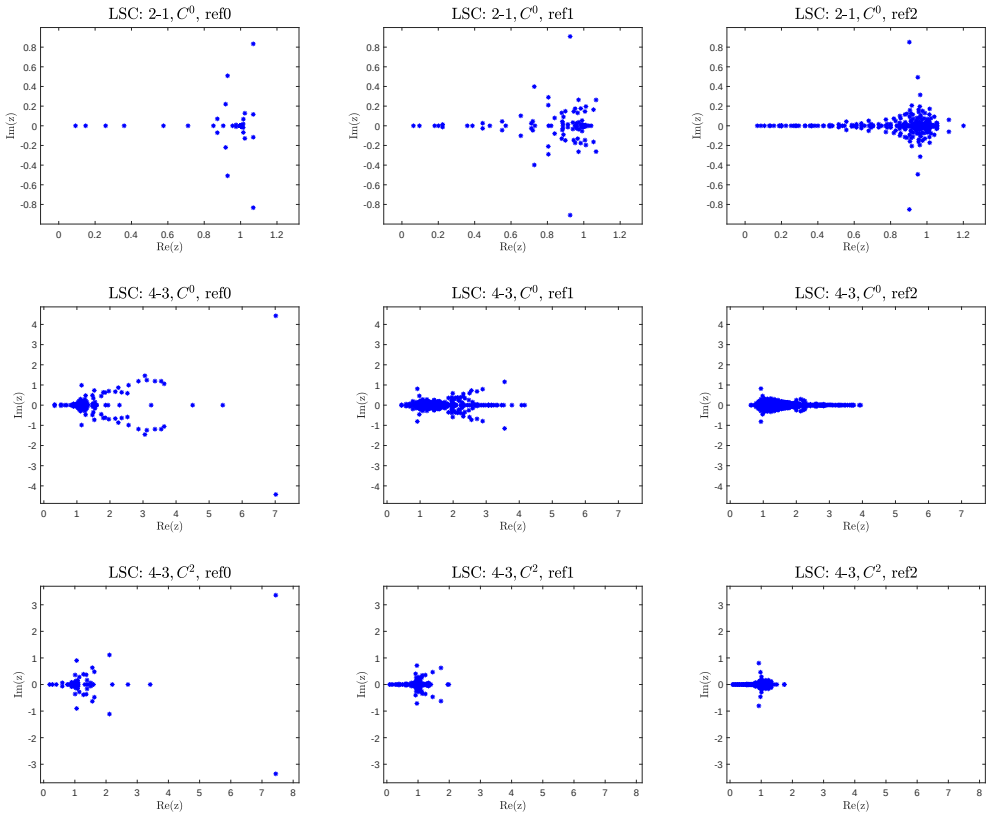


Figure B.5: LSC, steady-state BFS-2D, $\nu = 0.01$.

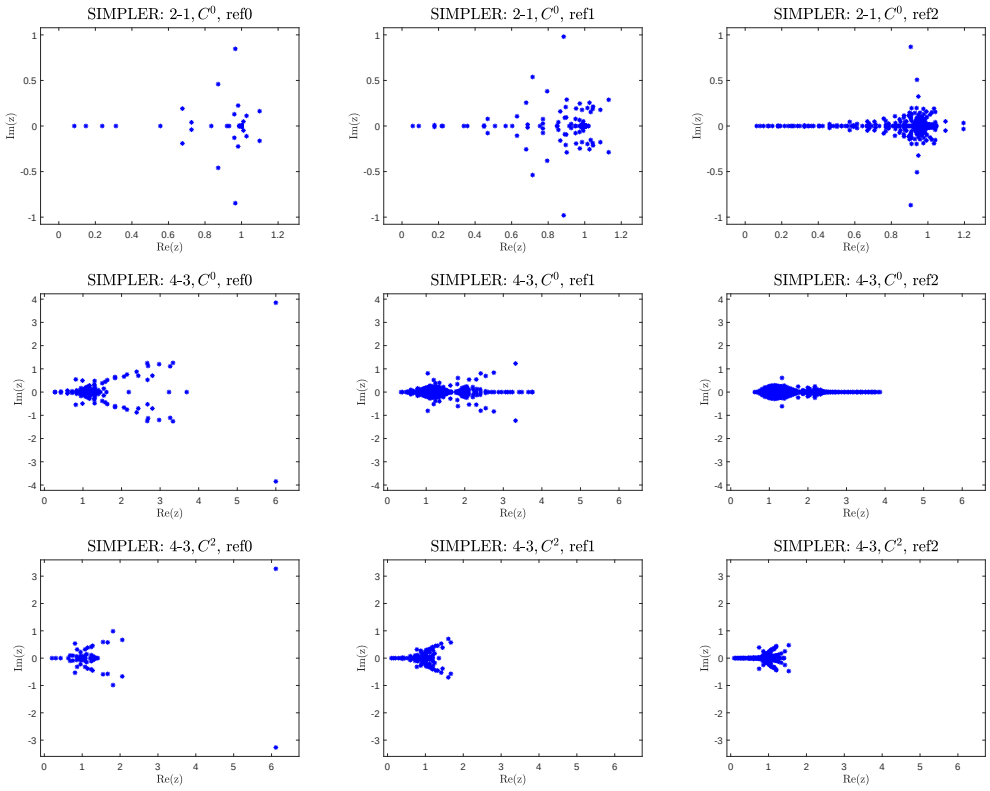


Figure B.6: SIMPLER, steady-state BFS-2D, $\nu = 0.01$.

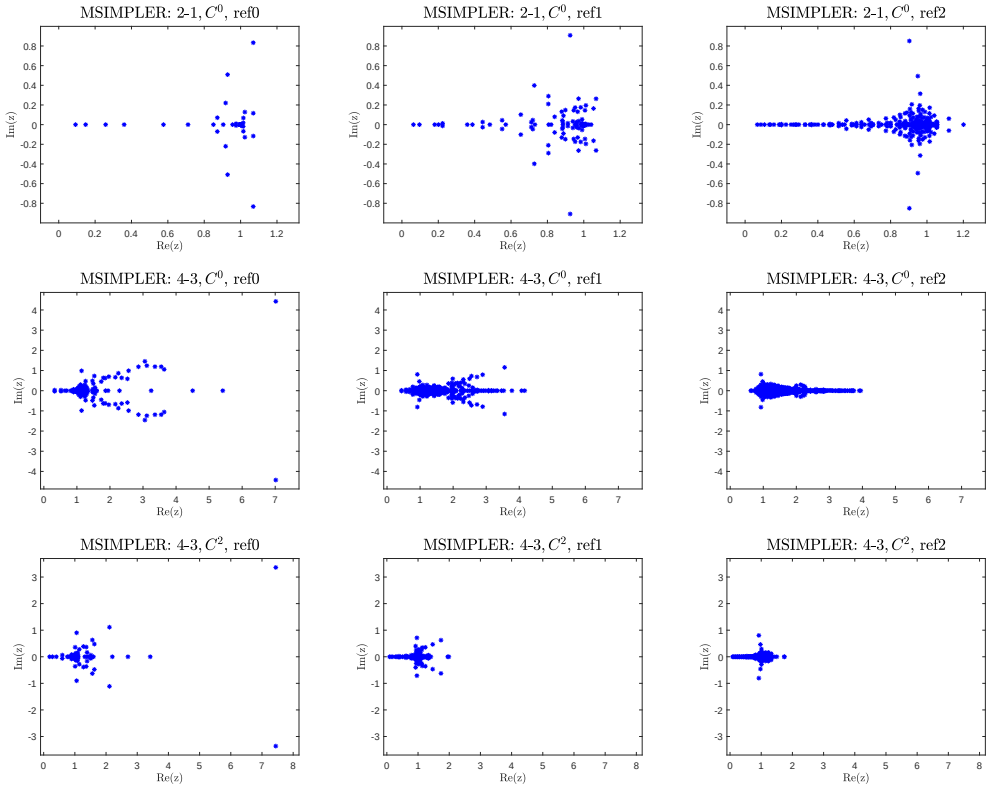


Figure B.7: MSIMPLER, steady-state BFS-2D, $\nu = 0.01$.

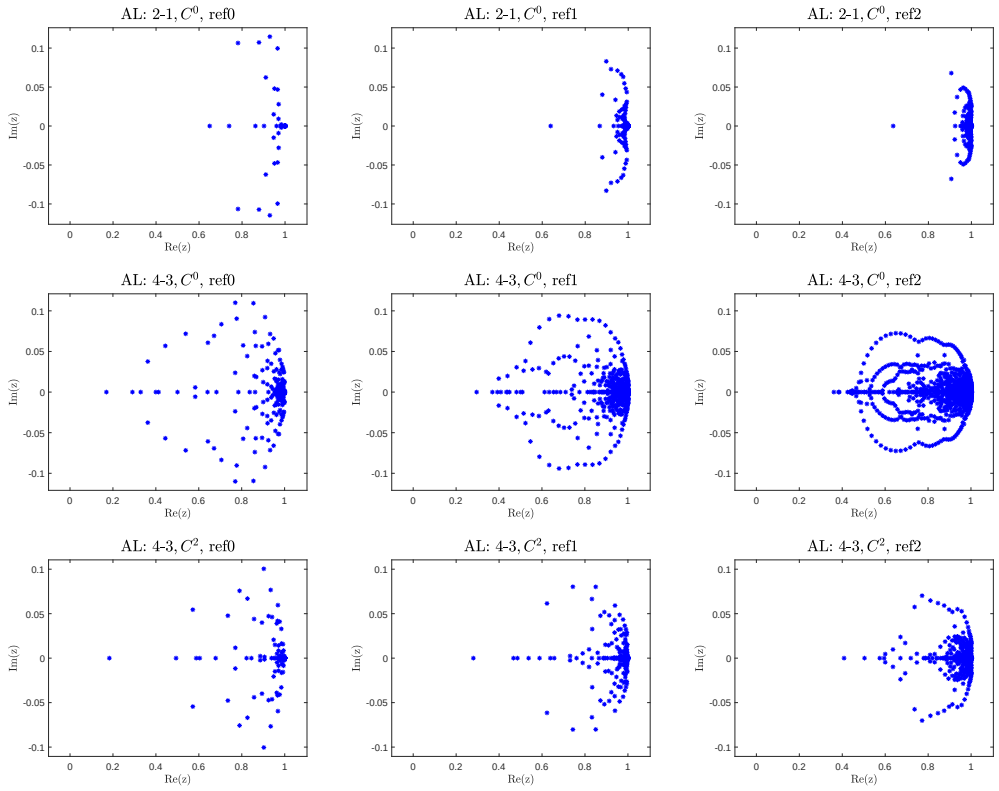


Figure B.8: AL, steady-state BFS-2D, $\nu = 0.01$.

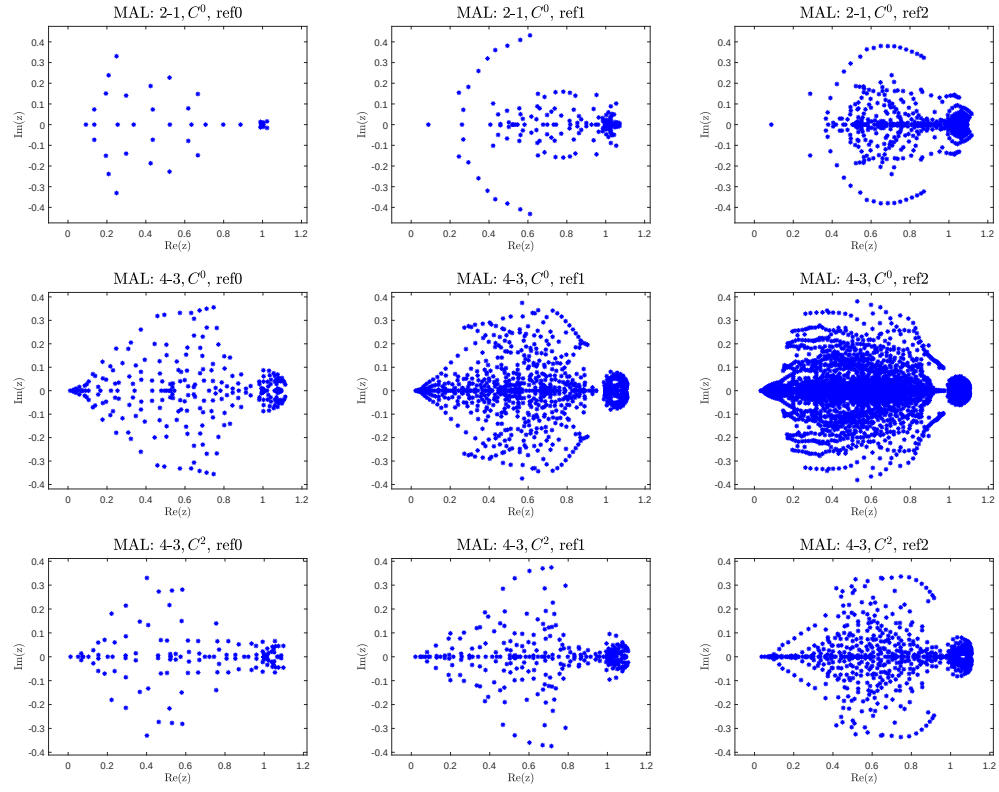


Figure B.9: MAL, steady-state BFS-2D, $\nu = 0.01$.

B.3.4 Lid-driven cavity 3D

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	M1	10	15	11	22	18	12
	M2	13	15	13	23	17	15
	M3	16	17	17	23	19	19
PCD	M1	43	55	46	62	59	52
	M2	49	62	55	34	33	58
	M3	56	33	31	33	32	31
SIMPLE	M1	18	59	31	190	116	53
	M2	29	69	38	235	141	55
	M3	49	70	50	227	137	55
SIMPLER	M1	8	13	10	20	18	11
	M2	11	14	11	20	18	12
	M3	15	24	12	29	16	12
MSIMPLER	M1	9	14	10	23	18	12
	M2	11	13	12	21	16	13
	M3	14	15	15	21	17	17
AL	M1	9	18	17	45	30	31
$\gamma = 2$	M2	9	13	9	30	22	16
MAL	M1	56	287	133	>300	>300	>300
$\gamma = 0.02$	M2	68	187	164	>300	>300	230

Table B.17: Comparison of block preconditioners for the steady-state LDC-3D problem with $\nu = 0.01$, uniform meshes.

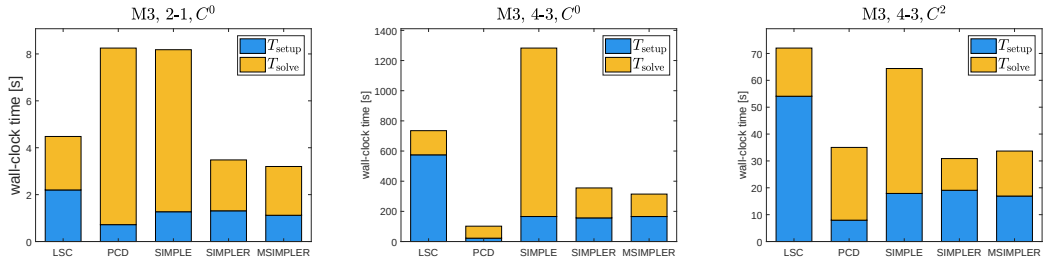


Figure B.10: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the steady-state LDC-3D problem with $\nu = 0.01$ on the mesh M3.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	$\nu = 0.1$	9	10	9	14	11	10
	$\nu = 0.05$	9	11	10	15	11	11
	$\nu = 0.01$	13	15	13	23	17	15
	$\nu = 0.005$	17	19	17	28	24	18
PCD	$\nu = 0.1$	36	21	20	21	20	19
	$\nu = 0.05$	40	24	41	24	23	22
	$\nu = 0.01$	49	62	55	34	33	58
	$\nu = 0.005$	59	72	63	40	38	67
SIMPLE	$\nu = 0.1$	18	39	20	135	79	29
	$\nu = 0.05$	19	41	21	142	83	32
	$\nu = 0.01$	29	69	38	235	141	55
	$\nu = 0.005$	40	111	56	>300	218	84
SIMPLER	$\nu = 0.1$	8	13	7	18	13	8
	$\nu = 0.05$	9	13	7	18	13	8
	$\nu = 0.01$	11	14	11	20	18	12
	$\nu = 0.005$	15	17	15	22	24	16
MSIMPLER	$\nu = 0.1$	8	11	9	16	12	10
	$\nu = 0.05$	8	11	9	16	12	10
	$\nu = 0.01$	11	13	12	21	16	13
	$\nu = 0.005$	15	17	15	26	23	16
AL $\gamma = 2$	$\nu = 0.1$	17	36	21	79	60	36
	$\nu = 0.05$	8	27	16	60	46	28
	$\nu = 0.01$	9	13	9	30	22	16
	$\nu = 0.005$	8	10	13	23	17	14
MAL $\gamma = 0.02$	$\nu = 0.1$	75	171	86	>300	282	141
	$\nu = 0.05$	76	181	89	>300	>300	156
	$\nu = 0.01$	68	187	164	>300	>300	230
	$\nu = 0.005$	64	>300	160	>300	>300	>300

Table B.18: Comparison of block preconditioners for the steady-state LDC-3D problem on the mesh M2, various viscosity values.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	M1	5	9	6	13	11	8
	M2	5	8	6	12	10	7
	M3	5	8	6	11	8	6
PCD	M1	9	9	9	10	8	8
	M2	9	11	9	12	11	9
	M3	10	12	10	13	11	10
SIMPLE	M1	16	70	28	220	123	50
	M2	17	70	31	237	138	48
	M3	14	52	25	184	119	40
SIMPLER	M1	6	11	7	18	16	9
	M2	5	10	7	16	14	9
	M3	4	10	6	15	12	7
MSIMPLER	M1	6	10	8	16	14	9
	M2	6	10	7	15	11	8
	M3	5	9	6	13	9	7
AL $\gamma = 10$	M1	7	8	8	11	9	9
	M2	6	6	7	8	6	7
MAL $\gamma = 0.2$	M1	13	31	21	58	41	32
	M2	17	35	22	64	44	32

Table B.19: Comparison of block preconditioners for the time-dependent LDC-3D problem with $\nu = 0.01$ and $\Delta t = 0.01$, uniform meshes.

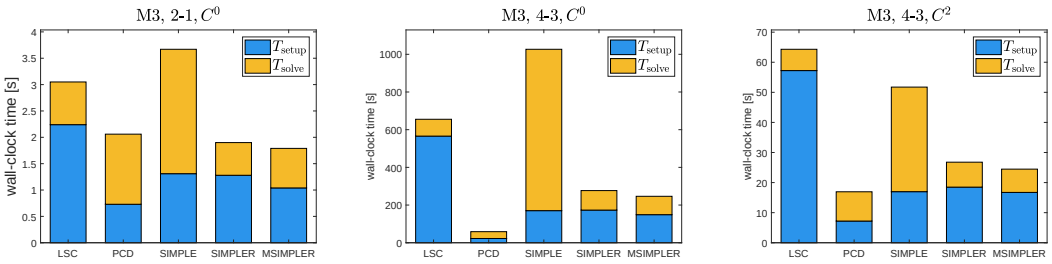


Figure B.11: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the time-dependent LDC-3D problem with $\nu = 0.01$ and $\Delta t = 0.01$ on the mesh M3.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	$\nu = 0.1$	4	8	5	11	8	6
	$\nu = 0.05$	4	8	6	11	8	6
	$\nu = 0.01$	5	8	6	12	10	7
	$\nu = 0.005$	5	9	7	12	10	8
PCD	$\nu = 0.1$	10	12	10	13	12	11
	$\nu = 0.05$	10	12	10	13	11	10
	$\nu = 0.01$	9	11	9	12	11	9
	$\nu = 0.005$	9	9	8	11	9	8
SIMPLE	$\nu = 0.1$	11	42	22	145	88	37
	$\nu = 0.05$	14	49	26	160	102	41
	$\nu = 0.01$	17	70	31	237	138	48
	$\nu = 0.005$	17	78	32	273	148	50
SIMPLER	$\nu = 0.1$	4	10	6	16	12	7
	$\nu = 0.05$	4	10	7	16	13	8
	$\nu = 0.01$	5	10	7	16	14	9
	$\nu = 0.005$	5	10	7	17	15	9
MSIMPLER	$\nu = 0.1$	5	10	7	14	11	7
	$\nu = 0.05$	5	9	7	14	10	7
	$\nu = 0.01$	6	10	7	15	11	8
	$\nu = 0.005$	6	10	8	15	12	8
AL $\gamma = 10$	$\nu = 0.1$	7	13	8	22	16	11
	$\nu = 0.05$	7	10	7	16	12	9
	$\nu = 0.01$	6	6	7	8	6	7
	$\nu = 0.005$	6	6	7	7	6	7
MAL $\gamma = 0.2$	$\nu = 0.1$	21	74	40	147	97	56
	$\nu = 0.05$	20	58	34	117	78	48
	$\nu = 0.01$	17	35	22	64	44	32
	$\nu = 0.005$	20	29	21	50	33	29

Table B.20: Comparison of block preconditioners for the time-dependent LDC-3D problem with $\Delta t = 0.01$ on the mesh M2, various viscosity values.

B.3.5 Backward-facing step 3D

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	M1	22	25	23	28	32	23
	M2	27	21	27	22	25	26
	M3	26	17	29	—	20	27
PCD	M1	31	34	31	28	30	28
	M2	35	27	32	25	24	29
	M3	30	24	27	—	22	24
SIMPLE	M1	51	212	129	>300	>300	229
	M2	93	178	148	>300	>300	185
	M3	133	140	133	—	299	137
SIMPLER	M1	23	26	27	28	36	27
	M2	28	26	28	21	31	24
	M3	28	31	25	—	24	22
MSIMPLER	M1	22	25	21	29	34	24
	M2	26	21	26	22	28	25
	M3	24	17	27	—	21	25
AL $\gamma = 2$	M1	8	13	12	29	23	24
	M2	8	8	11	—	—	18
MAL $\gamma = 0.1$	M1	36	68	53	181	128	102
	M2	40	61	56	—	—	104

Table B.21: Comparison of block preconditioners for the steady-state BFS-3D problem with $\nu = 0.01$, uniform meshes.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	$\nu = 0.1$	12	13	13	17	14	14
	$\nu = 0.05$	12	13	14	18	15	14
	$\nu = 0.01$	27	21	27	22	25	26
PCD	$\nu = 0.1$	17	17	17	17	17	17
	$\nu = 0.05$	18	19	17	19	18	17
	$\nu = 0.01$	35	27	32	25	24	29
SIMPLE	$\nu = 0.1$	28	44	32	131	92	43
	$\nu = 0.05$	34	60	43	179	123	60
	$\nu = 0.01$	93	178	148	>300	>300	185
SIMPLER	$\nu = 0.1$	13	18	11	22	16	12
	$\nu = 0.05$	13	17	11	20	16	12
	$\nu = 0.01$	28	26	28	21	31	24
MSIMPLER	$\nu = 0.1$	12	13	13	17	14	14
	$\nu = 0.05$	11	13	13	17	14	13
	$\nu = 0.01$	26	21	26	22	28	25
AL $\gamma = 2$	$\nu = 0.1$	8	13	11	—	—	22
	$\nu = 0.05$	7	12	9	—	—	17
	$\nu = 0.01$	8	8	11	—	—	18
MAL $\gamma = 0.1$	$\nu = 0.1$	29	61	45	—	—	92
	$\nu = 0.05$	29	61	45	—	—	89
	$\nu = 0.01$	40	61	56	—	—	104

Table B.22: Comparison of block preconditioners for the steady-state BFS-3D problem on the mesh M2, various viscosity values.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	M1	6	9	7	13	12	10
	M2	5	9	6	11	10	8
	M3	5	8	6	—	8	6
PCD	M1	9	11	11	11	13	14
	M2	9	10	10	10	9	11
	M3	8	9	10	—	9	11
SIMPLE	M1	9	50	22	175	98	45
	M2	9	54	21	205	107	36
	M3	8	45	19	—	98	30
SIMPLER	M1	7	12	10	17	15	12
	M2	7	11	8	15	14	9
	M3	6	11	7	—	12	8
MSIMPLER	M1	7	11	9	18	15	11
	M2	6	10	8	15	12	9
	M3	6	9	6	—	10	7
AL $\gamma = 10$	M1	14	15	16	17	17	18
	M2	16	17	16	—	—	17
MAL $\gamma = 10$	M1	16	17	18	20	19	20
	M2	18	19	18	—	—	19

Table B.23: Comparison of block preconditioners for the time-dependent BFS-3D problem with $\nu = 0.01$ and $\Delta t = 0.01$, uniform meshes.

		2-1, C^0	3-2, C^0	3-2, C^1	4-3, C^0	4-3, C^1	4-3, C^2
LSC	$\nu = 0.1$	5	7	5	10	8	6
	$\nu = 0.05$	5	7	6	10	8	6
	$\nu = 0.01$	5	9	6	11	10	8
PCD	$\nu = 0.1$	8	9	10	8	8	9
	$\nu = 0.05$	8	9	10	9	8	10
	$\nu = 0.01$	9	10	10	10	9	11
SIMPLE	$\nu = 0.1$	9	38	20	125	82	32
	$\nu = 0.05$	9	44	21	148	92	34
	$\nu = 0.01$	9	54	21	205	107	36
SIMPLER	$\nu = 0.1$	6	10	7	13	11	7
	$\nu = 0.05$	6	10	8	13	12	8
	$\nu = 0.01$	7	11	8	15	14	9
MSIMPLER	$\nu = 0.1$	5	9	6	13	10	7
	$\nu = 0.05$	5	9	6	13	10	8
	$\nu = 0.01$	6	10	8	15	12	9
AL $\gamma = 10$	$\nu = 0.1$	16	16	16	—	—	17
	$\nu = 0.05$	16	16	16	—	—	17
	$\nu = 0.01$	16	17	16	—	—	17
MAL $\gamma = 10$	$\nu = 0.1$	18	19	19	—	—	19
	$\nu = 0.05$	18	19	19	—	—	19
	$\nu = 0.01$	18	19	18	—	—	19

Table B.24: Comparison of block preconditioners for the time-dependent BFS-3D problem on the mesh M2 with $\Delta t = 0.01$, various viscosity values.

B.3.6 Turbine blade 2D

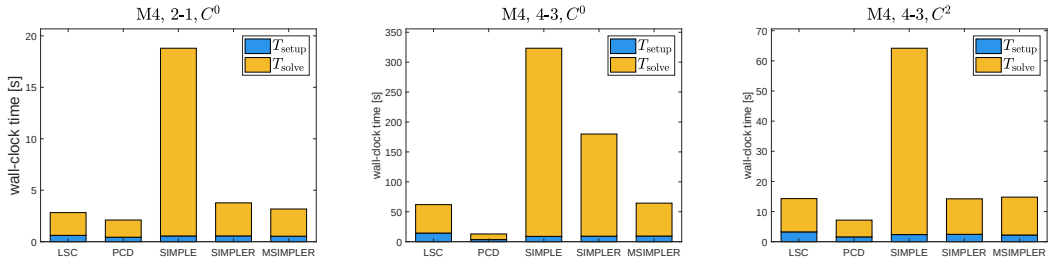


Figure B.12: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the steady-state TB-2D problem with $\nu = 0.01$ on the mesh M4.

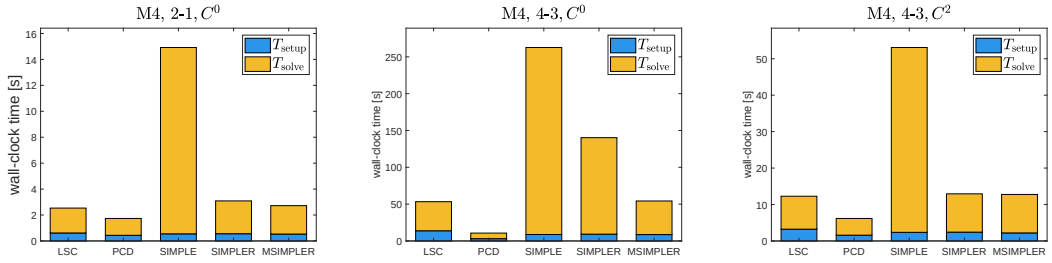
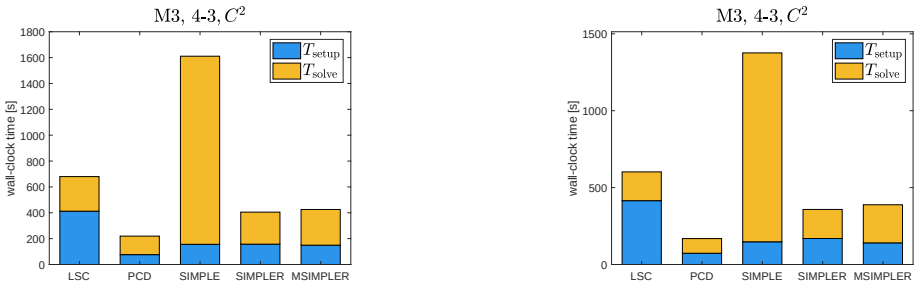


Figure B.13: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for selected discretizations of the time-dependent TB-2D problem with $\nu = 0.01$ on the mesh M4.

B.3.7 Turbine blade 3D



(a) Steady-state TB-3D.

(b) Time-dependent TB-3D.

Figure B.14: Wall-clock time of the preconditioner setup (T_{setup}) and the GMRES iterations (T_{solve}) for various preconditioners for the TB-3D problem with $\nu = 0.1$ on the mesh M3.

Vyjádření řešitele projektů

Ing. Hana Horníková se podílí na řešení výzkumných projektů od roku 2015, a to nejprve na projektu s názvem "MOTOR – Multi-ObjecTive design Optimization of fluid eneRgy machines" (9/2015-8/2018) financovaného Evropskou komisí v rámci programu Horizon 2020, následně na projektu s názvem "Moderní geometricko-numerické metody v simulaci nestlačitelného turbulentního proudění pro reálné úlohy velkého rozsahu" (1/2019-6/2022) financovaného GA ČR a v současné době na projektu s názvem "Moderní metody pro tvarovou optimalizaci Francisových turbín" (1/2022-12/2025) financovaného TA ČR. V rámci všech výše zmíněných projektů bylo jejím hlavním odborným zaměřením a výzkumných úkolem numerické řešení velkých soustav lineárních rovnic, vznikajících při numerickém řešení simulace turbulentního proudění ve 2D a 3D. Hlavní důraz byl kladen na studium iteračních řešičů v kombinaci s vhodným předpodmíněním pro diskretizaci pomocí tzv. isogeometrické analýzy. Významně se také podílela na implementaci numerického řešiče simulace turbulentního proudění ve 2D a 3D pro C++ knihovnu G+Smo.

Z výše uvedeného vyplývá, že obsah předkládané disertační práce představuje výsledky samostatné výzkumné činnosti Ing. Hany Horníkové, provedené v rámci výše uvedených vědeckých projektů. Všechny tyto projekty byly řešeny stejným výzkumným týmem šesti pracovníků, přičemž všichni pracovníci měli přibližně stejný úvazek. Podíl Ing. Hany Horníkové je tedy 17% na celkových výsledcích projektů.

doc. Ing. Bohumír Bastl, Ph.D.