Preconditioning for Linear Systems Arising from IgA Discretized Incompressible Navier-Stokes Equations

Horníková, Hana; Vuik, Cornelis

DOI
10.1007/978-3-030-49836-8_5

Publication date
2021

Document Version
Final published version

Published in
Isogeometric Analysis and Applications 2018

Citation (APA)
Horníková, H., & Vuik, C. (2021). Preconditioning for Linear Systems Arising from IgA Discretized Incompressible Navier-Stokes Equations. In H. van Brummelen, C. Vuik, M. Möller, C. Verhoosel, & B. Simeon (Eds.), *Isogeometric Analysis and Applications 2018* (pp. 77-97). (Lecture Notes in Computational Science and Engineering book series ; Vol. 133). Springer. https://doi.org/10.1007/978-3-030-49836-8_5

Preconditioning for Linear Systems Arising from IgA Discretized Incompressible Navier–Stokes Equations

Hana Horníková and Cornelis Vuik

Abstract We deal with efficient techniques for numerical simulation of the incompressible fluid flow based on the Navier–Stokes equations discretized using the isogeometric analysis approach. Typically, the most time-consuming part of the simulation is solving the large saddle-point type linear systems arising from the discretization. These systems can be efficiently solved by Krylov subspace methods, but the choice of the preconditioner is crucial.

In our study we test several preconditioners developed for the incompressible Navier–Stokes equations discretized by a finite element method, which can be found in the literature. We study their efficiency for the linear systems arising from the IgA discretization, where the matrix is usually less sparse compared to those from finite elements.

Our aim is to develop a fast solver for a specific problem of flow in a water turbine. It brings several complications like periodic boundary conditions at nonparallel boundaries and computation in a rotating frame of reference. This makes the system matrix even less sparse with a more complicated sparsity pattern.

1 Introduction

This work is motivated by numerical simulation of the incompressible fluid flow modeled by the Navier–Stokes/RANS equations with the aim of automatic shape optimization of runner blades of a water turbine. The governing equations are discretized using the isogeometric analysis (IgA) approach. The IgA approach has many common features with the finite element analysis (FEA). The main difference

H. Horníková (✉)
Faculty of Applied Sciences, University of West Bohemia, Plzeň, Czech Republic
e-mail: hhornik@kma.zcu.cz

C. Vuik
Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands
e-mail: c.vuik@tudelft.nl

H. van Brummelen et al. (eds.), *Isogeometric Analysis and Applications 2018*,
Lecture Notes in Computational Science and Engineering 133,
https://doi.org/10.1007/978-3-030-49836-8_5

is higher smoothness of the solution yielding higher accuracy per degree of freedom than standard finite elements with basis functions of the same order. Another advantage, which is important in the context of incompressible flows, is that it is possible to construct divergence conforming discretization spaces for complex domains using the isogeometric generalization of Raviart-Thomas elements [7, 8]. IgA is also suitable for the purpose of shape optimization, because it allows us to represent the domain boundaries exactly. Similarly to FEA, the IgA discretization of the linearized Navier–Stokes/RANS equations using an LBB stable pair of solution spaces leads to a sequence of sparse saddle-point type linear systems.

The solution of these linear systems represents the main bottleneck of the simulation process. The exact solution is unrealizable for large real world problems, because of very high time and memory requirements of the direct solvers. A promising approach to the iterative solution of these systems is the combination of Krylov subspace methods for nonsymmetric matrices with so-called block triangular preconditioners or SIMPLE-type preconditioners, both based on splitting the system into a velocity and a pressure part. As examples of block triangular preconditioners, we name the pressure convection-diffusion preconditioner (PCD) proposed by Kay, Loghin, Wathen in [12], the least-squares commutator (LSC) preconditioner by Elman et al. [4, 6] or the augmented Lagrangian preconditioner by Benzi and Olshanskii [1]. For an overview of the SIMPLE-type preconditioners, see e.g. [18].

In this paper, we present results of some numerical experiments for GMRES with these preconditioners applied to several linear systems arising from IgA discretization of the incompressible Navier–Stokes equations. We observe some of their properties like dependence of the convergence on the mesh refinement or the Reynolds number for a classical benchmark problem of flow in a 2D backward facing step domain. We also test their performance for a simple 2D problem with periodic boundary conditions on nonparallel sides and a mesh refined locally along these sides, which mimics some of the typical aspects of computations in the water turbine, since the turbine domain is radially symmetric and we usually need to refine the mesh near the blades. Although these 2D domains are very simple, they are intentionally described using higher degree B-splines which is also typical for the turbine geometries.

The structure of this paper is as follows. In Sect. 2 the discretization of the unsteady incompressible Navier–Stokes equations is given and the structure of the matrices obtained from the discretization of the problem in the water turbine is described in more detail. Section 3 gives a brief overview of selected preconditioners. In Sect. 4 we present the results of the numerical experiments and we conclude with Sect. 5.

2 Problem Formulation

The mathematical simulation of incompressible viscous Newtonian flow is based on the incompressible Navier–Stokes equations (NSE). Let $\Omega \subset \mathbb{R}^d$ be a bounded domain, d being the number of spatial dimensions, with the boundary $\partial\Omega$ consisting of two complementary parts, Dirichlet $\partial\Omega_D$ and Neumann $\partial\Omega_N$, and $T > 0$ is an upper bound of the time interval of interest $[0, T]$. The initial boundary value incompressible Navier–Stokes problem is given as a system of $d + 1$ equations together with initial conditions and mixed boundary conditions

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p &= \mathbf{0} && \text{in } \Omega \times [0, T], \\ \nabla \cdot \mathbf{u} &= 0 && \text{in } \Omega \times [0, T], \\ \mathbf{u}(\mathbf{x}, 0) &= \mathbf{u}_0(\mathbf{x}) && \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g}_D && \text{on } \partial\Omega_D, \\ \nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - \mathbf{n}p &= \mathbf{0} && \text{on } \partial\Omega_N, \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the flow velocity, p is the kinematic pressure, ν is the kinematic viscosity and $\mathbf{u_0}$, $\mathbf{g}_D$ are given functions. Note that the condition $\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - \mathbf{n}p = \mathbf{0}$ represents the classical "do-nothing" boundary condition resulting from the weak formulation of the momentum equations in (1). It does not have a physical meaning, but it is suitable at artificial outflow boundaries when modeling flows through a truncated domain, assuming that the physical domain continues further (see e.g. [9, 11] where different outflow boundary conditions for such problems are discussed).

2.1 Discretization and Linearization

The isogeometric analysis approach is based on the Galerkin method. One of the approaches to the discretization of time-dependent problems is to discretize the time derivative using finite differences first, arriving to a set of spatial problems that can be discretized using the Galerkin method. Using a backward finite difference with time step Δt, we obtain the following set of equations

$$\begin{aligned} \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + (\mathbf{u}^{n+1} \cdot \nabla)\mathbf{u}^{n+1} - \nu\Delta\mathbf{u}^{n+1} + \nabla p^{n+1} &= \mathbf{0} && \text{in } \Omega, \\ \nabla \cdot \mathbf{u}^{n+1} &= 0 && \text{in } \Omega. \end{aligned} \tag{2}$$

The weak formulation is as follows: find $\mathbf{u}^{n+1} \in V$ and $p^{n+1} \in L_2(\Omega)$ such that

$$\begin{aligned} \frac{1}{\Delta t}\int_\Omega \mathbf{u}^{n+1}\cdot\mathbf{v} + \nu\int_\Omega \nabla\mathbf{u}^{n+1} : \nabla\mathbf{v} + \int_\Omega (\mathbf{u}^{n+1}\cdot\nabla\mathbf{u}^{n+1})\cdot\mathbf{v} \\ -\int_\Omega p^{n+1}\nabla\cdot\mathbf{v} = \frac{1}{\Delta t}\int_\Omega \mathbf{u}^{n}\cdot\mathbf{v}, \\ \int_\Omega q\nabla\cdot\mathbf{u}^{n+1} = 0, \end{aligned} \tag{3}$$

for all $\mathbf{v} \in V_0$ and $q \in L_2(\Omega)$, where

$$\begin{aligned} V &= \{\mathbf{u} \in H^1(\Omega)^d \,|\, \mathbf{u} = \mathbf{g}_D \text{ on } \partial\Omega_D\}, \\ V_0 &= \{\mathbf{v} \in H^1(\Omega)^d \,|\, \mathbf{v} = \mathbf{0} \text{ on } \partial\Omega_D\}. \end{aligned} \tag{4}$$

After discretization using a stable pair of discrete solution spaces (e.g. the isogeometric generalization of Taylor–Hood, Nédélec or Raviart-Thomas elements [3]) and linearization of the convective term, we get a sequence of saddle-point type linear systems of the form

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{5}$$

where $F \in \mathbb{R}^{d\cdot n_u \times d\cdot n_u}$ is block diagonal (in case of Picard linearization, which is used in this paper) with the diagonal blocks containing the discretization of the convection-diffusion operator and the term coming from the discretized time derivative on the left-hand side of (3). The matrices $B^T \in \mathbb{R}^{d\cdot n_u \times n_p}$ and $B \in \mathbb{R}^{n_p \times d\cdot n_u}$ are discrete gradient and negative divergence operators, respectively, and n_u, n_p denote the number of velocity and pressure unknowns. The right-hand side of (5) contains the remaining part of the discretized time derivative and the eliminated Dirichlet boundary conditions.

2.2 Motivational Problem

As mentioned above, this work is motivated by flow simulation in water turbines. It involves two phenomena, which influence the matrix structure of the resulting linear system: periodic boundary conditions and a rotating frame of reference in the runner wheel.

In Fig. 1 we show an example of a Kaplan turbine geometry (left picture) with the computational domains for the stationary and rotating part (right top and

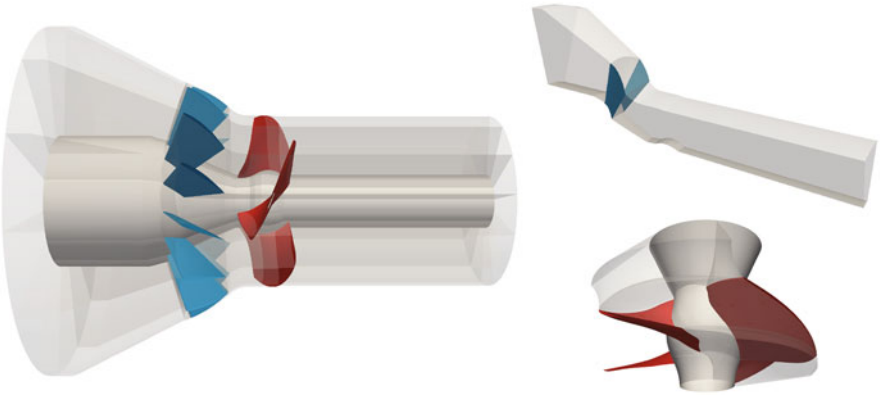

Fig. 1 The Kaplan turbine geometry (left) and particular computational domains for stationary part (right top) and rotating part (right bottom)

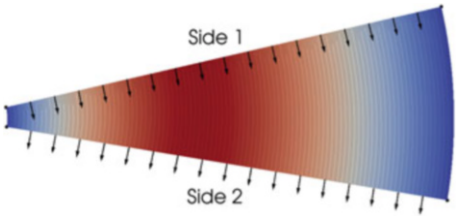

Fig. 2 Velocity field in a cross-section of a periodic 3D domain

bottom picture). Since both stationary and rotating part are radially periodic, we can define each computational domain as a strip between two vanes/blades with periodic conditions.

Since the periodic sides are not parallel and velocity is a vector quantity, we cannot simply identify the degrees of freedom on both sides. In Fig. 2 we display a cross-section of the stationary domain with velocity direction vectors along the periodic sides. It can be seen that instead of just copying the vector from side 1 to side 2, we have to rotate it first.

Without loss of generality, assume that the axis of the turbine is identified with the x-axis. Then the relation between corresponding velocity vectors is

$$\begin{bmatrix} u_1^1 \\ u_2^1 \\ u_3^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_3^2 \end{bmatrix}, \tag{6}$$

where u_i^1, $i = 1, 2, 3$, are velocity components on side 1 and u_i^2, $i = 1, 2, 3$, are velocity components on side 2, $\varphi = 2\pi/n_b$ and n_b is the number of blades/vanes.

The application of the periodic conditions can be done by discretization of the given problem with no boundary conditions at the periodic sides, arriving to a linear system with the structure

$$\begin{bmatrix} A & 0 & 0 & B_1^T \\ 0 & A & 0 & B_2^T \\ 0 & 0 & A & B_3^T \\ B_1 & B_2 & B_3 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ p \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ g \end{bmatrix}, \tag{7}$$

and adding multiples of the rows and columns corresponding to the periodic side 2 to the rows and columns corresponding to the periodic side 1 using the transformation (6) and deleting them from the system. This leads to the modified system

$$\begin{bmatrix} \tilde{A} & 0 & 0 & \tilde{B}_1^T \\ 0 & \hat{A} & -C & \tilde{B}_2^T \\ 0 & C & \hat{A} & \tilde{B}_3^T \\ \tilde{B}_1 & \tilde{B}_2 & \tilde{B}_3 & 0 \end{bmatrix} \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{p} \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \tilde{f}_3 \\ \tilde{g} \end{bmatrix}, \tag{8}$$

which is smaller than the original system by the number of degrees of freedom (DOFs) on side 2. After solving the system, the solution coefficients corresponding to side 2 can be obtained by rotating the coefficients corresponding to side 1 back by angle $-\varphi$.

Another nonzero off-diagonal blocks are added to the system matrix in the runner wheel domain, i.e. considering the rotating frame of reference. However, since this paper only presents results of some of the first numerical experiments on simpler domains which do not involve rotation, we do not go into details here.

3 Solution Methods

The linear systems resulting from the discretization can be solved either directly or iteratively. Direct solvers are applicable only for relatively small systems because of their very high time and memory requirements. In practice, we usually deal with very large systems, therefore an efficient iterative method is needed. Among iterative methods, Krylov subspace methods are the most commonly used in applications and can be very efficient if combined with a good preconditioning technique. Since our matrices are nonsymmetric, we have to use a Krylov subspace method for nonsymmetric matrices. The most popular ones are GMRES (generalized minimum residual) and BiCGSTAB (biconjugate gradient stabilized).

A good preconditioner for a Krylov subspace method should be such that the preconditioned matrix has a low degree minimal polynomial, which implies a low maximal dimension of the generated Krylov subspace. In other words, it is

desirable that the preconditioned matrix has only a few distinct eigenvalues and is diagonalizable or at least its Jordan canonical form has only small Jordan blocks.

3.1 Block Triangular Preconditioners

A popular class of preconditioners for the saddle-point type problems are the block triangular preconditioners based on splitting the system into a velocity and a pressure part, developed for finite element discretizations of the Navier–Stokes equations. An overview of these preconditioners can be found e.g. in [17]. Their construction is based on the block LDU decomposition of the system matrix in (5)

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{9}$$

where $S = -BF^{-1}B^T$ is the Schur complement matrix. This suggests the following choice of the preconditioner matrix

$$P = \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix}. \tag{10}$$

Then the right preconditioned matrix has obviously all eigenvalues equal to one, since it is a lower triangular matrix with all ones on its main diagonal. As shown in [15], the minimal polynomial of the preconditioned matrix is of degree 2 and therefore GMRES converges in at most 2 iterations. The same holds for left preconditioning.

The computation of $P^{-1}r$ is performed by solving the linear system

$$\begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \begin{bmatrix} z_u \\ z_p \end{bmatrix} = \begin{bmatrix} r_u \\ r_p \end{bmatrix} \tag{11}$$

in the steps summarized in Algorithm 1.

Algorithm 1 Application of P^{-1}

1: Solve $Sz_p = r_p$
2: Update $r_u = r_u - B^T z_p$
3: Solve $Fz_u = r_u$

In practice, the system with F is usually solved approximately, e.g. by a small number of iterations of some iterative method or one or more V-cycles of a multigrid solver for convection-diffusion equations. The solution of the system with the Schur complement is not that straightforward. We do not construct S explicitly, because it would require the explicit construction of F^{-1}, since it is multiplied with rectangular matrices from both sides. Furthermore, it is a dense matrix. Therefore we have to find some inexpensive approximation $\hat{S} \approx S$ first. The choice of the approximation yields different preconditioners.

3.1.1 Least-Squares Commutator Preconditioner

The least-squares commutator (LSC) preconditioner proposed by Elman [4, 5] is based on the idea of approximate commutators similar to the pressure convection-diffusion (PCD) preconditioner [12]. The general algebraic idea of these methods is to find a matrix X for which

$$B^T X \approx F B^T \tag{12}$$

and consequently

$$S = -BF^{-1}B^T \approx -BB^T X^{-1}, \tag{13}$$

i.e., the inverse is moved so that it is no longer between the two rectangular matrices.

The construction of these preconditioners is based on the fact that F is a discretization of the operator

$$\mathcal{L} = \frac{1}{\Delta t} - \nu\Delta + (\mathbf{w} \cdot \nabla) \tag{14}$$

defined on the discrete velocity space, where $\mathbf{w}$ is the approximation of the velocity computed in the previous Picard iteration. Suppose that an analogous operator is well defined also on the discrete pressure space and denote it by $\mathcal{L}_p$. It can be expected that the commutator with the gradient operator

$$\mathcal{E} = \mathcal{L}\nabla - \nabla\mathcal{L}_p \tag{15}$$

is small for smooth $\mathbf{w}$ [5]. The discrete version of the commutator in terms of finite element matrices takes the form

$$E = (M_u^{-1}F)(M_u^{-1}B^T) - (M_u^{-1}B^T)(M_p^{-1}F_p), \tag{16}$$

where M_u and M_p are the velocity and pressure mass matrix, respectively, and F_p is a discrete version of $\mathcal{L}_p$. Assume that E is also small, which means that

$$(M_u^{-1}F)(M_u^{-1}B^T) \approx (M_u^{-1}B^T)(M_p^{-1}F_p). \tag{17}$$

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

$$S = -BF^{-1}B^T \approx -BM_u^{-1}B^T F_p^{-1}M_p. \tag{18}$$

The LSC preconditioner avoids the explicit construction of F_p (unlike PCD) and defines the j-th column of F_p as a solution of the following weighted least-squares problem

$$\min || [M_u^{-1}FM_u^{-1}B^T]_j - M_u^{-1}B^T M_p^{-1}[F_p]_j ||_{M_u}, \tag{19}$$

where $||x||_{M_u} = \sqrt{x^T M_u x}$ is a discrete analogue of the continuous L_2 norm on the velocity space. The vector $[F_p]_j$ is obtained by solving the normal equations

$$M_p^{-1}BM_u^{-1}B^T M_p^{-1}[F_p]_j = [M_p^{-1}BM_u^{-1}FM_u^{-1}B^T]_j, \tag{20}$$

which leads to the following definition of F_p:

$$F_p = M_p(BM_u^{-1}B^T)^{-1}(BM_u^{-1}FM_u^{-1}B^T). \tag{21}$$

Substituting this into (18) we get the following approximation of the Schur complement

$$\hat{S}_{LSC} = -(BM_u^{-1}B^T)(BM_u^{-1}FM_u^{-1}B^T)^{-1}(BM_u^{-1}B^T). \tag{22}$$

Since the inverse of the velocity mass matrix M_u^{-1} is dense, we replace M_u by its diagonal $\hat{M}_u$.

In Algorithm 2 we can see the individual steps of the LSC preconditioner application. It involves solving two subsystems with the matrix $A_L = B\hat{M}_u^{-1}B^T$, which is essentially a discrete Laplace operator. Thus, two Poisson-type solves for pressure and one velocity solve are needed.

Algorithm 2 LSC preconditioner

1: Solve $A_L z_p = r_p$, where $A_L = B\hat{M}_u^{-1}B^T$
2: Update $r_p = (B\hat{M}_u^{-1}F\hat{M}_u^{-1}B^T)z_p$
3: Solve $A_L z_p = -r_p$
4: Update $r_u = r_u - B^T z_p$
5: Solve $Fz_u = r_u$

3.1.2 Augmented Lagrangian Preconditioner

A different approach has been proposed by Benzi and Olshanskii [1]. The original system (5) is replaced with the equivalent system

$$\begin{bmatrix} F_\gamma & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f_\gamma \\ g \end{bmatrix}, \tag{23}$$

where $F_\gamma = F + \gamma B^T W^{-1} B$, $f_\gamma = f + \gamma B^T W^{-1} g$, $\gamma > 0$ is a parameter and W is a positive definite matrix. The system (23) is then preconditioned with the block triangular preconditioner

$$P_{AL} = \begin{bmatrix} F_\gamma & B^T \\ 0 & \hat{S}_{AL} \end{bmatrix}, \tag{24}$$

where the inverse of the Schur complement approximation is given by

$$\hat{S}_{AL}^{-1} := -\nu \tilde{M}_p^{-1} - \gamma W^{-1} \tag{25}$$

and $\tilde{M}_p$ is a pressure mass matrix approximation, usually a diagonal matrix. The matrix W is often chosen to be equal to $\tilde{M}_p$.

Of course, the choice of the parameter γ is important. A large value would lead to small number of iterations of the preconditioned Krylov method, but for large γ the block F_γ becomes increasingly ill-conditioned and makes the solution of the subsystems expensive [17]. Hence, it is often set $\gamma \approx 1$.

The main difficulty of this approach is the choice of the approximate solver for the subsystems with F_γ. The additional term $\gamma B^T W^{-1} B$ makes the matrix less sparse compared to F and introduces a coupling between the velocity components which is not present in the discretization of the Picard linearized Navier–Stokes equations (without rotation or the periodic conditions mentioned in Sect. 2.2). The authors in [1] develop a multigrid method suitable for these subsystems.

Modified Version

One way to simplify the solution of the systems with F_γ is the modified version of AL preconditioner introduced in [2]. Let us denote the particular blocks of F_γ in two dimensions as follows

$$F_\gamma = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{26}$$

The modified approach suggests to replace this block by its upper block triangle

$$\tilde{F}_\gamma =: \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \tag{27}$$

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

3.2 SIMPLE-Type Preconditioners

SIMPLE (Semi-Implicit Method for Pressure Linked Equations) is an algorithm for numerical solution of the Navier–Stokes equations developed for finite volume and finite difference discretizations by Patankar and Spalding [16]. It is based on decoupling the system of equations and solving the velocity and pressure part separately. First, the velocity is solved from the momentum equations assuming that the pressure is known from the previous iteration. Then, the pressure and velocity are corrected in order to satisfy the discrete continuity equation.

This algorithm can be written in the form of block matrices and preconditioners for the systems arising from finite element discretizations can be derived. An overview of these preconditioners is given in [18]. The derivation is based on the block LU decomposition of the coefficient matrix in (5) and rewriting the system as

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{28}$$

The SIMPLE algorithm is obtained by using the approximation $F^{-1} \approx D^{-1} = \mathrm{diag}(F)^{-1}$, introducing intermediate values u^*, p^* and corrections δu, δp such that

$$u = u^* + \delta u, \quad p = p^* + \delta p, \tag{29}$$

and solving the system in the following two steps:

$$\begin{bmatrix} F & 0 \\ B & \hat{S}_S \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{30}$$

and

$$\begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} u^* \\ \delta p \end{bmatrix}, \tag{31}$$

where $\hat{S}_S = -BD^{-1}B^T$. These steps are performed recursively, leading to the Algorithm 3, where the intermediate pressure p^* is estimated from the prior iterations. One iteration of the SIMPLE algorithm is used as preconditioner with $p^* = 0$.

Algorithm 3 SIMPLE algorithm

1: Solve $Fu^* = f - B^T p^*$
2: Solve $\hat{S}_S \delta p = g - Bu^*$
3: Update $u = u^* + \delta u$, where $\delta u = -D^{-1}B^T \delta p$
4: Update $p = p^* + \delta p$

There are several modifications of the algorithm. One of them is called SIMPLER, where p^* is obtained as a solution of the system

$$\hat{S}_S p^* = g - BD^{-1}((D - F)u^k + f), \tag{32}$$

where u^k is the velocity from the previous iteration in the original algorithm, but in case of preconditioner it is taken equal to zero.

Another variant is MSIMPLER algorithm, which is obtained from SIMPLER by replacing all occurrences of D by an approximation of the velocity mass matrix $\hat{M}_u$. The choice of $\hat{M}_u$ depends on the particular type of elements. For more details on this preconditioner, see [18].

4 Numerical Experiments

In this section we present results of some numerical experiments. The linear systems used in the experiments are obtained from an in-house isogeometric incompressible flow solver implemented in C++ within a framework of the G+Smo[1] library. G+Smo is an open-source object-oriented template C++ library, that implements the concept of IgA, based on abstract classes for geometry, discretization basis, assemblers, solvers etc. For more information about the library, see the documentation [14]. The linear algebra tools available in G+Smo are mostly inherited from the Eigen library [10].

[1] http://github.com/gismo, http://gismo.github.io.

For the discretization we use the isogeometric Taylor–Hood element, which means that the pressure basis is taken from the geometry and the velocity basis is obtained by p-refinement (degree elevation) of the pressure basis.

We implemented the selected preconditioners, namely LSC, AL, modified AL (MAL), SIMPLE, SIMPLER and MSIMPLER, also in the framework of the G+Smo library. For now, a direct solver (sparse LU decomposition from Eigen) is used for solving all subsystems in the preconditioners.

The numerical experiments were performed using a machine with the following parameters: Windows Server 2012, 2× Intel Xeon CP E5-2690 v2 @ 3.00GHz, 256 GB RAM.

In the experiments, we use full GMRES with various preconditioners to solve one linear system obtained after performing several Picard iterations in the steady case or several time steps in the unsteady case and we track the relative residual norm $||r||_2/||b||_2$, where b denotes the right-hand side, and the solution time in seconds.

4.1 Backward Step 2D

As a simple test example, we consider a 2D backward facing step domain consisting of three B-spline patches with conforming mesh, where the degrees of freedom on the interfaces are identified. The individual patches are described as B-splines of degree 3.

For comparison, we consider three meshes with different level of uniform refinement with 11,229, 42,005 and 162,309 degrees of freedom (DOFs). The coarsest mesh is shown in Fig. 3. The step height h as well as the inlet height is equal to 1. We prescribe a parabolic velocity profile with the maximum of 1 at the inlet boundary, zero velocity on the walls (upper and lower boundaries) and the "do-nothing" boundary condition at the outlet. For unsteady computations, the initial conditions are taken as the solution of the corresponding steady Stokes problem. The Reynolds number $Re = \frac{1}{\nu}$, where ν is the kinematic viscosity.

In Table 1 we present the number of iterations and the computational time in seconds (in parentheses) needed to reach the relative residual norm smaller than 10^{-10} for the individual preconditioners in the steady (top) and unsteady (bottom) case. The computational time is reported in two separate parts, the time of the preconditioner setup involving the factorization of the subsystem matrices and the

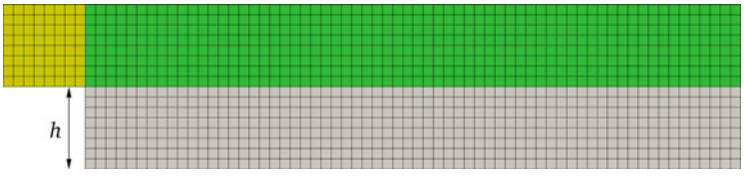

Fig. 3 The computational mesh with 11,229 DOFs

Table 1 Number of iterations and computational time in seconds (in parentheses) needed to fulfill $||r||_2/||b||_2 < 10^{-10}$, backward step steady (top) and unsteady (bottom) case for $Re = 100$

Steady	Mesh 1	Mesh 2	Mesh 3
LSC	32 (0.60 + 0.55)	40 (4.34 + 3.55)	55 (32.8 + 22.5)
SIMPLE	171 (0.47 + 3.19)	> 200	> 200
SIMPLER	28 (**0.49 + 0.51**)	28 (**3.87 + 2.59**)	37 (**30.4 + 15.02**)
MSIMPLER	32 (0.48 + 0.58)	40 (3.87 + 3.70)	54 (30.4 + 22.2)
AL	15 (10.8 + 1.01)	10 (232 + 5.70)	8 (4592 + 33.8)
MAL	> 200	> 200	> 200
Unsteady	Mesh 1	Mesh 2	Mesh 3
LSC	11 (0.70 + 0.22)	11 (4.95 + 1.12)	11 (33.1 + 4.71)
SIMPLE	34 (0.57 + 0.62)	29 (4.42 + 2.53)	22 (31.0 + 7.67)
SIMPLER	12 (**0.59 + 0.26**)	11 (4.47 + 1.16)	11 (**31.5 + 4.91**)
MSIMPLER	12 (**0.59 + 0.26**)	11 (**4.46 + 1.16**)	12 (31.6 + 5.33)
AL	47 (10.6 + 3.02)	47 (215 + 22.2)	47 (4063 + 168)
MAL	49 (3.46 + 2.07)	51 (80.4 + 14.6)	62 (1482 + 130)

time of the subsequent GMRES iterations. The lowest total time for a given problem is displayed in bold. The value $\gamma = 1$ was used for the AL and MAL preconditioners in these experiments and 200 was the maximum number of iterations. In the steady case, the number of iterations for LSC and SIMPLE-type preconditioners increases for finer meshes. On the contrary, the number of iterations for AL decreases, but this preconditioner is obviously very expensive with the direct solver for the subsystems, hence it is really necessary to use some efficient iterative method to solve them. The convergence of the modified AL is very slow, probably because $\gamma = 1$ is far from the optimal value of γ for MAL in this case. In the unsteady case, the convergence is generally faster than in the steady case for most preconditioners and most of them (except MAL) seem independent of the mesh size.

We note that in real applications, especially three-dimensional problems, the subsystems cannot be solved by direct solution methods. We refer to [13] and [18] where comparable problems are solved (using finite volume and finite element method). It appears that the solution of the subsystems can be approximated by 1 or 2 iterations of a MG V-cycle, or a solution with PCG or preconditioned Bi-CGSTAB with a moderate accuracy (reduction of the relative residual by a factor 10 or 100). Since the preconditioner is used to find a suitable search direction for the outer Krylov method, it appears that it is not very sensitive to the accuracy of the inner solves. If MG can be used, the total solver becomes scalable.

In the next experiment, we consider a steady flow with $\nu = 0.01, 0.009, \ldots,$ 0.003 and an unsteady flow with $\nu = 10^{-2}, 10^{-3}, 10^{-4}$ in the backward step domain with mesh 2 to study the dependence on the Reynolds number. Here, the value of γ for MAL was chosen as the "optimal" value from the interval [0.1, 2.5] (found experimentally with step 0.1). Figure 4 shows the evolution of the relative residual norm during 100 iterations of GMRES for the individual preconditioners in the steady case. We can see that the convergence slows down after several first iterations

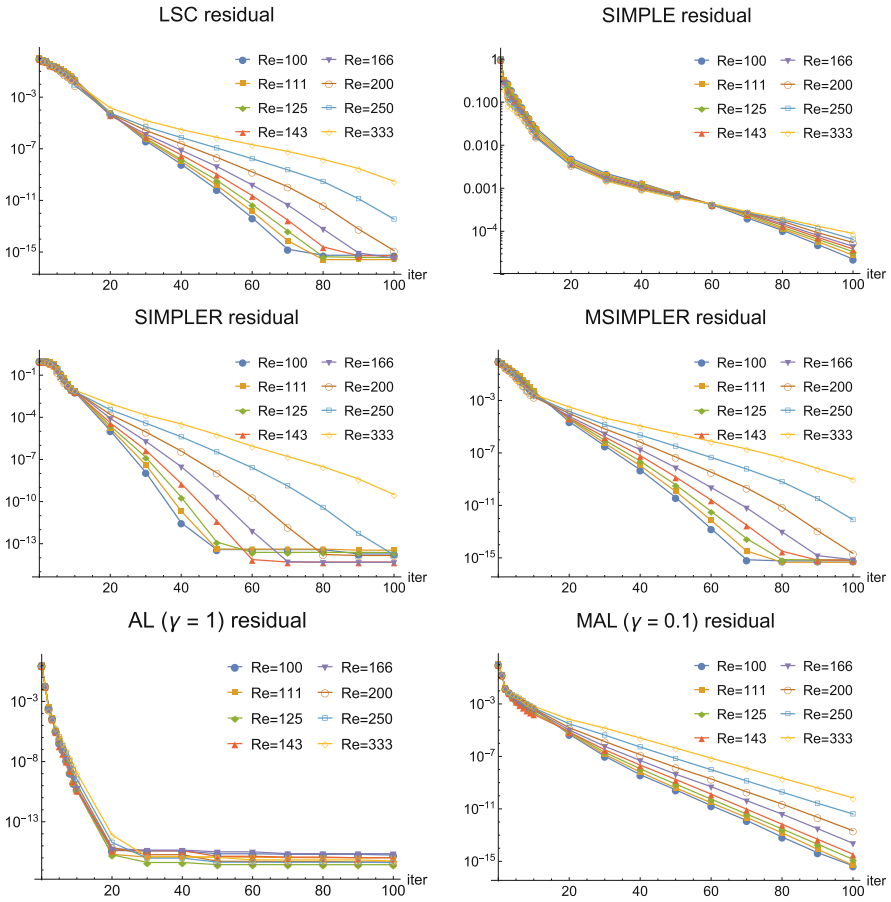

Fig. 4 Residual evolution for the individual preconditioners, a linear system from the steady Navier–Stokes with Reynolds number Re varying between 100 and 333, backward step with mesh 2

for increasing Reynolds number for all preconditioners except AL. A comparison of the preconditioners for the lowest and highest Reynolds numbers $Re = 100$ and $Re = 333$ is given in the left and right picture of Fig. 5, respectively. The convergence of AL is significantly faster than of all the other preconditioners.

Figure 6 shows the evolution of the relative residual norm in the unsteady case. The convergence of all preconditioners is almost independent of the Reynolds number, except for SIMPLE, for which it is slightly dependent. AL does not show superior convergence behavior anymore in the unsteady case.

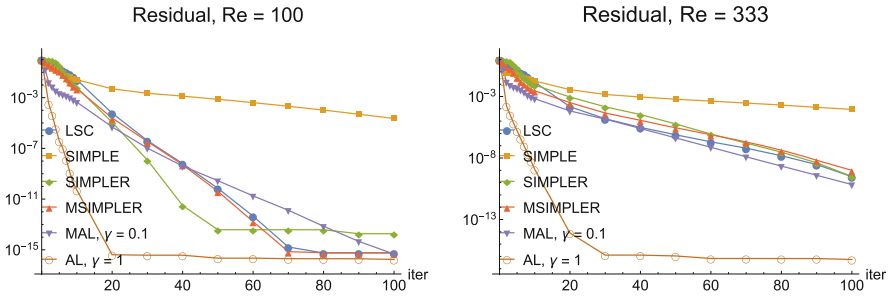

Fig. 5 Residual evolution for various preconditioners, a linear system from steady Navier–Stokes, backward step with mesh 2, $Re = 100$ (left) and $Re = 333$ (right)

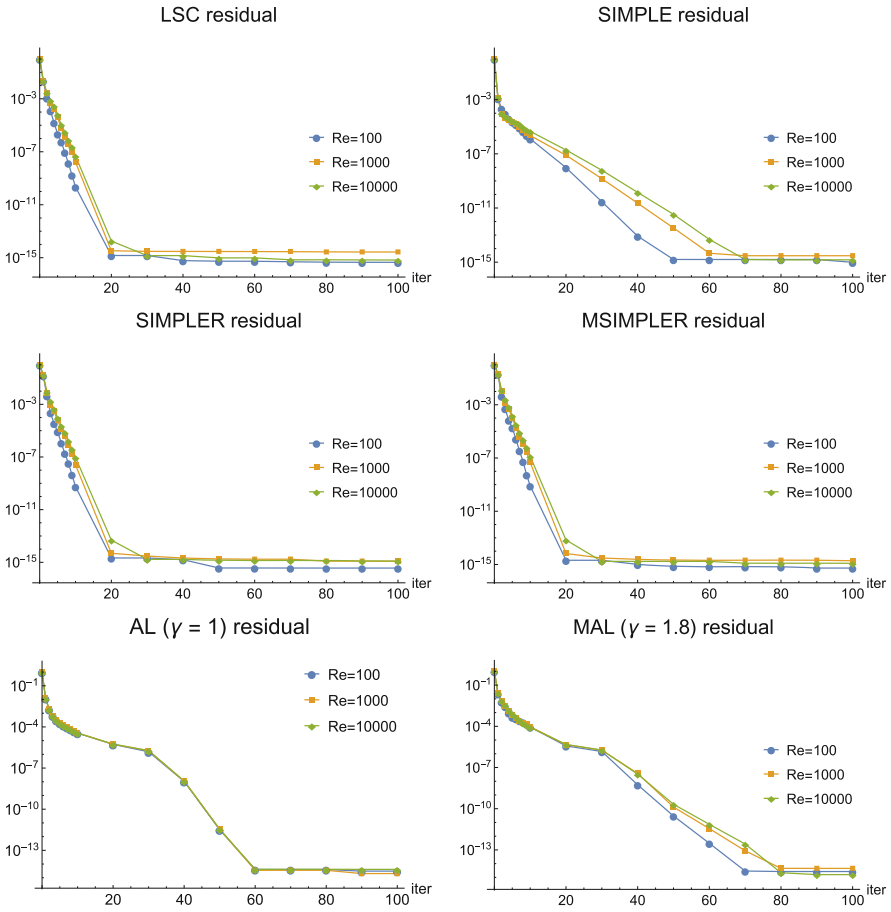

Fig. 6 Residual evolution for the individual preconditioners, a linear system from the unsteady Navier–Stokes with Reynolds number Re varying between 100 and 10000, backward step with mesh 2

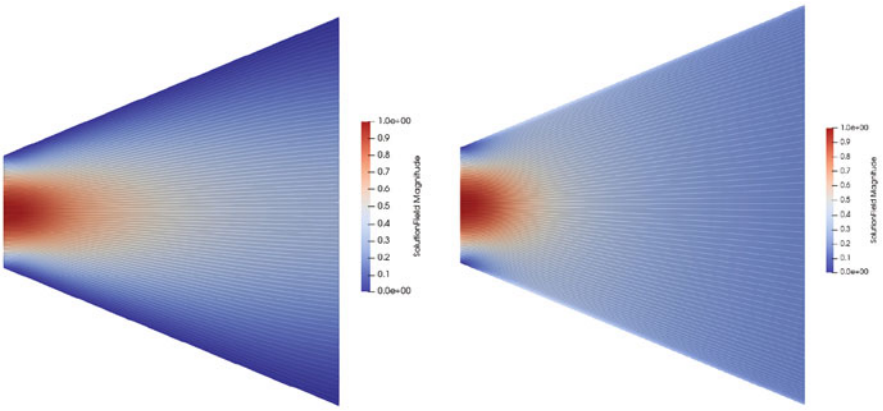

Fig. 7 Velocity with streamlines in a trapezoidal domain where the lower and upper boundaries are solid walls (left) or periodic sides (right)

4.2 Periodic Domain 2D

In order to test the effect of periodic boundary conditions on the convergence of the iterative solvers, consider a simple (artificial) test example in 2D – a flow in a trapezoidal domain shown in Fig. 7. The left and right boundary is the inflow and outflow boundary, respectively, and the lower and upper boundaries are either solid walls or periodic sides. The pictures display the velocity solution of an unsteady Navier-Stokes problem with viscosity $\nu = 10^{-2}$ after 20 time steps with $\Delta t = 10^{-2}$. The geometry is described as a B-spline of degree 3, hence the degree of the basis functions is 3 for pressure and 4 for velocity. The experiments were done for three meshes, a uniformly refined mesh with around 10,000 DOFs and two meshes with different levels of local refinement near the upper and lower boundary with around 11,000 and 12,000 DOFs. The maximum aspect ratio of the meshes is approximately 4, 16 and 64, respectively. Such local refinement is of interest, because in the real computations in the water turbine we simulate turbulent flow using the RANS equations with a turbulence model and therefore we need to refine near the vanes/blades (i.e. the periodic sides).

In the periodic case, where the block F is no longer block diagonal (see (8)), the subsystems with F cannot be split into several smaller subsystems corresponding to the velocity components. In the 2D example, the system structure is as in (8) without the first block row and column. For experimental purposes, we implemented several ways to solve these subsystems (exactly or approximately):

- `Fdiag`
  We neglect all off-diagonal blocks and split the system into d subsystems with the diagonal blocks.

- `Fmod`
  We replace F by its upper block triangle, similarly to the modified AL approach.
- `Fwhole`
  We solve the whole system $Fz_u = r_u$.

We do not apply these variants to the AL preconditioner, since it requires solution of systems with the matrix F_γ instead of F, which is already not block diagonal and the whole system has to be solved even without periodic conditions. We remind that MAL is a modification of AL replacing F_γ with its upper block triangle (in both periodic and nonperiodic case).

In Tables 2 and 3 we present the number of iterations and the computational time needed to reach the relative residual norm smaller than 10^{-10} for the individual preconditioners in the nonperiodic and periodic case, respectively. Again, $\gamma = 1$ was used for the AL and MAL preconditioners in these experiments.

It can be seen from Table 2 that the convergence of most of the preconditioners (except SIMPLER and AL) is dependent on the local refinement, i.e. the aspect ratio of the mesh, in the nonperiodic case.

In the periodic case (Table 3), we compare the approaches to the subsystems with block F described above. For the uniformly refined mesh, there are only small differences in the number of iterations for different approaches, hence, it seems to be sufficient to use `Fdiag` in this case. However, for the locally refined meshes the differences become more significant. Further, LSC, SIMPLER and MSIMPLER give similar numbers of iterations for all meshes with `Fwhole` (i.e. if the subsystem with F is solved exactly), but the dependence on the aspect ratio with the approximations of F is more significant for LSC than for SIMPLER and MSIMPLER. SIMPLER is no longer independent of the aspect ratio even with `Fwhole`. The AL preconditioner gives exactly the same number of iterations as in the nonperiodic case and stays independent of the aspect ratio, but it is very expensive. The modified version of AL needs approximately twice as many iterations and half the computational time compared to AL with the same γ in the case of uniform refinement, but its convergence is dependent on the aspect ratio and the dependence is much stronger in the periodic case. Note that $\gamma = 1$ is not an

Table 2 Number of iterations and computational time in seconds (in parentheses) needed to fulfill $||r||_2/||b||_2 < 10^{-10}$, nonperiodic trapezoidal domain

Precond.	Uniform	loc1	loc2
LSC	11 (0.47 + 0.14)	13 (0.56 + 0.20)	21 (0.66 + 0.37)
SIMPLE	27 (0.36 + 0.32)	28 (0.43 + 0.39)	34 (0.51 + 0.57)
SIMPLER	11 (**0.37 + 0.16**)	11 (**0.45 + 0.19**)	11 (**0.53 + 0.22**)
MSIMPLER	12 (0.37 + 0.17)	14 (0.45 + 0.23)	22 (0.53 + 0.43)
AL	22 (7.03 + 1.09)	22 (8.83 + 1.29)	22 (9.80 + 1.43)
MAL	41 (3.05 + 1.46)	53 (3.61 + 2.18)	63 (3.96 + 2.89)

Bold values indicate the lowest total computational time

Table 3 Number of iterations and computational time in seconds (in parentheses) needed to fulfill $||r||_2/||b||_2 < 10^{-10}$, periodic trapezoidal domain

Precond.	Uniform	loc1	loc2
LSC			
`Fdiag`	18 (0.73 + 0.26)	33 (0.95 + 0.57)	70 (1.20 + 1.48)
`Fmod`	15 (0.73 + 0.22)	24 (0.95 + 0.42)	50 (1.21 + 1.05)
`Fwhole`	11 (1.26 + 0.22)	13 (1.64 + 0.30)	17 (1.99 + 0.45)
SIMPLE			
`Fdiag`	31 (0.61 + 0.42)	38 (0.81 + 0.63)	61 (1.05 + 1.21)
`Fmod`	29 (0.61 + 0.41)	36 (0.81 + 0.61)	54 (1.05 + 1.09)
`Fwhole`	28 (1.13 + 0.51)	31 (1.50 + 0.67)	42 (1.84 + 1.05)
SIMPLER			
`Fdiag`	13 (0.62 + 0.20)	22 (0.82 + 0.41)	44 (1.06 + 0.97)
`Fmod`	12 (**0.62 + 0.19**)	17 (**0.83 + 0.32**)	34 (**1.06 + 0.76**)
`Fwhole`	11 (1.15 + 0.23)	16 (1.52 + 0.38)	26 (1.85 + 0.71)
MSIMPLER			
`Fdiag`	14 (0.62 + 0.22)	22 (0.82 + 0.41)	45 (1.06 + 0.99)
`Fmod`	13 (0.62 + 0.21)	18 (0.82 + 0.34)	35 (1.06 + 0.78)
`Fwhole`	12 (1.15 + 0.25)	13 (1.51 + 0.32)	19 (1.85 + 0.53)
AL	22 (6.78 + 1.29)	22 (9.04 + 1.43)	22 (12.0 + 1.68)
MAL	46 (2.16 + 1.70)	92 (2.97 + 4.03)	195 (4.08 + 10.6)

Bold values indicate the lowest total computational time

optimal value for MAL in this case. We can get a bit faster convergence for a better choice of γ, but the aspect ratio dependence is similar.

5 Conclusions

In this paper we studied convergence behavior of selected block triangular preconditioners (LSC, AL, MAL) and SIMPLE-type preconditioners (SIMPLE, SIMPLER, MSIMPLER) for linear systems obtained from IgA discretization of steady and unsteady incompressible Navier–Stokes equations. All subsystems in the preconditioners were solved with a direct solver.

We investigated the dependence on the mesh size and the Reynolds number for a simple 2D backward facing step example. The experiments show, that most of the preconditioners are dependent on both, the mesh size and the Reynolds number, in the steady case, but independent or almost independent in the unsteady case. Further, it seems that AL outperforms the other preconditioners in terms of number of iterations in the steady case, but it is much more expensive, at least if the subsystems are solved directly. The convergence of LSC and SIMPLE-type preconditioners is generally faster for the unsteady problem than for the steady problem. Such behavior is expectable due to the nature of the systems. The main block diagonal of the matrix

F in the linear system arising from the unsteady case contains the velocity mass matrix multiplied by $1/\Delta t$. Thus, the mass matrix becomes a dominant part of the system matrix, which makes the system easier to solve.

Then we considered a simple 2D domain to simulate some of the typical features in the water turbines - periodic boundary conditions on nonparallel sides and locally refined meshes along them. We tested several simplifications of the solution of the system with block F, which is no longer block diagonal in the periodic case. In general, all tested preconditioners except AL are dependent on the aspect ratio. According to the experiments, we can neglect the off-diagonal blocks of F in the periodic case for low aspect ratio, but it seems necessary to solve the system with the full block F for higher aspect ratios. This issue is a topic for further research.

Another interesting topic for future research is the dependence of the convergence on the order of continuity of the IgA solution. We also plan to focus on iterative solution of the subsystems in the preconditioners in the future.

Acknowledgments This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 678727 and by the project LO1506 of the Czech Ministry of Education, Youth and Sports under the program NPU I.

References

1. M. Benzi and M. A. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. *SIAM J. Sci. Comput.*, 28(6):2095–2113, 2006.
2. M. Benzi, M. A. Olshanskii, and Z. Wang. Modified augmented Lagrangian preconditioners for the incompressible Navier–Stokes equations. *Int. J. Numer. Meth. Fluids*, 66(4):486–508, 2011.
3. A. Buffa, C. de Falco, and G. Sangalli. Isogeometric analysis: stable elements for the 2D Stokes equation. *International Journal for Numerical Methods in Fluids*, 65(11–12):1407–1422, 2011.
4. H. Elman. Preconditioning for the steady-state Navier–Stokes equations with low viscosity. *SIAM J. Sci. Comput.*, 20(4):1299–1316, 1999.
5. H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. Block preconditioners based on approximate commutators. *SIAM J. Sci. Comput.*, 27(5):1651–1668, 2006.
6. 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(1):290–311, 2007.
7. J. A. Evans and T. J. R. Hughes. Isogeometric divergence-conforming B-splines for the steady Navier–Stokes equations. *Math. Models Methods Appl. Sci.*, 23(8):1421–1478, 2013.
8. J. A. Evans and T. J. R. Hughes. Isogeometric divergence-conforming B-splines for the unsteady Navier–Stokes equations. *J. Comput. Phys.*, 241:141–167, 2013.
9. J. Fouchet-Incaux. Artificial boundaries and formulations for the incompressible Navier–Stokes equations. Applications to air and blood flows. *SeMA Journal*, 64:1–40, 2014.
10. G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.
11. J. G. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier–Stokes equations. *Int. J. Numer. Meth. Fluids*, 22:325–352, 1996.
12. D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier–Stokes equations. *SIAM J. Sci. Comput.*, 24(1):237–256, 2002.

13. 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*, 71:830–849, 2013.
14. A. Mantzaflaris and others (see website). G+Smo (Geometry plus Simulation modules) v0.8.1. http://github.com/gismo, 2018.
15. M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.*, 21(6):1969–1972, 2000.
16. 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*, 15:1787–1805, 1972.
17. A. Segal, M. ur Rehman, and C. Vuik. Preconditioners for incompressible Navier–Stokes solvers. *Numer. Math. Theor. Meth. Appl.*, 3(3):245–275, 2010.
18. M. ur Rehman, C. Vuik, and G. Segal. SIMPLE-type preconditioners for the Oseen problem. *Int. J. Numer. Meth. Fluids*, 61(4):432–452, 2009.