An Assessment of Some Solvers for Saddle Point Problems Emerging from the Incompressible Navier–Stokes Equations

Naveed Ahmed[a], Clemens Bartsch[a], Volker John[a,b,*], Ulrich Wilbrandt[a]

[a] *Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstr. 39, 10117 Berlin, Germany*
[b] *Freie Universität Berlin, Department of Mathematics and Computer Science, Arnimallee 6, 14195 Berlin, Germany*

Abstract

Efficient incompressible flow simulations, using inf-sup stable pairs of finite element spaces, require the application of efficient solvers for the arising linear saddle point problems. This paper presents an assessment of different solvers: the sparse direct solver UMFPACK, the flexible GMRES (FGMRES) method with different coupled multigrid preconditioners, and FGMRES with Least Squares Commutator (LSC) preconditioners. The assessment is performed for steady-state and time-dependent flows around cylinders in 2d and 3d. Several pairs of inf-sup stable finite element spaces with second order velocity and first order pressure are used. It turns out that for the steady-state problems often FGMRES with an appropriate multigrid preconditioner was the most efficient method on finer grids. For the time-dependent problems, FGMRES with LSC preconditioners that use an inexact iterative solution of the velocity subproblem worked best for smaller time steps.

Keywords: linear saddle point problems, inf-sup stable pairs of finite element spaces, UMFPACK, flexible GMRES, coupled multigrid preconditioners with Vanka smoother, Least Squares Commutator preconditioners

1. Introduction

Inf-sup stable finite element methods are a popular approach for the spatial discretization of incompressible flow problems. Within a Picard or Newton-type

[*] Corresponding author

Email addresses: `ahmed@wias-berlin.de` (Naveed Ahmed), `bartsch@wias-berlin.de` (Clemens Bartsch), `john@wias-berlin.de` (Volker John), `wilbrandt@wias-berlin.de` (Ulrich Wilbrandt)

iteration, one has to solve in each step an algebraic linear saddle point problem

$$\mathcal{A}\underline{x} = \underline{b}, \quad \underline{x}, \underline{b} \in \mathbb{R}^n, \quad \mathcal{A} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}. \tag{1}$$

The time spent for solving systems of form (1) constitutes usually a large part, often even the dominant part, of the total simulation time. For this reason, efficient algebraic solvers are of great importance.

This paper considers several solvers for algebraic linear saddle point problems, in particular iterative methods with an outer Krylov subspace iteration. It is well known that the efficiency of such methods depends on an appropriately constructed preconditioner $\mathcal{P}$. The eigenvalues of the preconditioned system should be clustered since this property is favorable for Krylov subspace methods [14, p. 361]. Standard preconditioners, like the Jacobi method or SOR, cannot be applied for systems of type (1) because of the zero block in the diagonal of $\mathcal{A}$.

Concerning iterative solvers (or preconditioners) of linear saddle point problems, one can distinguish two classes. The class of coupled methods deals with the full matrix as given in (1). Alternatively, methods were designed that solve equations connected with the first and second row of blocks separately. In this paper, representatives from both classes are studied.

Coupled multigrid methods belong to the class of coupled preconditioners. Initially developed as solvers themselves, they have been proven to be more efficient when applied as left preconditioners in Krylov subspace methods for the solution of linear saddle point problems arising in the linearization and discretization of the Navier–Stokes equations, [22, 23]. Using a multigrid preconditioner, our experience is that the use of a flexible Krylov subspace method is of advantage for the robustness of the overall solver. Each application of the multigrid method might represent a (slightly) different preconditioner by construction, e.g., when applying an iterative scheme on the coarsest grid whose termination criterion is not a fixed number of iterations.

In recent years, a lot of effort has been spent to develop and improve approaches belonging to the second class of methods. Within this class, two principal approaches have been pursued, leading to Least Squares Commutator (LSC) preconditioners [12] and Augmented Lagrangian preconditioners [4]. Overviews are provided, e.g., in [11, 30, 35].

LSC preconditioners are right preconditioners. They are based on similar ideas as the previously developed Pressure Convection-Diffusion (PCD) preconditioner [27]. The LSC approach showed slightly better results than PCD [14, pp. 389] and it performed even better in a modified version [14, p. 386]. A detailed description of the LSC approach is given in Section 4.

The Augmented Lagrangian preconditioner was introduced in [4] and analyzed in [5, 7]. It can be used as left or right preconditioner. A modified version, which considers basically the upper triangular block of the Augmented Lagrangian preconditioner, was proposed in [6]. The Augmented Lagrangian preconditioner is similar to a matrix that is obtained by augmenting a mixed finite element method with a grad-div stabilization. In fact, in [19], a preconditioner based on the grad-div stabilization, instead of adding the augmentation, is proposed and studied. The case of the so-called sparse grad-div stabilization is considered in [29].

This paper presents an assessment of solvers for algebraic linear saddle point problems (1) which arise in the linearization and finite element discretization of the incompressible Navier–Stokes equations. As Krylov subspace method, the flexible GMRES (FGMRES) method [32] is used. Our main interest consists in comparing the performance of preconditioners from the two classes mentioned above: a coupled multigrid method with Vanka-type smoothers [37] and LSC-type preconditioners. To the best of our knowledge, such a comparison is not yet available in the literature. We preferred to study LSC-type preconditioners, instead of the Augmented Lagrangian preconditioner, because LSC preconditioners do not possess an algorithmic parameter that has to be chosen by the user, which is in our opinion an advantage. In fact, it is known that the parameter of the Augmented Lagrangian method should neither be chosen too small nor too large [18]. Since we think that it is of much interest for a broad audience, also the sparse direct solver UMFPACK [10] is included in the assessment. The numerical studies consider two- and three-dimensional as well as steady-state and time-dependent problems.

We would like to add that for time-dependent problems there is a popular class of methods where the solution of linear saddle point problems is avoided, namely splitting schemes, see the review in [17]. However, our own experience with some standard splitting schemes is that they are inferior with respect to the accuracy of the computed solutions compared with inf-sup stable mixed discretizations, see [24, Ex. 7.101]. Altogether, an assessment of coupled approaches vs. splitting schemes concerning various aspects (accuracy, efficiency, scalability on parallel computers) is certainly of great interest. However, it is outside the scope of the present paper.

This paper is organized as follows. Section 2 explains briefly the used discretization strategies for the Navier–Stokes equations. In Section 3, the coupled multigrid methods are presented, with some emphasis on the used smoothers. The LSC preconditioners are described in Section 4. Section 5 contains the numerical studies and in Section 6 some aspects of the application of the methods on parallel computers are discussed. The most important results are summarized in Section 7.

2. The Navier–Stokes equations and linear saddle-point problems

2.1. The stationary case

Let Ω be a bounded domain in $\mathbb{R}^d$, $d \in \{2, 3\}$. A stationary flow of an incompressible Newtonian fluid which is not exerted to external forces is modeled with the steady-state Navier–Stokes equations

$$\begin{aligned} -\nu\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p = \boldsymbol{0} \quad & \text{in } \Omega, \\ \nabla\cdot\boldsymbol{u} = 0 \quad & \text{in } \Omega, \end{aligned} \tag{2}$$

where the velocity field $\boldsymbol{u}$ and the pressure p are the unknown quantities. Bold symbols are used to distinguish vector-valued from scalar quantities. The material constant ν is the kinematic viscosity (in (2) already in dimensionless form). The inverse of this dimensionless viscosity is the Reynolds number, which is commonly used to characterize and classify flows.

In order to define a well-posed problem, system (2) has to be equipped with boundary conditions. In the considered examples, the boundary $\partial\Omega$ of Ω is decomposed as $\partial\Omega = \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \cup \Gamma_{\text{no-slip}}$, where the decomposition is disjoint. Then inflow, outflow, and no-slip boundary conditions are prescribed as follows

$$\begin{array}{rcll} \boldsymbol{u} & = & \boldsymbol{g}(\boldsymbol{x}) & \text{on } \Gamma_{\text{in}}, \\ (\nu\nabla\boldsymbol{u} - p\boldsymbol{I})\boldsymbol{n} & = & \boldsymbol{0} & \text{on } \Gamma_{\text{out}}, \\ \boldsymbol{u} & = & \boldsymbol{0} & \text{on } \Gamma_{\text{no-slip}}. \end{array} \tag{3}$$

With these boundary conditions, one can derive a weak formulation of (2). For this purpose, one introduces the velocity test and ansatz spaces

$$V_0 = \left(H^1_\Gamma(\Omega)\right)^d, \quad V_{\boldsymbol{g}} = \left\{\boldsymbol{v} \ : \ \boldsymbol{v} \in \left(H^1(\Omega)\right)^d \text{ with } \boldsymbol{v}|_{\Gamma_{\text{in}}} = \boldsymbol{g}, \boldsymbol{v}|_{\Gamma_{\text{no-slip}}} = \boldsymbol{0}\right\},$$

and the pressure space $Q = L^2(\Omega)$. The subspace $H^1_\Gamma(\Omega)$ of $H^1(\Omega)$ consists of those functions which vanish on $\Gamma = \Gamma_{\text{in}} \cup \Gamma_{\text{no-slip}}$. Now the weak formulation of (2) reads as follows: Find $(\boldsymbol{u}, p) \in V_{\boldsymbol{g}} \times Q$ such that for all pairs of test functions $(\boldsymbol{v}, q) \in V_0 \times Q$, it holds

$$\begin{aligned} (\nu\nabla\boldsymbol{u}, \nabla\boldsymbol{v}) + ((\boldsymbol{u}\cdot\nabla)\boldsymbol{u}, \boldsymbol{v}) - (\nabla\cdot\boldsymbol{v}, p) &= 0, \\ (\nabla\cdot\boldsymbol{u}, q) &= 0, \end{aligned} \tag{4}$$

where $(\cdot,\cdot)$ denotes the inner product in $L^2(\Omega)$. The numerical solution of (4) requires a linearization of the nonlinear convective term $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ and a discretization in space.

The linearization is achieved by a Picard-type fixed point iteration. With a known approximation to the solution, the convection field $\boldsymbol{u}^m \in V_{\boldsymbol{g}}$, the nonlinear convective term $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ is replaced with $(\boldsymbol{u}^m\cdot\nabla)\boldsymbol{u}$, leading to a so-called Oseen problem

$$\begin{aligned} (\nu\nabla\boldsymbol{u}^{m+1}, \nabla\boldsymbol{v}) + \big((\boldsymbol{u}^m\cdot\nabla)\boldsymbol{u}^{m+1}, \boldsymbol{v}\big) - (\nabla\cdot\boldsymbol{v}, p^{m+1}) &= 0 \quad \forall\, \boldsymbol{v} \in V_0, \\ (\nabla\cdot\boldsymbol{u}^{m+1}, q) &= 0 \quad \forall\, q \in Q, \end{aligned} \tag{5}$$

to obtain a new approximation $\boldsymbol{u}^{m+1}$. This process is iterated until a sufficiently accurate approximate solution is reached.

For the solution of (5), a spatial discretization of test and ansatz spaces with inf-sup stable pairs of finite element spaces is used. Let $V_0^h \subset V_0$, $V_{\boldsymbol{g}}^h \subset V_{\boldsymbol{g}}$, and $Q^h \subset Q$ be conforming finite element spaces, where V_0^h and Q^h satisfy a discrete inf-sup condition with a constant independent of the refinement level, then the finite element formulation of (5) reads as follows: Find $(\boldsymbol{u}^{h,m+1}, p^{h,m+1}) \in$

$V_{\boldsymbol{g}}^h \times Q^h$ such that

$$\begin{aligned}(\nu\nabla\boldsymbol{u}^{h,m+1}, \nabla\boldsymbol{v}^h) + \big((\boldsymbol{u}^{h,m}\cdot\nabla)\boldsymbol{u}^{h,m+1}, \boldsymbol{v}^h\big) - (\nabla\cdot\boldsymbol{v}^h, p^{h,m+1}) &= 0 \quad \forall\ \boldsymbol{v}^h \in V_0^h,\\ (\nabla\cdot\boldsymbol{u}^{h,m+1}, q^h) &= 0 \quad \forall\ q^h \in Q^h.\end{aligned} \tag{6}$$

System (6) is equivalent to an algebraic linear saddle point system of the form

$$\mathcal{A}\begin{pmatrix} \underline{u}^{m+1} \\ \underline{p}^{m+1} \end{pmatrix} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} \underline{u}^{m+1} \\ \underline{p}^{m+1} \end{pmatrix} = \begin{pmatrix} \underline{f} \\ \underline{g} \end{pmatrix}, \tag{7}$$

which has to be solved in every step of the fixed point iteration. Here, $A \in \mathbb{R}^{n\times n}$ is a nonsingular square matrix, and $B \in \mathbb{R}^{k\times n}$ is a full rank rectangular matrix where $k < n$. The unknowns $(\underline{u}^{m+1}, \underline{p}^{m+1})$ are coefficient vectors for the finite element functions of the ansatz spaces. The right-hand side in (7) arises, e.g., from boundary conditions and in the special case considered here, it is $\underline{g} = \underline{0}$.

2.2. The time-dependent case

Whenever some data of the Navier–Stokes equations depend on time or ν is sufficiently small, i.e., the Reynolds number of the considered problem is sufficiently large, the behavior of the flow field becomes time-dependent. In such situations, the flow is modeled by the time-dependent Navier–Stokes equations which read, for Ω as before and $T \in \mathbb{R}_+$,

$$\begin{aligned}\partial_t\boldsymbol{u} - \nu\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p &= \boldsymbol{0} \quad \text{in } (0,T]\times\Omega,\\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } (0,T]\times\Omega.\end{aligned} \tag{8}$$

These equations have to be equipped with boundary conditions, which are in the considered examples analogous to (3):

$$\begin{array}{rcll} \boldsymbol{u} &=& \boldsymbol{g}(t,\boldsymbol{x}) & \text{on } (0,T]\times\Gamma_{\text{in}},\\ (\nu\nabla\boldsymbol{u} - p\boldsymbol{I})\boldsymbol{n} &=& \boldsymbol{0} & \text{on } (0,T]\times\Gamma_{\text{out}},\\ \boldsymbol{u} &=& \boldsymbol{0} & \text{on } (0,T]\times\Gamma_{\text{no-slip}}.\end{array}$$

The equations are closed with an initial condition

$$\boldsymbol{u}(0,\cdot) = \boldsymbol{u}_0 \quad \text{in } \Omega,$$

where $\boldsymbol{u}_0$ has to satisfy the divergence constraint and the boundary conditions in an appropriate sense.

In order to proceed as in the steady-state case, one can start with a discretization of the temporal derivative. In the simulations presented in this paper, the Crank–Nicolson scheme was used with a fixed time step length Δt. Applying this scheme and multiplying the whole system with Δt leads to the following time-discrete version of (8) at the discrete time t_n:

$$\begin{array}{rcl} & & \boldsymbol{u}_n + \dfrac{1}{2}\Delta t\left(-\nu\Delta\boldsymbol{u}_n + (\boldsymbol{u}_n\cdot\nabla)\boldsymbol{u}_n\right) + \Delta t\nabla p_n\\ &=& \boldsymbol{u}_{n-1} - \dfrac{1}{2}\Delta t\left(-\nu\Delta\boldsymbol{u}_{n-1} + (\boldsymbol{u}_{n-1}\cdot\nabla)\boldsymbol{u}_{n-1}\right),\\ \Delta t\nabla\cdot\boldsymbol{u}_n &=& 0,\end{array}$$

where $(\boldsymbol{u}_n, p_n)$ denote velocity and pressure at t_n. For a brief remark concerning the treatment of the pressure term, it is referred to [24, Rem. 7.50]. To derive a variational formulation, one uses similar ansatz and test spaces as in the steady-state case. The only difference is that the velocity ansatz space for every time point t_n is replaced by $V_{\boldsymbol{g}}(t_n, \cdot)$, which is defined in the same way as was $V_{\boldsymbol{g}}$, accounting for a time-dependent inflow. The variational formulation is once again linearized with a Picard-type iteration. In one iterative step, the following problem has to be solved for the unknown $(\boldsymbol{u}_n^{m+1}, p_n^{m+1}) \in V_{\boldsymbol{g}}(t_n, \cdot) \times Q$ and given $\boldsymbol{u}_n^m \in V_{\boldsymbol{g}}(t_n, \cdot)$:

$$\begin{aligned}(\boldsymbol{u}_n^{m+1}, \boldsymbol{v}) + \frac{1}{2}\Delta t \left((\nu\nabla\boldsymbol{u}_n^{m+1}, \nabla\boldsymbol{v}) + ((\boldsymbol{u}_n^m \cdot \nabla)\boldsymbol{u}_n^{m+1}, \boldsymbol{v})\right) - \Delta t(p_n^{m+1}, \nabla\cdot\boldsymbol{v}) \\ = (\boldsymbol{u}_{n-1}, \boldsymbol{v}) - \frac{1}{2}\Delta t\left(\nu(\nabla\boldsymbol{u}_{n-1}, \nabla\boldsymbol{v}) + ((\boldsymbol{u}_{n-1}\cdot\nabla)\boldsymbol{u}_{n-1}, \boldsymbol{v})\right), \\ 0 = \Delta t(\nabla\cdot\boldsymbol{u}_n^{m+1}, q).\end{aligned} \tag{9}$$

Here $\boldsymbol{u}_n^m$ is the convection field, which is again chosen as the solution of the previous iteration step. As a starting point for the iteration serves the solution of the previous time step, $(\boldsymbol{u}_n^0, p_n^0) = (\boldsymbol{u}_{n-1}, p_{n-1})$. The spatial discretization is performed with an inf-sup stable pair of finite element spaces. Its detailed description results in almost literally the same text as written after (5), with the index h for all finite element functions, $V_{\boldsymbol{g}}^h(t_n, \cdot)$ replacing $V_{\boldsymbol{g}}^h$, and applied to system (9). From the point of view of linear algebra, a linear saddle point system of form (7) is obtained.

Popular alternatives to the fully implicit approach (9) are IMEX schemes, which use as convection field not $\boldsymbol{u}_n^m$ but an extrapolation of the velocity from previous time instances, e.g., see [20] or [24, Rem. 7.61] for concrete proposals. Thus, in IMEX schemes one has to solve only one linear saddle point problem in each discrete time. Since we have a long and good experience with the fully implicit approach, we decided to use it also in the numerical studies presented in this paper. A brief discussion of IMEX schemes is provided at the end of Section 5.3.

2.3. Main difference of stationary and time-dependent case

Although the linearization and discretization of the stationary and time-dependent Navier–Stokes equations lead to the same type (7) of linear saddle point problems, there is an essential difference in the properties of the system matrix. If the time-step is not too large, then the matrix A in the time-dependent case is dominated by the mass matrix $(\boldsymbol{\phi}_j, \boldsymbol{\phi}_i)_{ij}$, which arises in the discretization of the temporal derivative. All other contributions of A and also all other matrices are multiplied with Δt. In the stationary case, one has to distinguish two regimes. If the viscosity of the flow is large, the dominating contribution in A is $(\nu\nabla\boldsymbol{\phi}_j, \nabla\boldsymbol{\phi}_i)_{ij}$, which comes from the discretization of the viscous term. In the more interesting case that ν is small, the dominating contribution arises from the convective term.

The numerical studies will consider steady-state situations with dominating convection and time-dependent problems with sufficiently small time steps. Hence, the matrix A in both cases has different properties that might have a different impact on the efficiency of the studied solvers.

3. Coupled multigrid preconditioning

Multigrid approaches for solving large systems of linear equations show their full potential when used as preconditioners for Krylov subspace methods like flexible GMRES (FGMRES) [32]. Besides giving below a brief overview on the components of the multigrid method, a detailed description of the used smoother will be provided since the chosen smoother is key to the efficiency of a multigrid method.

3.1. The standard and the multiple-discretization coupled multigrid approach

From the linearization and discretization of the Navier–Stokes equations arise linear systems of equations with a block structure as in (7). Multigrid approaches that do not decouple these systems but solve for velocity $\underline{u}$ and pressure $\underline{p}$ simultaneously are referred to as coupled multigrid approaches.

To define a multigrid method, the following components have to be specified:

- the grid hierarchy,
- the grid transfer operators, i.e., restriction and prolongation,
- the grid cycle, i.e., the sequence in which the levels of the grid hierarchy are addressed,
- the smoother, i.e., an approximate solver on levels which are not the coarsest one,
- the solver on the coarsest grid.

In the standard multigrid (MG) approach, there is a one-to-one mapping between the geometric refinements of an initial grid and the levels in the multigrid hierarchy. In addition, all levels are equipped with the same type of finite elements and discretization.

The grid transfer operators are the tools used to pass information between the different levels. Consider the meshes $\mathcal{T}_{l-1}$ and $\mathcal{T}_l$, where $\mathcal{T}_l$, $l > 1$, originates from the refinement of the coarser mesh $\mathcal{T}_{l-1}$, and the corresponding finite element spaces V_{l-1} and V_l for the velocity as well as Q_{l-1} and Q_l for the pressure. The prolongation operators

$$P^{\boldsymbol{u}}_{l-1,l} : V_{l-1} \to V_l, \quad P^{p}_{l-1,l} : Q_{l-1} \to Q_l, \quad 1 \le l \le L,$$

and the restriction operators

$$R^{\boldsymbol{u}}_{l,l-1} : V_l \to V_{l-1}, \quad R^{p}_{l,l-1} : Q_l \to Q_{l-1} \quad 1 \le l \le L,$$

map finite element functions between grids. The operators proposed in [34] are based on the concept of local and global functionals and can be used for an almost arbitrary choice of finite element spaces. The used code utilizes the

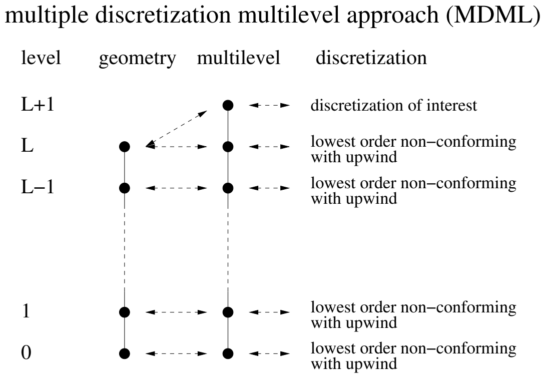
multiple discretization multilevel approach (MDML)
level
geometry
multilevel
discretization
L+1
L
L−1
1
0
discretization of interest
lowest order non−conforming with upwind
lowest order non−conforming with upwind
lowest order non−conforming with upwind
lowest order non−conforming with upwind

Figure 1: Sketch of the multiple discretization multilevel (MDML) method.

concept of mapped finite elements. Thus, the implementation of the grid transfer operators from [34] is as follows. For each mesh cell, the residual or function to be transferred is mapped to a reference cell, the transfer operation is performed, and the result is mapped back. The transfer operators on the reference cell are given by small matrices, depending on the involved finite element spaces. These matrices are computed only once and then stored in a database such that subsequent transfer operations can be computed efficiently by matrix-vector products. For a detailed discussion of the applied operators it is referred to [24, Sec. 9.2.2], and [34].

Concerning the grid cycle, usual choices comprise the V-, F-, and W-cycle. The least work per cycle is needed in the V-cycle but the W-cycle is sometimes considerably more stable. The F-cycle, which is in between the V- and the W-cycle, is in our experience a good compromise.

The multigrid method works solely with the finite element spaces. This feature enables the definition of multigrid-type methods with different finite element spaces on different geometric grids and even with more than one finite element space defined on the same geometric grid. An example of this approach is the multiple discretization multilevel (MDML) method introduced in [26] and analyzed in [21], see Figure 1. The main motivation for constructing this method is the experience that multigrid methods are usually very efficient for lowest order finite elements. Thus, all coarser multigrid levels of the MDML method are equipped with an inf-sup stable and convection-stabilized lowest order discretization of the Oseen problems. On the finest geometric grid, the discretization of interest forms the finest multigrid level and the lowest order discretization forms the next coarser multigrid level.

If Dirichlet boundary conditions are prescribed on the whole boundary, the pressure finite element space is a subspace of $L_0^2(\Omega)$, i.e., the integral mean value of the pressure has to vanish. In this situation, coupled multigrid preconditioners require a preprocessing step at the beginning of each step of the Picard or Newton iteration and a postprocessing step after the termination of this iteration. Here, the approach that is applied in the used code ParMooN is described. In the preprocessing step, the arithmetic mean of the pressure components of the residual vector is computed and then subtracted from these components. In this way, the right-hand side of the equation for the error, which is considered in multigrid methods, is projected to the range of the matrix. In the postprocessing step, the integral mean of the computed pressure is calculated and subtracted from the pressure to ensure that the finite element pressure solution is in $L_0^2(\Omega)$.

3.2. Vanka-type smoothers

As already mentioned, the choice of the smoother is essential for the efficiency of a multigrid method. It was already discussed in the introduction that the difficulty for linear saddle point problems consists in the fact that standard smoothers cannot be applied because of the zero diagonal block and special smoothers need to be designed.

The most popular class of smoothers for problems of type (7) are Vanka-type smoothers proposed in [37]. Since the smoothers are the essential part of the multigrid method and to keep this paper self-contained, the used Vanka smoothers will be described in some detail.

Vanka-type smoothers can be understood as block Gauss–Seidel methods. Let the problem on multigrid level l have the form (7) and denote the sets of velocity and pressure degrees of freedom by $\mathcal{V}^l$ and $\mathcal{Q}^l$, respectively. These sets are decomposed into (not necessarily disjoint) subsets

$$\mathcal{V}^l = \bigcup_{j=1}^{J} \mathcal{V}_j^l, \quad \mathcal{Q}^l = \bigcup_{j=1}^{J} \mathcal{Q}_j^l. \tag{10}$$

Local matrices $\mathcal{A}_j^l$ are defined to contain those entries of the global matrix $\mathcal{A}^l$, whose row and column correspond to degrees of freedom from $\mathcal{V}_j^l \cup \mathcal{Q}_j^l$. Then each smoothing step with a Vanka-type smoother consists in a loop from $j = 1, \dots, J$ and in solving a small linear system of equations connected with the corresponding degrees of freedom. Denote by $(\cdot)_j$ the restriction of a vector to the rows corresponding to the degrees of freedom in $\mathcal{V}_j^l \cup \mathcal{Q}_j^l$. Then in each smoothing step, a solution update of the form

$$\begin{pmatrix} \underline{u} \\ \underline{p} \end{pmatrix}_j \leftarrow \begin{pmatrix} \underline{u} \\ \underline{p} \end{pmatrix}_j + \mathcal{A}_j^{-1} \left(\begin{pmatrix} \underline{f} \\ \underline{g} \end{pmatrix} - \mathcal{A} \begin{pmatrix} \underline{u} \\ \underline{p} \end{pmatrix} \right)_j$$

is performed. This block Gauss–Seidel approach is called multiplicative Vanka smoother. It is completely described if the decomposition (10) is given. Our strategy to define a decomposition is as follows:

- Take some pressure degrees of freedom which form $\mathcal{Q}_j^l$.
- The corresponding velocity degrees of freedom in $\mathcal{V}_j^l$ are all those that are connected to at least one of the pressure degrees of freedom in $\mathcal{Q}_j^l$ by an existing entry in the sparsity pattern in the off-diagonal block B of $\mathcal{A}$.

It can be seen that with this strategy a Vanka-type smoother is determined by the particular choice of the $\mathcal{Q}_j^l$, $j = 1, \ldots, J$.

To define the sets $\mathcal{Q}_j^l$, in our experience, it is helpful to distinguish between discretizations with continuous and discontinuous pressure finite element spaces. For discontinuous approximations, the following Vanka smoother is appropriate:

- *Mesh-cell-oriented Vanka smoother (cell Vanka smoother).* This smoother takes for $\mathcal{Q}_j^l$ all pressure degrees of freedom that belong to one mesh cell. It turns out that the corresponding velocity degrees of freedom are all those which belong to the same mesh cell. The number of local systems to be solved in one smoothing step then equals the number of cells in the mesh $\mathcal{T}^l$ and all local systems are of the same size.

Smoothers for discretizations with continuous pressure are the following:

- *Pressure-node-oriented Vanka smoother (nodal Vanka smoother).* To define this smoother, one takes for $\mathcal{Q}_j^l$ only one pressure degree of freedom. Then the number of local systems to be solved in each smoothing step is the number of pressure degrees of freedom, and systems of different sizes appear. The size of the systems depends on several aspects, like the geometric position of the (Lagrangian) pressure degree of freedom, the local grid, or the proximity to the boundary, see [24, Sec. 9.2.2] for more details and some examples.
- *Cell-patch-oriented Vanka smoother (patch Vanka smoother).* This approach can be understood as a cell-oriented Vanka smoother applied to a continuous pressure approximation. Each set $\mathcal{Q}_j^l$ is defined by gathering all pressure degrees of freedom belonging to one mesh cell. For a continuous pressure approximation, the pressure degrees of freedom are connected via B to velocity degrees of freedom in neighboring cells. The number of local systems per smoothing step equals the number of mesh cells. This approach leads to local systems of different sizes, depending on the local mesh structure or the proximity to the boundary. It is clear by construction that the dimension of the local systems is generally larger than for the nodal Vanka smoother.

In previous studies of Vanka-type smoothers, the application of the patch Vanka smoother was not yet an option due to the relatively large local systems to be solved, e.g., see the statement of even applying an iterative scheme for solving the local systems if the dimension exceeded 100 in [23]. However, with the enormous increase of computing power during the last decade, methods that apply direct solvers for the solution of larger local systems gained efficiency. The current paper will explore, besides other issues, whether the patch Vanka smoother is already competitive.

Note that for the lowest order nonconforming discretizations P_1^{nc}/P_0 [9] and Q_1^{rot}/Q_0 [31] the cell and the nodal Vanka smoothers are identical. In numerical tests, we could observe that for higher order discretizations with discontinuous pressure, the cell Vanka smoother performed much more efficient than the nodal Vanka smoother. Thus, for the sake of brevity, the combination of the nodal Vanka smoother and higher order discretizations with discontinuous pressure will not be considered in the numerical studies presented in Section 5.

In our implementation, the local systems with the matrices $\{\mathcal{A}_j\}$ are solved directly by using the LAPACK routines `dgetrf` and `dgetrs`. All local matrices are collected from the global matrix every time they are needed and then the routines for the factorization and solution of the triangular systems of equations are called. An alternative approach consists in storing all factorized local matrices on all levels of the multigrid. This approach results in an considerable increase of the memory requirements. The factorized matrices can be used for all smoothing steps within one Picard or Newton iteration. For the next iteration, new matrices have to be build and factorized. Some of our experience with this alternative approach is summarized at the end of Section 5.2.

4. Least Squares Commutator (LSC) preconditioners

The LSC preconditioner decouples the update of the velocity and pressure degrees of freedom.

4.1. The basic approach

The LSC preconditioner is derived from the LU decomposition of the matrix $\mathcal{A}$ and the approximation of the pressure Schur complement by keeping a certain operator commutator error small. This approach will be presented briefly. A detailed explanation is available in [14] and some hints on the intuition when introducing the commutator can be found in the original work [12].

A formal Gaussian elimination of $\mathcal{A}$ from (7) gives the LU decomposition

$$\mathcal{A} = \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & -BA^{-1}B^T \end{pmatrix} = LU. \tag{11}$$

As lower right matrix block appears the so-called Schur complement of $\mathcal{A}$,

$$S := -BA^{-1}B^T.$$

Since from (11) it follows that $\mathcal{A}U^{-1} = L$, which has perfectly clustered eigenvalues, the upper triangular factor U is a good starting point for building preconditioners. Its drawback is the appearance of the Schur complement which is not explicitly available and even if this would be the case, then the Schur complement would be a dense matrix, since A^{-1} is dense. Constructing a good approximation to the Schur complement is the difficulty that is addresses by the LSC preconditioner.

The basic idea of the LSC preconditioner is to search for a regular matrix $A_p \in \mathbb{R}^{k\times k}$, acting on (coefficients of) the pressure space, that solves the equation

$$B^T A_p = AB^T \tag{12}$$

and thus gives, by transforming (12) equivalently and multiplying with B from the left,

$$-BA^{-1}B^T = -BB^T A_p^{-1}. \tag{13}$$

The right-hand side of (13) is a better to handle form of the Schur complement. For this form, applying U as a preconditioner requires approximating the action of $\left(-BB^T A_p^{-1}\right)^{-1}$, which is more easily done, as A_p is known and BB^T is positive definite and symmetric, and it represents basically a discretization of a pressure Poisson problem.

The difficulty is that $B^T \in \mathbb{R}^{n\times k}$, $n > k$, is a full rank rectangular matrix and so (12) is in general an overdetermined system and can only be solved in a minimizing sense

$$\min_{A_p} \ \left\| AB^T - B^T A_p \right\| \tag{14}$$

for some matrix norm $\|\cdot\|$ to be defined later.

The derivation of the LSC preconditioner as commutator-based is now motivated by the interpretation of the matrices appearing in (14) as discrete counterparts of the underlying continuous operators from the (steady-state) Navier–Stokes equations. In fact the matrix B^T stems from the finite element discretization of the gradient operator and the matrix A from a convection-diffusion operator $-\nu\Delta + \boldsymbol{u}^m \cdot \nabla$ acting on the velocity space. The unknown matrix A_p is now assumed to originate from the discretization of a somewhat hypothetical convection-diffusion operator acting on the pressure space. Problem (14) can then be interpreted as minimizing the discrete commutation error of velocity and pressure convection-diffusion operator with the gradient operator. To support this interpretation, one has to account for the concrete choice of the finite element spaces and to introduce appropriate weights by multiplying with the inverses of the velocity and pressure mass matrices $M_v \in \mathbb{R}^{n\times n}$ and $M_p \in \mathbb{R}^{k\times k}$. One now replaces (14) by the minimizing problem

$$\min_{A_p} \ \left\| M_v^{-1} A M_v^{-1} B^T - M_v^{-1} B^T M_p^{-1} A_p \right\|. \tag{15}$$

Observe that by multiplication from left with $BA^{-1}M_v$ and from right with $A_p^{-1}M_p$ the term inside the norm gives rise to a formula for the approximation of the Schur complement

$$S = -BA^{-1}B^T \approx -BM_v^{-1}B^T A_p^{-1} M_p. \tag{16}$$

The LSC approach now proceeds by specifying the minimizing problem (15) as minimizing columnwise in a M_v-weighted vector norm

$$\|v\|_{M_v} = \langle M_v v, v\rangle^{\frac{1}{2}}.$$

This choice leads to the eponymous least squares problems

$$\min_{[a_p]_j} \ \left\| [M_v^{-1} A M_v^{-1} B^T]_j - M_v^{-1} B^T M_p^{-1} [a_p]_j \right\|_{M_v}, \quad j = 1, \dots, k, \tag{17}$$

where the unknowns $[a_p]_j$ are the columns of A_p. The first order optimality conditions read

$$M_p^{-1} B M_v^{-1} B^T M_p^{-1} \left[a_p\right]_j = \left[M_p^{-1} B M_v^{-1} A M_v^{-1} B^T\right]_j, \quad j = 1, \dots, k.$$

In this way, one finally gets the representation

$$A_p = M_p \left(BM_v^{-1}B^T\right)^{-1} \left(BM_v^{-1}AM_v^{-1}B^T\right). \tag{18}$$

The LSC preconditioner is now obtained by replacing M_v^{-1} with $(\mathrm{diag}(M_v))^{-1} = D_v^{-1}$ in (18) and inserting the arising formula in (16)

$$S_{\mathrm{LSC}} := -\left(BD_v^{-1}B^T\right)\left(BD_v^{-1}AD_v^{-1}B^T\right)^{-1}\left(BD_v^{-1}B^T\right). \tag{19}$$

This expression approximates the lower right block in (11).

Note that in the application of the preconditioner, pressure Poisson-type problems have to be solved by inverting the first and last term in parentheses in (19). A problem for the velocity has to be solved by inverting the upper left matrix in (11).

In our implementation, the saddle point problem (1) is assembled for all degrees of freedom including those on the Dirichlet boundary. Then, the respective rows of A and B^T are changed such that the Dirichlet values are imposed. With this approach, the matrix B of (1) maintains full rank and this matrix and its transposed are used for building the matrix $BD_v^{-1}B^T$ of the pressure system. Hence, the matrix for the pressure system is non-singular, independently of having Dirichlet conditions on the whole boundary or not.

4.2. Incorporating boundary effects

Since its original development in [12] the LSC preconditioner has experienced two major extensions. The first one, to extend the approach to inf-sup-stabilized finite element approximations, has been performed in [13]. Since stabilized discretizations were not used in our numerical studies, here only the second modification, which considers the incorporation of boundary conditions into the pressure convection-diffusion operator, will be described. It is reported, e.g., in [14, Section 9.2.4], that this modification led to improvements in the performance compared with the LSC preconditioner.

The derivation starts with the continuous version of a commutator

$$\mathrm{div}\,\mathfrak{A} - \mathfrak{A}_p\,\mathrm{div}$$

with velocity convection-diffusion operator $\mathfrak{A}$ and a hypothetical pressure convection-diffusion operator $\mathfrak{A}_p$. Observe that the original notion of "commutator with the gradient operator" has been replaced by "commutator with divergence operator", an approach justified in [15]. Assuming that making the commutator error of the continuous version "small" also makes the discrete commutator error small motivates the investigation of the commutator error at the domain boundaries. A careful analysis given in [15] shows that in the one-dimensional case the commutator error (continuous and discrete) vanishes if $\mathfrak{A}_p$ is equipped with a Robin boundary conditions at the inflow and a Dirichlet boundary condition at the outflow. Then, it was proceeded with transferring these observations to the higher-dimensional case by splitting the commutator error into components associated with coordinate directions of its factors. It was shown that these error components depend too strongly on each other to set them to zero simultaneously, but the most perturbing parts can be suppressed by carefully weighting the least squares problem (17).

The least squares problem (17) is replaced by

$$\min_{[a_p]_j} \left\| [M_v^{-1} A M_v^{-1} B^T]_j - M_v^{-1} B^T M_p^{-1} [a_p]_j \right\|_{\tilde{M_v}}, \quad j = 1, \dots, k,$$

where the modified norm $\| \cdot \|_{\tilde{M_v}} = \langle \tilde{M_v} \cdot, \cdot \rangle^{\frac{1}{2}}$ with

$$\tilde{M_v} = M_v D^{\frac{1}{2}} M_v^{-1} D^{\frac{1}{2}} M_v$$

is employed. The diagonal matrix $D = (d_{ij})_{i,j}$ is responsible for suppressing certain error contributions. It was proposed in [15] to suppress contributions tangential to the Dirichlet boundaries. For a grid aligned domain[1] in two dimensions, D takes the block form

$$D = \begin{pmatrix} D_x & 0 \\ 0 & D_y \end{pmatrix}.$$

The entries of the diagonal sub-matrix D_x are responsible for suppressing contributions tangential to horizontal (x-aligned) boundaries. Its diagonal entries are defined by

$$d_{ii} = \begin{cases} \varepsilon & \text{if the velocity degree of freedom } i \text{ is connected to a pressure} \\ & \text{degree of freedom on a horizontal Dirichlet boundary by an} \\ & \text{entry in the sparsity pattern of } B, \\ 1 & \text{else.} \end{cases}$$

The entries of D_y are defined in a similar way for tangential velocity degrees of freedom near vertical Dirichlet domain boundaries. The parameter ε, responsible for the suppressing, is chosen empirically as $\varepsilon = 0.1$. This choice is proposed in [14]. The definition of D can be adapted to the three-dimensional case and non-grid aligned domains, see [14].

Analogously to the treatment of (17), one obtains the boundary-corrected LSC preconditioner by determining the first order optimality conditions and plugging the result in (16). One gets

$$S_{\text{LSC}}^{\text{bdry}} := -\left(BH^{-1}B^T\right)\left(BD_v^{-1}AH^{-1}B^T\right)^{-1}\left(BD_v^{-1}B^T\right),$$

where now

$$H = D^{-\frac{1}{2}} D_v D^{-\frac{1}{2}}.$$

Numerical results given in [14] show a significant improvement of the boundary-corrected LSC preconditioner compared with its basic version in terms of needed GMRES iterations to achieve a certain error tolerance.[2]

[1] A domain with a rectangular/hexahedral grid where each mesh edge lies parallel to one boundary part.

[2] Our implementation of both LSC variants was verified by reproducing the results from

5. Numerical studies

5.1. General setup

Numerical studies were performed mostly on benchmark problems for the steady-state and time-dependent Navier–Stokes equations in 2d and 3d, namely for flow around cylinder examples defined in [33]. Only for the case of a time-dependent flow in 3d, a modified setup was used, which is motivated and described below. Besides [33], descriptions of these examples can be found at many places, e.g., in [22, 23, 24, 26], such that here only a brief explanation is provided.

The Picard iterations for solving the nonlinear problems were terminated if the Euclidean norm of the residual vector was less than 10^{-8}. For discretizing the arising saddle point problems, the Galerkin finite element method with inf-sup stable pairs of finite element spaces with second order velocity and first order pressure was used. On simplicial grids, the Taylor–Hood pair P_2/P_1 and the pair $P_2^{\mathrm{bubble}}/P_1^{\mathrm{disc}}$ were studied and on quadrilateral/hexahedral grids the Taylor–Hood pair Q_2/Q_1 and the pair Q_2/P_1^{disc} were applied. Hence, for both types of grids, a pair with continuous and a pair with discontinuous pressure was used. Both Taylor–Hood pairs and Q_2/P_1^{disc} belong to the most popular choices of inf-sup stable finite elements for incompressible flow problems.

As discretization of the temporal derivative, the Crank–Nicolson scheme with an equidistant time step Δt was used.

The following solvers for the arising linear saddle point problems were studied:

- UMFPACK: sparse direct solver [10],
- FGMRES + MG(cell): flexible GMRES, preconditioner standard multigrid with cell Vanka smoother, only discretizations with discontinuous pressure,
- FGMRES + MDML(cell): flexible GMRES, preconditioner MDML method with cell Vanka smoother, only discretizations with discontinuous pressure,
- FGMRES + MG(nodal): flexible GMRES, preconditioner standard multigrid with nodal Vanka smoother, only discretizations with continuous pressure,
- FGMRES + MDML(nodal): flexible GMRES, preconditioner MDML method with nodal Vanka smoother, only discretizations with continuous pressure,
- FGMRES + MG(patch): flexible GMRES, preconditioner standard multigrid with patch Vanka smoother, only discretizations with continuous pressure,
- FGMRES + MDML(patch): flexible GMRES, preconditioner MDML method with patch Vanka smoother, only discretizations with continuous pressure,
- FGMRES + LSC(dir): flexible GMRES, preconditioner least squares commutator, sparse direct solver for all linear subsystems,

[14] for the lid-driven cavity example. The characteristic number for the performance of the preconditioners is the number of GMRES iterations needed in the last step of the Picard iteration. Whereas the original LSC exhibits an unfavorable grid dependency, the boundary-corrected LSC fixes this issue.

- FGMRES + LSC(ite): flexible GMRES, preconditioner least squares commutator, iterative solver for velocity system, see Sections 5.2 and 5.3 for details,
- FGMRES + boundary-corr. LSC(dir): flexible GMRES, preconditioner boundary-corrected least squares commutator, sparse direct solver for all linear subsystems,
- FGMRES + boundary-corr. LSC(ite): flexible GMRES, preconditioner boundary-corrected least squares commutator, iterative solver for velocity system.

As in the classical multigrid approach, the systems on the coarsest grids were solved directly with UMFPACK.

In the multigrid approaches there is the possibility, and generally the necessity, to apply a damping. There are two opportunities for damping, namely for the update that is proposed from the Vanka smoother and for the update that comes from the prolongation from a coarse level to the next finer one. Both possibilities for damping are independent and generally, one gets the most efficient method by a different choice of the corresponding damping parameters. However, to facilitate the numerical studies and the application of the multigrid methods, only configurations with a single damping parameter for both places were considered.

The simulations were performed with the finite element code ParMooN [39] on compute servers HP BL460c Gen9 2xXeon, 2600MHz, using only one processor for all routines. All simulations were performed five times, the fastest and the slowest computing time were neglected and the average of the remaining three times is presented below. The given computing times include the times for assembling, setting up the preconditioners, and for solving the linear saddle point problems, thus representing the complete time for solving the considered problem, which is the relevant time in applications.

5.2. Steady-state flows around a cylinder

For both, 2d and 3d, there is a prescribed parabolic inflow at the left-hand side of the channel and outflow boundary conditions were used at the right-hand side, for details see [33, Test cases 2D-1, 3D-1Z]. The coarsest grids (level 0) used in our simulations are presented in Fig. 2.

The Picard iteration was started with velocity zero for all degrees of freedom, only Dirichlet nodes were set to their appropriate values. An important control mechanism for the efficiency of the iterative solvers is the accuracy required for FGMRES in each step of the Picard iteration. In our experience, it is sufficient to compute only an approximate solution of the linear saddle point problem (5) before going to the next Picard step. Thus, we prescribed the termination of FGMRES after having reduced the Euclidean norm of the residual vector by the factor 10. In addition, for the multigrid methods, at most 10 iterations should be performed. It turned out that the LSC-type methods required more iterations. In numerical studies, it was found that FGMRES(50), where in parentheses the restart parameter is given, with 100 iterations was

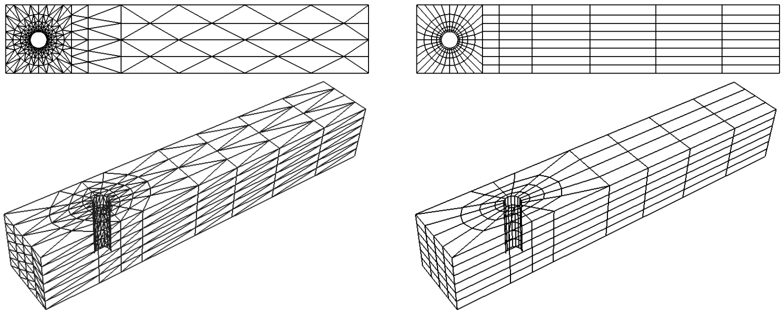

Figure 2: Initial grids, level 0.

Table 1: Steady-state flow around a cylinder, damping parameters used in the multigrid methods. No entry means that the combination of smoother and finite element spaces were not considered, see Section 3.2 for explanations.

		P_2/P_1	Q_2/Q_1	$P_2^{\mathrm{bubble}}/P_1^{\mathrm{disc}}$	Q_2/P_1^{disc}
MG(cell)	2d			0.7	0.9
	3d			0.8	0.9
MDML(cell)	2d			0.7	0.9
	3d			0.8	0.9
MG(nodal)	2d	0.8	0.7		
	3d	0.8	0.6		
MDML(nodal)	2d	0.8	0.7		
	3d	0.8	0.8		
MG(patch)	2d	0.9	0.9		
	3d	0.9	0.9		
MDML(patch)	2d	0.9	0.9		
	3d	0.9	0.9		

an appropriate choice. One approach that was studied used the direct solver UMFPACK for all linear systems of the LSC-type preconditioners. Note that the factorization of both system matrices in the LSC-type iterations needs to be computed only in the first FGMRES iteration. Moreover, the factorization of the pressure matrix $BD_v^{-1}B^T$ has to be computed even only once at the beginning of the Picard iteration. As an alternative, LSC-type preconditioners with iterative solvers for the velocity system with matrix A were tested. The only approach that resulted sometimes in a convergent Picard iteration was to use a routine provided by the library PETSc [1, 2, 3], namely FGMRES with the Boomer AMG preconditioner using the same flags as in [39]. But even then, the simulations were not more efficient than with the direct solver, see Fig. 5 for a representative result. In addition, this approach failed usually for P_2/P_1^{disc} in 2d and on the hexahedral grids in 3d. In our experience, the multigrid F-cycle is a good compromise between efficiency and stability and thus, the multigrid

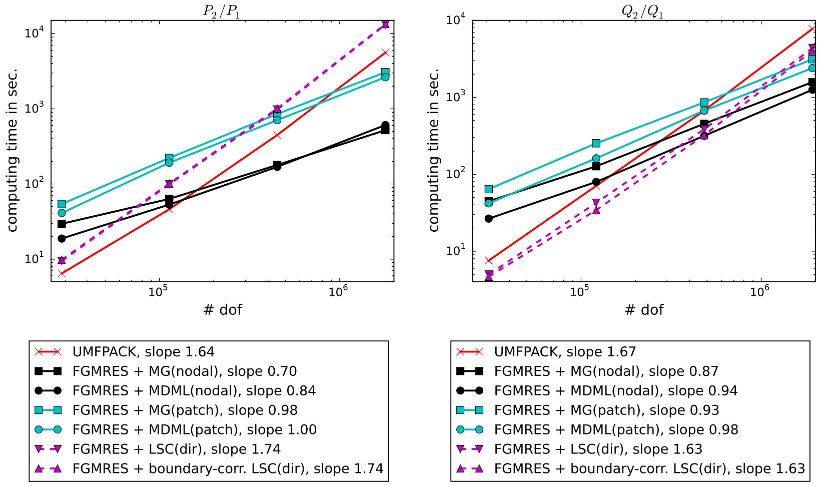

Figure 3: Steady-state flow around a cylinder in 2d: computing times and slope of best-fit line for continuous pressure approximations, first solution level is level 2.

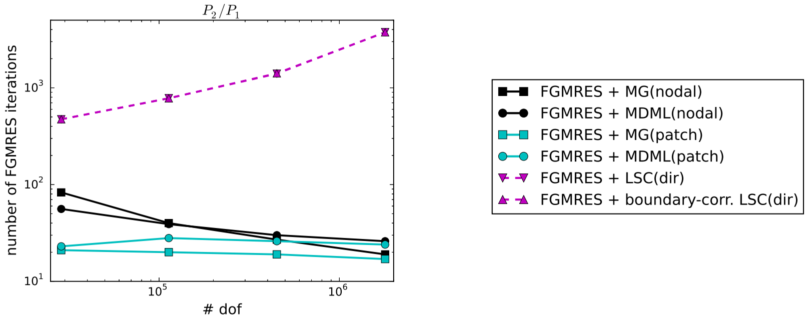

Figure 4: Steady-state flow around a cylinder in 2d: total number of FGMRES iterations, first solution level is level 2.

methods were used with the F(2,2) cycle. Table 1 summarizes the used damping parameters.

Results for the steady-state 2d flow are presented in Figs. 3 – 5 and for the 3d flow in Figs. 6 – 8. It can be seen at first glance that there is no solver which performed best in all situations.

First, the 2d results are evaluated. For discretizations with continuous pressure, Fig. 3, the direct solver and the LSC-type preconditioners were most efficient on coarser grids whereas the multigrid approaches showed a superior efficiency on finer grids. The MDML approach was generally a little bit faster than the standard multigrid scheme. Using the nodal Vanka smoother was more efficient than applying the patch Vanka smoother. In Fig. 4, it can be seen that the number of necessary FGMRES iterations for the patch Vanka smoother

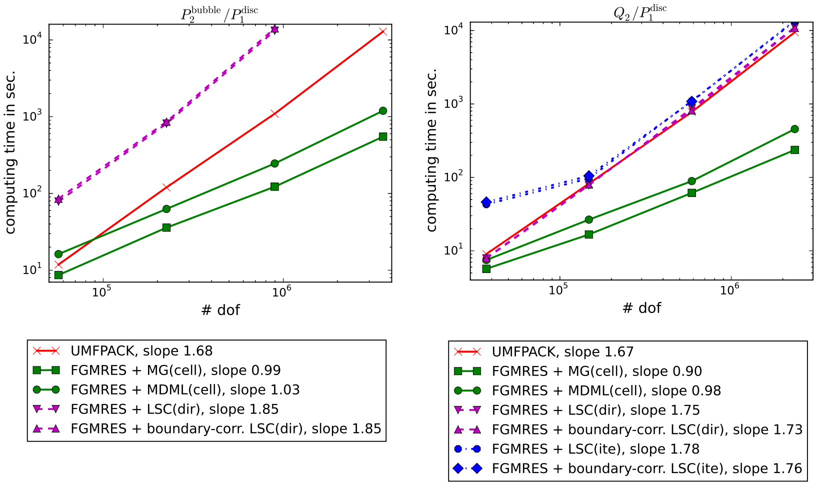

Figure 5: Steady-state flow around a cylinder in 2d: computing times and slope of best-fit line for discontinuous pressure approximations, first solution level is level 2.

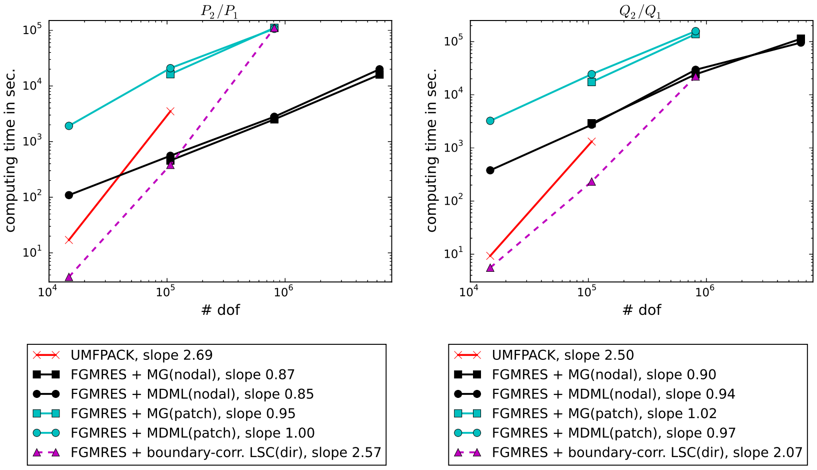

Figure 6: Steady-state flow around a cylinder in 3d: computing times and slope of best-fit line for continuous pressure approximations, first solution level is level 0.

was smaller than for the nodal Vanka smoother. However, the costs for each smoothing step were larger for the patch Vanka smoother. For all multigrid approaches, the number of necessary FGMRES iterations did not increase if the grid was refined, Fig. 4. In contrast, these numbers increased considerably for the LSC-type preconditioners.

The situation is somewhat different for the discretizations with discontinuous pressure approximation, Fig. 5. In these cases, the multigrid approaches with

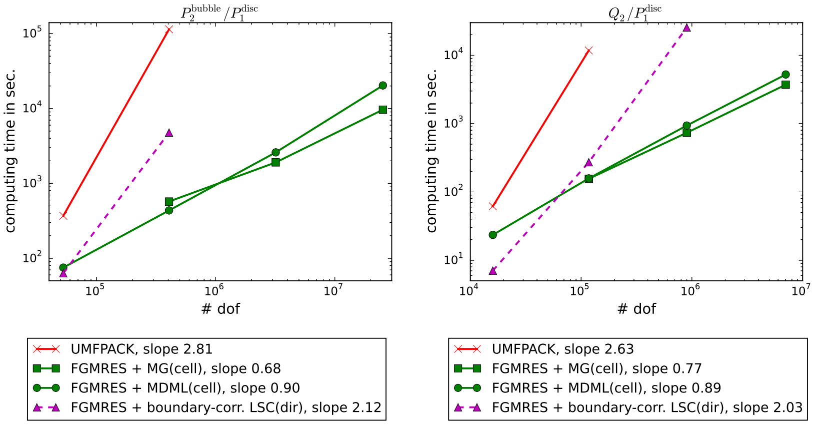

Figure 7: Steady-state flow around a cylinder in 3d: computing times and slope of best-fit line for discontinuous pressure approximations, first solution level is level 0.

the cell Vanka smoother were most efficient on all considered levels, with the standard method being more efficient than the MDML approach. The LSC-type schemes performed considerably worse compared with the discretizations with continuous pressure.

The LSC and boundary-corrected LSC preconditioner behaved always very similarly.

Based on the ansatz

$$\text{computing time} = C(\text{number of dofs})^{\alpha},$$

the power α was computed with a linear regression (best-fit line in the double logarithmic plots). It can be seen that α is around 1 for all multigrid approaches, thus showing the desired linear dependency. For the sparse direct solver and the LSC-type schemes, α is larger than 1.6. Since the LSC-type methods used a sparse direct solver for the arising linear subproblems, a similar behavior can be expected for both approaches.

The evaluation of the results for the 3d problem arrives often at the same conclusions as for the 2d problem. The systems on the finest grids could be solved only with multigrid preconditioners, whereas on coarser grids, the LSC-type methods were often more efficient. The latter situation is most notable for Q_2/Q_1 in Fig. 6. Again, the results for both LSC-type methods were very similar such that only those for the boundary-corrected variant are presented. In 3d, the patch Vanka smoother is considerably more expensive than the nodal Vanka smoother. For all multigrid methods, there were only small differences between the standard and the MDML approach, with the standard approach often a little bit more efficient. Concerning two-level methods (level 0 for MDML and level 1 for MG), we could observe that often the times spent on the coarsest grid were negligible. The only notable exceptions were the cell Vanka methods, where sometimes around half of the computing time was taken for the coarsest

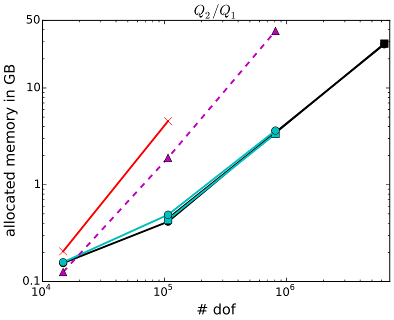

Figure 8: Steady-state flow around a cylinder in 3d: Peak resident set size (RSS) in main memory during entire computation, same legend as in Fig. 6.

grid. The reason of this high percentage is that the cell Vanka smoother used considerably less time on the finer grid than the other smoothers. Whereas the multigrid methods still showed an approximately linear relation between the number of degrees of freedom and computing time, the order of increase was in 3d even higher, compared with 2d, for UMFPACK (by around one order) and the LSC-type preconditioners (often by half an order).

A representative observation for the memory requirements of the methods is presented in Fig. 8. One can observe that all multigrid approaches needed more or less the same amount of memory whereas the other methods required considerably more memory, with FGMRES + LSC(dir) needing a smaller amount than UMFPACK.

We also examined the option of storing the local Vanka systems, see the end of Section 3.2, where one expects a gain in speed and higher memory requirements. In investigations of the stationary 2d problem, the P_2/P_1 pair of finite element spaces, and FGMRES + MG(nodal), the computation sped up by factor 4 on level 2. The same problem on the finest grid (level 5) could be solved 2.5 times faster. The memory requirement on level 2 was two times as much as in the basic version of the algorithm. On level 5, it was already three times as much. For 3d, experience for the Q_2/P_{disc}^1 discretization with FGMRES + MG(cell) is reported. There we found that the trade-off between memory consumption and speed gain improved somewhat with refinement. Storing the local systems led to a speed-up of factor 1.5 on level 1, while it was factor 2 on refinement levels 2 and 3. The memory requirements behaved roughly the same. One can conclude that storing the local systems can offer a valuable reduction of computing times, but it should only be applied, if sufficient RAM is available and memory is a minor concern.

5.3. Time-dependent flows around a cylinder

Similar to the steady-state problem, there is a prescribed parabolic inflow at the left-hand side of the channel and outflow boundary conditions were applied at the right-hand side. In 2d, a steady-state inflow was used, test case 2D-2

Table 2: Time-dependent flow around a cylinder, damping parameters and cycle type used in the multigrid methods. The combinations of smoother and finite element spaces without entries were not considered, see Section 3.2 for explanations.

		P_2/P_1	Q_2/Q_1	$P_2^{\text{bubble}}/P_1^{\text{disc}}$	Q_2/P_1^{disc}
MG(cell)	2d			F(1,1): 0.9	F(2,2): 0.9
	3d			F(2,2): 0.9	F(1,1): 0.9
MDML(cell)	2d			F(1,1): 0.9	F(2,2): 0.8
	3d			F(1,1): 0.9	F(2,2): 0.8
MG(nodal, F(1,1))	2d	0.6	0.7		
	3d	0.9	0.9		
MDML(nodal, F(1,1))	2d	0.7	0.7		
	3d	0.9	0.9		
MG(patch, F(1,1))	2d	0.8	0.8		
	3d	0.9	0.7		
MDML(patch, F(1,1))	2d	0.9	0.8		
	3d	0.7	0.9		

in [33], which leads to a Kármán vortex street. Since one period of this vortex street is approximately 0.34 s, the final time in our simulations was set to be $T = 0.34$. The initial solutions were precomputed, fully developed flow fields. The corresponding 3d problem in [33], test case 3D-2Z, does not lead to a time-dependent flow. For our simulations, we computed the steady-state solutions as initial conditions. Then, the original steady-state inflow was scaled with

$$\frac{1}{2}\sin\left(2\pi\left(2t-\frac{1}{4}\right)\right)+\frac{3}{2}, \quad 0 \le t \le T = \frac{1}{2},$$

such that a time-dependent flow occurs due to the time-dependent inflow condition.

A goal of the simulations was to study the impact of the time step on the computing times. With respect to the spatial resolution, grids were considered on which all solvers behaved reasonably well for the steady-state problems. With respect to controlling the Picard iteration and the FGMRES method, essentially the same strategy was used as for the steady-state problems, in particular the inexact solution of the linear system in each Picard iteration. The only difference was that the maximal number of FGMRES iterations for the multigrid preconditioners was set to be 5. For these preconditioners, both the F(1,1)- and the F(2,2)-cycle were studied. The results for the more efficient approach are presented below and the other one are commented briefly. Again, the application of damping was essential for the efficiency of the multigrid preconditioners and the used damping parameters for the more efficient type of cycle are presented in Table 2. For the LSC-type methods, as well the solution of both systems with UMFPACK as the solution of the system with the matrix A with the iterative method BiCGStab [36] with SSOR preconditioner ($\omega = 1$) was investigated. Since in the second approach it is inefficient to solve the sys-

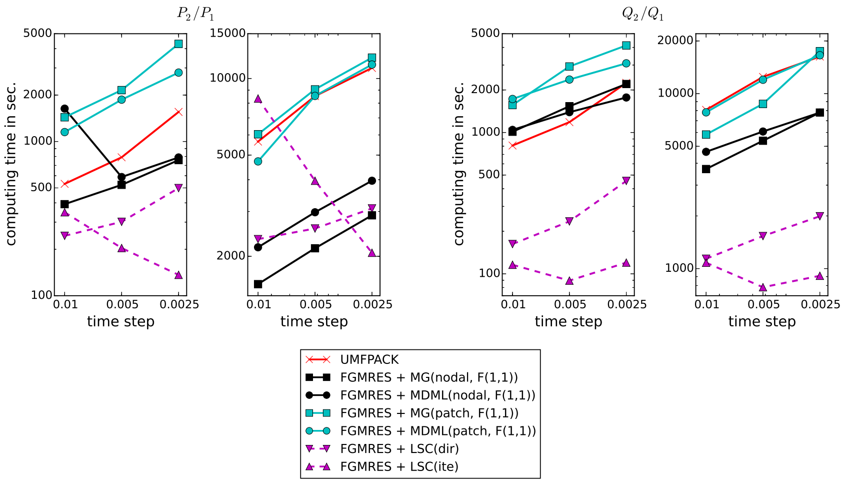

Figure 9: Time-dependent flow around a cylinder in 2d: computing times on level 3 (left pictures) and level 4 (right pictures). The numbers of degrees of freedom in space correspond to the second and third marker from left in Fig. 3, respectively.

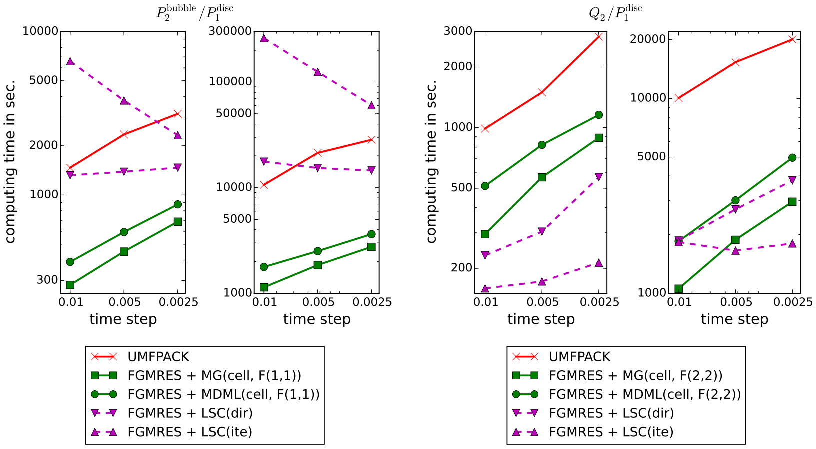

Figure 10: Time-dependent flow around a cylinder in 2d: computing times on level 3 (left pictures) and level 4 (right pictures). The numbers of degrees of freedom in space correspond to the second and third marker from left in Fig. 5, respectively.

tem with A very accurately, one needs an appropriate stopping criterion. Some numerical tests showed that terminating the BiCGStab iteration after having reduced the Euclidean norm of the residual vector by the factor 100 worked often well for the considered problems. In addition, at most 1000 BiCGStab iterations were performed.

Figures 9 – 14 present the computing times and the total number of FGMRES iterations for the different solvers. Again, the LSC and boundary-corrected

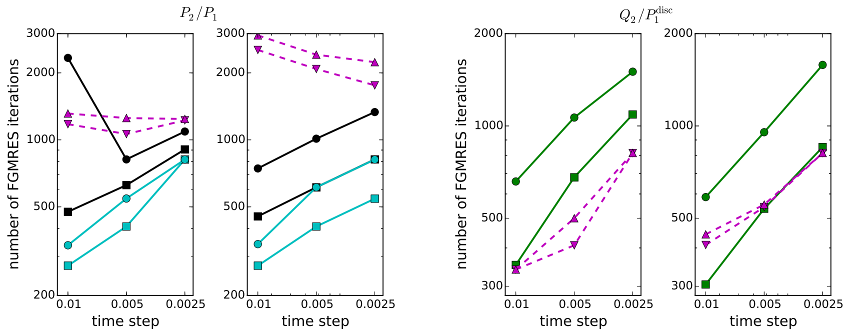

Figure 11: Time-dependent flow around a cylinder in 2d: total number of FGMRES iterations on level 3 (left pictures) and level 4 (right pictures). Qualitatively, the same behavior as for P_2/P_1 was observed for $P_2^{\rm bubble}/P_1^{\rm disc}$ and for Q_2/Q_1 as for $Q_2/P_1^{\rm disc}$, same legends as in Figs. 9 and 10.

LSC preconditioners behaved very similarly, such that only the results for the first one are shown. First, the results for the 2d time-dependent problem, Figs. 9 – 11, are discussed. One can see at first glance that there is only one solver that profits from smaller time steps and the dominance of the mass matrix, compare Section 2.3: FGMRES with the LSC-type preconditioner and the iterative solution of the system with matrix A. A reason is that the number of BiCGStab iterations decreased considerably if the time step became smaller. In addition, the expensive setup and factorization of the Poisson-type problems had to be performed only in the first Picard iteration of the initial time step. The more time steps were computed, the less was the impact of the first time step on the total computing time. On the triangular grids, even the total number of FGMRES iterations decreased slightly when the time step was reduced. FGMRES + LSC(ite) was the most efficient solver for the smallest time step in almost all cases, save for $P_2^{\rm bubble}/P_1^{\rm disc}$. For discretizations with continuous pressure, Fig. 9, the LSC preconditioner performed usually well with the direct solver for larger time steps and the iterative solver for smaller time steps. Only for P_2/P_1 on level 4 and for large time steps, the multigrid preconditioner with F(1,1)-cycle and nodal Vanka smoother was more efficient. For both, the nodal and the patch Vanka smoother, the F(1,1)-cycle was generally considerably faster than the F(2,2)-cycle. Generally, the standard multigrid and the MDML approaches needed computing times of the same order. Figures 9 and 11 contain a result which illustrates that the choice of an appropriate damping parameter in the multigrid methods might depend on the time step. For FGMRES + MDML(nodal), the used damping parameter was too large for $\Delta t = 0.01$ but it was appropriate for smaller time steps. Regarding the discretizations with discontinuous pressure spaces, the situation is somewhat different. For the $P_2^{\rm bubble}/P_1^{\rm disc}$ pair, the standard multigrid method with the cell Vanka smoother was always the most efficient approach. The LSC preconditioner with the iterative solution of the system with matrix A was often the best performing method

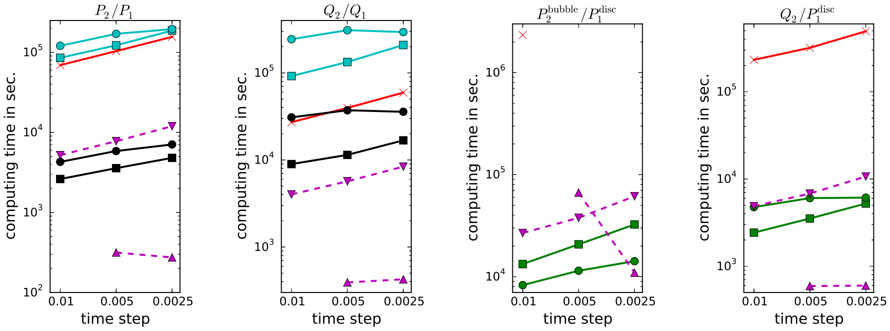

Figure 12: Time-dependent flow around a cylinder in 3d: computing times, same legends as in Figs. 9 and 10, multigrid cycles given in Table 2. The LSC(ite) preconditioner blew up for $\Delta t = 0.01$. The number of degrees of freedom in space corresponds to the second marker from left in Figs. 6 and 7.

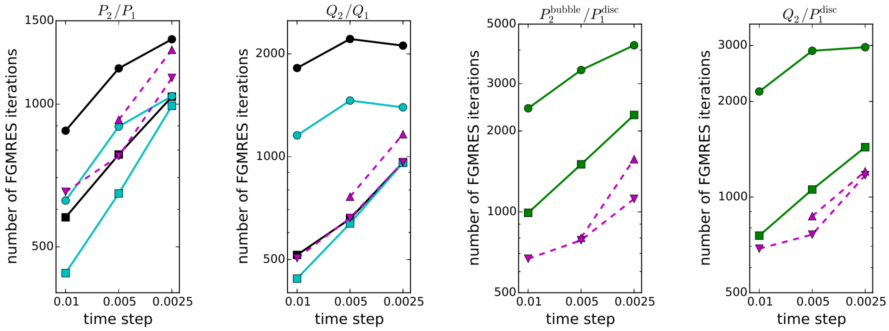

Figure 13: Time-dependent flow around a cylinder in 3d: total number of FGMRES iterations, same legends as in Figs. 9 and 10, multigrid cycles given in Table 2. The LSC(ite) preconditioner blew up for $\Delta t = 0.01$.

for the Q_2/P_1^{disc} pair. Concerning the total number of FGMRES iterations, one can often observe a linear increase with a factor smaller than two for all multigrid preconditioners if the time step is halved, compare Fig. 11. Using the standard multigrid approach required less iterations than applying the MDML method.

Simulations of the 3d time-dependent problem were performed on level 1. Hence, the standard multigrid method is just a two-level method. The results presented in Fig. 12 show that FGMRES + LSC(ite) was generally the most efficient method for smaller steps $\Delta t \in \{0.005, 0.0025\}$. Often, it was by far faster than the other methods. For larger time steps, FGMRES with one of the multigrid preconditioners were best. But altogether, it was generally much more efficient to apply FGMRES + LSC(ite) with a small time step than to use the multigrid preconditioner with a large time step. In addition, apart from P_2/P_1, the total number of FGMRES iterations for FGMRES + LSC(ite) was

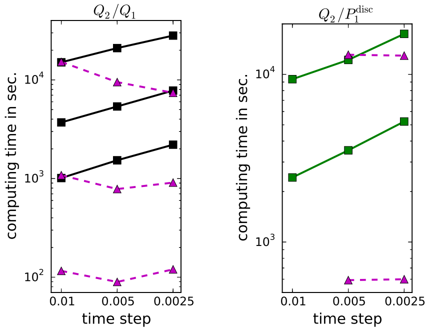

Figure 14: Time-dependent flow around a cylinder in 2d (left) and 3d (right): computing times with the LSC-type preconditioner and coupled multigrid preconditioners. 2d: level 3 (lowest curves), level 4 (middle curves), and level 5 (upper curves); 3d: level 1 (lower curves) and level 2 (upper curves); same legends as in Figs. 9 and 10. The number of degrees of freedom in space corresponds to the second to fourth marker from left in Fig. 3 (2d) and to the second and third marker from left in Fig. 7 (3d).

comparatively small, see Fig. 13. The sparse direct solver and the multigrid approaches with patch Vanka smoother were in all studies not competitive. For the cell Vanka smoother, the computing times with the F(1,1)- and F(2,2)-cycle were of the same order, whereas for the nodal and patch Vanka smoothers, the use of the F(1,1) cycle was much more efficient.

Figure 14 presents studies of the LSC preconditioner and multigrid preconditioners which include an additional refinement in space. It can be observed that, on the one hand, the superiority of the LSC preconditioner became smaller with increasing spatial refinement. But on the other hand, this preconditioner was still more efficient than the multigrid preconditioners, in particular for small time steps.

The memory requirements on level 1 were very similar to the steady-state case, compare Fig. 8. The method FGMRES + LSC(ite) needed around four times the memory of the multigrid methods but less than half times the memory of UMFPACK.

Finally, a brief summary of our experience with IMEX schemes is given. In these schemes, only one linear saddle point problem in each discrete time has to be solved, but this system has to be solved accurately. The only one of the studied solvers that really takes advantage of this fact is UMFPACK. For the considered examples, the solver times with UMFPACK were reduced by a factor of 4-4.5 for the largest time step and 2-2.5 for the smallest time step if an IMEX scheme was applied. These reductions are less than an order of magnitude. Concerning the iterative methods, we have the experience that there is not that much difference between the computing times for solving one linear saddle point problem accurately compared with solving several systems inexactly (and assembling in between new matrix blocks A), as it is done in our numerical studies. Altogether, there is no qualitative difference of the computing times of the fully implicit and an IMEX approach for the considered examples.

6. Some remarks on the application of the methods on parallel computers

The numerical studies presented in this paper were restricted to the serial case. First, we think that this case is still of interest for many users. And second, to extend the numerical studies by incorporating the behavior of the considered methods with respect to their performance on parallel computers would be beyond the scope of a single paper. However, problems coming from applications are simulated nowadays often on parallel computers. For this reason, we like to provide here at least a discussion on some aspects of the parallel implementation of the methods and on the available numerical experience.

A parallel implementation of the coupled multigrid preconditioners is available in ParMooN, see [16] for details of the implementation. In this implementation, the Vanka-type smoothers depend on the distribution of the mesh cells to the processors. For each processor, the block Gauss–Seidel Vanka smoothers as described in Section 3.2 are applied. Then, the values at the degrees of freedom at the processor interfaces are computed as an arithmetic average of the contributions from all processors sharing this interface. In this way, the smoother becomes an outer Jacobi-type method (with respect to the processors) with an inner block Gauss–Seidel iteration. The coarsest grid is available on all processors such that an iterative method or a direct solver, applied simultaneously on all processors, can be applied. In [39], numerical results are reported for the 3d steady-state flow around a cylinder, which is described in Section 5.2, and the Q_2/P_1^{disc} pair of finite element spaces. On the finest grid, which corresponds to the rightmost marker in the right picture of Fig. 7, a fairly good strong scalability of around 66 % was observed up to 20 processors.

A parallel version of the LSC preconditioner is provided by the library PETSc, [1, 2, 3]. This preconditioner was applied in the numerical studies of [39] for the solution of the 3d steady-state flow around a cylinder from Section 5.2 and in [38], where the flow in a helically coiled tube was simulated. In [39], several options to call this preconditioner were tested, but only an approach with a direct solution of the arising linear systems of equations worked. However, it was much more inefficient than the parallelized coupled multigrid method. On the second finest grid, second marker from the right in the right picture of Fig. 7, a reasonable scaling was observed up to 12 processors. It could be seen in Section 5.3 that the costs of the initial factorization of the matrix $BD_v^{-1}B^T$ are more than compensated if sufficiently many time steps have to be performed. It is an open question up to which number of processors the building of $BD_v^{-1}B^T$ and the direct solution of the arising system is feasible. A Krylov subspace method, which requires only the multiplication of this matrix with vectors, and a Jacobi-type preconditioner, which needs only the explicit computation of the diagonal entries of this matrix, might be an efficient option for a large number of processors.

The parallel implementation and performance of the modified Augmented Lagrangian preconditioner proposed in [6] is studied in [8]. Several steady-state examples in 2d and 3d were considered. A good strong parallel scaling was observed generally up to 32 processors, of course depending on the problem size.

As already mentioned in the introduction, splitting schemes are an alternative to coupled approaches for the solution of time-dependent incompressible flow problems. This alternative might be particularly attractive from the point of view of parallel scalability, since highly parallelized splitting schemes have been already proposed in the literature, e.g., in [28].

In summary, there are some experiences with the available preconditioners on parallel computers. But these are restricted to a rather small number of processors.

7. Summary and Outlook

This paper studied a number of solvers for linear saddle point problems that arise in the linearization and discretization of the incompressible Navier–Stokes equations using inf-sup stable second order velocity and first order pressure finite element spaces. It could be seen that the efficiency of the solvers depends on the concrete pair of spaces, the fineness of the spatial mesh, and the length of the time step. The most important conclusions of these studies are as follows. For steady-state problems on fine grids, in particular in 3d, FGMRES with an appropriate coupled multigrid preconditioner (nodal Vanka for continuous pressure, cell Vanka for discontinuous pressure) was generally the most efficient approach. The simulation of time-dependent problems, discretized with sufficiently small time steps, could be performed fastest with FGMRES and the LSC-type preconditioners with an iterative and inexact solution of the velocity subproblem. In almost all situations, the use of an appropriate iterative solver was more efficient, often even by orders of magnitude, than the application of the sparse direct solver UMFPACK. Further findings include first that the computing times of the standard multigrid and the MDML preconditioner were generally similar and second that the patch Vanka smoother for discretizations with continuous finite element pressure is not competitive to the nodal Vanka smoother.

The presented numerical studies cover just the basic setup: the Galerkin discretization of laminar flow problems simulated in a sequential way. The behavior of the solvers with respect to the size of the viscosity coefficient (or equivalently to the Reynolds number) is a topic of future research, in particular for small ν. Very small values of ν lead to turbulent flows whose simulation requires the use of a turbulence model. Such models might change the block structure of the matrix A from block-diagonal to full. So far, coupled multigrid preconditioners have been used by ourselves for performing turbulent flow simulations, e.g., in [25]. It is an open question whether LSC-type or Augmented Lagrangian-type preconditioners could be more efficient in this case. As already discussed in Section 6, there are only few experiences and several open questions concerning the performance and algorithmic options of the considered preconditioners on parallel computers.

Acknowledgment. We would like to thank two anonymous referees whose suggestions helped us to improve this paper.

References

[1] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2016.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[3] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[4] Michele Benzi and Maxim A. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. *SIAM J. Sci. Comput.*, 28(6):2095–2113, 2006.

[5] Michele Benzi and Maxim A. Olshanskii. Field-of-values convergence analysis of augmented Lagrangian preconditioners for the linearized Navier-Stokes problem. *SIAM J. Numer. Anal.*, 49(2):770–788, 2011.

[6] Michele Benzi, Maxim A. Olshanskii, and Zhen Wang. Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 66(4):486–508, 2011.

[7] Michele Benzi and Zhen Wang. Analysis of augmented Lagrangian-based preconditioners for the steady incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 33(5):2761–2784, 2011.

[8] Michele Benzi and Zhen Wang. A parallel implementation of the modified augmented Lagrangian preconditioner for the incompressible Navier-Stokes equations. *Numer. Algorithms*, 64(1):73–84, 2013.

[9] M. Crouzeix and P.-A. Raviart. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. I. *Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge*, 7(R-3):33–75, 1973.

[10] Timothy A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software*, 30(2):196–199, 2004.

[11] Howard Elman, V. E. Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 227(3):1790–1808, 2008.

[12] Howard Elman, Victoria E. Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro. Block preconditioners based on approximate commutators. *SIAM J. Sci. Comput.*, 27(5):1651–1668, 2006.

[13] Howard Elman, Victoria E. Howle, John Shadid, David Silvester, and Ray Tuminaro. Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations. *SIAM J. Sci. Comput.*, 30(1):290–311, 2007/08.

[14] Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, second edition, 2014.

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

[16] S. Ganesan, V. John, G. Matthies, R. Meesala, S. Abdus, and U. Wilbrandt. An object oriented parallel finite element scheme for computing pdes: Design and implementation. In *IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW) Hyderabad*, pages 106–115. IEEE, 2016.

[17] J. L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Comput. Methods Appl. Mech. Engrg.*, 195(44-47):6011–6045, 2006.

[18] Xin He, Maya Neytcheva, and Stefano Serra Capizzano. On an augmented Lagrangian-based preconditioning of Oseen type problems. *BIT*, 51(4):865–888, 2011.

[19] Timo Heister and Gerd Rapin. Efficient augmented Lagrangian-type preconditioning for the Oseen problem using grad-div stabilization. *Internat. J. Numer. Methods Fluids*, 71(1):118–134, 2013.

[20] Ross Ingram. A new linearly extrapolated Crank-Nicolson time-stepping scheme for the Navier-Stokes equations. *Math. Comp.*, 82(284):1953–1973, 2013.

[21] V. John, P. Knobloch, G. Matthies, and L. Tobiska. Non-nested multi-level solvers for finite element discretisations of mixed problems. *Computing*, 68(4):313–341, 2002.

[22] Volker John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3D Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 40(6):775–798, 2002.

[23] Volker John. On the efficiency of linearization schemes and coupled multigrid methods in the simulation of a 3D flow around a cylinder. *Internat. J. Numer. Methods Fluids*, 50(7):845–862, 2006.

[24] Volker John. *Finite Element Methods for Incompressible Flow Problems*, volume 51 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2016.

[25] Volker John and Adela Kindl. A variational multiscale method for turbulent flow simulation with adaptive large scale space. *J. Comput. Phys.*, 229(2):301–312, 2010.

[26] Volker John and Gunar Matthies. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *Internat. J. Numer. Methods Fluids*, 37(8):885–903, 2001.

[27] David Kay, Daniel Loghin, and Andrew Wathen. A preconditioner for the steady-state Navier-Stokes equations. *SIAM J. Sci. Comput.*, 24(1):237–256, 2002.

[28] Benjamin Krank, Niklas Fehn, Wolfgang A. Wall, and Martin Kronbichler. A high-order semi-explicit discontinuous Galerkin solver for 3d incompressible flow with application to DNS and LES of turbulent channel flow. Technical Report 1607.01323, arXiv, 2016.

[29] Sabine Le Borne and Leo G. Rebholz. Preconditioning sparse grad-div/augmented Lagrangian stabilized saddle point systems. *Comput. Vis. Sci.*, 16(6):259–269, 2013.

[30] Maxim A. Olshanskii and Yuri V. Vassilevski. Pressure Schur complement preconditioners for the discrete Oseen problem. *SIAM J. Sci. Comput.*, 29(6):2686–2704, 2007.

[31] R. Rannacher and S. Turek. Simple nonconforming quadrilateral Stokes element. *Numer. Methods Partial Differential Equations*, 8(2):97–111, 1992.

[32] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[33] M. Schäfer and S. Turek. Benchmark computations of laminar flow around a cylinder. (With support by F. Durst, E. Krause and R. Rannacher). In *Flow simulation with high-performance computers II. DFG priority research programme results 1993 - 1995*, pages 547–566. Wiesbaden: Vieweg, 1996.

[34] F. Schieweck. A general transfer operator for arbitrary finite element spaces. Preprint 00-25 172, Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, 2000.

[35] A. Segal, M. ur Rehman, and C. Vuik. Preconditioners for incompressible Navier-Stokes solvers. *Numer. Math. Theory Methods Appl.*, 3(3):245–275, 2010.

[36] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[37] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *J. Comput. Phys.*, 65(1):138–158, 1986.

[38] Viktoria Wiedmeyer, Felix Anker, Clemens Bartsch, Andreas Voigt, Volker John, and Kai Sundmacher. Continuous crystallization in a helically coiled flow tube: Analysis of flow field, residence time behavior, and crystal growth. *Industrial & Engineering Chemistry Research*, 56(13):3699–3712, 2017.

[39] Ulrich Wilbrandt, Clemens Bartsch, Naveed Ahmed, Najib Alia, Felix Anker, Laura Blank, Alfonso Caiazzo, Sashikumaar Ganesan, Swetlana Giere, Gunar Matthies, Raviteja Meesala, Abdus Shamim, Jagannath Venkatesan, and Volker John. ParMooN—A modernized program package based on mapped finite elements. *Comput. Math. Appl.*, 74(1):74–88, 2017.