



# A low-rank update for relaxed Schur complement preconditioners in fluid flow problems

Rebekka S. Beddig<sup>1</sup> · Jörn Behrens<sup>2</sup> · Sabine Le Borne<sup>1</sup>

Received: 18 January 2023 / Accepted: 26 March 2023

© The Author(s) 2023

## Abstract

The simulation of fluid dynamic problems often involves solving large-scale saddle-point systems. Their numerical solution with iterative solvers requires efficient preconditioners. Low-rank updates can adapt standard preconditioners to accelerate their convergence. We consider a multiplicative low-rank correction for pressure Schur complement preconditioners that is based on a (randomized) low-rank approximation of the error between the identity and the preconditioned Schur complement. We further introduce a relaxation parameter that scales the initial preconditioner. This parameter can improve the initial preconditioner as well as the update scheme. We provide an error analysis for the described update method. Numerical results for the linearized Navier–Stokes equations in a model for atmospheric dynamics on two different geometries illustrate the action of the update scheme. We numerically analyze various parameters of the low-rank update with respect to their influence on convergence and computational time.

**Keywords** Preconditioner · Saddle-point system · Low-rank update · Schur complement

**Mathematics Subject Classification (2010)** 65F08 · 65F10 · 65N22 · 65F55

## 1 Introduction

Large-scale linear saddle-point systems arise in computational fluid dynamics. A stable discretization of the incompressible time-dependent linearized Navier–Stokes equa-

---

✉ Sabine Le Borne  
leborne@tuhh.de

<sup>1</sup> Hamburg University of Technology, Institute of Mathematics, Hamburg, Germany

<sup>2</sup> Department of Mathematics, University of Hamburg, Hamburg, Germany

tions leads to systems of the form

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \quad (1)$$

where  $A \in \mathbb{R}^{n_u \times n_u}$ ,  $B \in \mathbb{R}^{n_p \times n_u}$ ,  $\mathbf{u}, \mathbf{f} \in \mathbb{R}^{n_u}$ ,  $p \in \mathbb{R}^{n_p}$ . The matrix block  $A$  is dominated by the velocity mass matrix and includes diffusion and advection, both scaled by the time step length. The matrix blocks  $B$ ,  $B^T$  correspond to the divergence and gradient operator, resp.. We solve for the fluid velocity  $\mathbf{u}$  and the pressure  $p$ . The right-hand side  $\mathbf{f}$  describes the external forcing. The obtained high-dimensional systems are solved with iterative solvers which require efficient preconditioners.

An overview of the numerical solution of saddle-point problems arising in fluid flow applications and suitable preconditioners are given in [1] or [2]. A typical choice for preconditioners of  $2 \times 2$ -block systems as given in (1) is based on (approximations of) their block LU factorizations. Such block preconditioners typically require approximations to the inverse of the upper left block  $A$  of the system matrix in (1) as well as the (negative) pressure Schur complement  $S := BA^{-1}B^T$ . Well-known preconditioning techniques for the Schur complement are SIMPLE-type preconditioners [3, 4], the least-squares commutator (LSC) [2, 5], the pressure-convection-diffusion commutator [6], grad-div preconditioners [7–9], augmented Lagrangian preconditioners [4, 10, 11] as well as hierarchical matrix preconditioners [12]. An alternative approach that is not based on the block LU factorization is given by the so-called nullspace method [13].

Any given preconditioner can be modified by low-rank corrections to hopefully improve its performance in an iterative solver. Typical objectives are to modify certain spectral properties of the preconditioned matrix. Large eigenvalues but also small eigenvalues (close to the origin) may decelerate the convergence of iterative methods. Deflation strategies aim to shift small eigenvalues to zero to make them “invisible” to the iterative solver. Spectral preconditioners aim at shifting small eigenvalues by one. Tuning strategies aim to shift large eigenvalues to one. An overview of popular low-rank techniques is given in the survey [14]. In [15], a low-rank update is applied to the velocity-related block  $A$  of the system matrix but not to the Schur complement. In [16–18], low-rank techniques are applied to precondition the Schur complement that arises in domain decomposition methods.

The construction of low-rank corrections may pertain to matrices that are not given in explicit form but only defined by their action on a vector. In [15, 16], Arnoldi iterations are used to compute the low-rank approximations. Another approach are methods from randomized numerical linear algebra. In [17], the Nystrom method is used which is suitable for symmetric positive definite matrices. In [19, 20], a randomized low-rank approximation is obtained by approximating the range of a matrix by sampling with a few random vectors.

The motivation for this work is to analyze whether a certain type of multiplicative low-rank update that has been recently introduced in [16] in the context of domain decomposition methods can also be used to improve existing Schur complement preconditioners in block preconditioners for the saddle-point system in (1). The update in [16] is based on a low-rank approximation of the error between the identity matrix

and the preconditioned pressure Schur complement. Our conclusion will be that the impact of such a low-rank update on the convergence of the iterative method differs not only for different model problems but also for different parameters that are relevant for the construction of the low-rank updates.

In particular, we will consider two different initial Schur complement preconditioners, two different schemes to compute low-rank approximations, and two different model problems all of which we will briefly outline in the following.

The (negative) Schur complement  $S = BA^{-1}B^T$  is not available explicitly since  $A^{-1}$  is typically only available in the form of (approximate) matrix–vector multiplications and an explicit  $S$  would result in a dense matrix. We consider two different preconditioning approaches as initial preconditioners: (i) the SIMPLE preconditioner [3] which uses an approximation of the Schur complement that replaces  $A^{-1}$  by a diagonal matrix, and (ii) the least-squares commutator (LSC) preconditioner [5] which provides an approximation to the Schur complement inverse based on approximate algebraic commuting (i.e., without first computing the explicit Schur complement).

In order to compute low-rank corrections for these initial Schur complement preconditioners, we use the following two methods: (i) Arnoldi iteration, and (ii) a randomized power range finder [19].

Our two model problems both stem from a model for buoyancy-driven atmospheric dynamics and differ in the computational domains which consist of a three-dimensional cube as well as a shell.

A further novel component of our work is the introduction of a relaxation parameter that scales the initial preconditioner before the computation of the update. It turns out that such a relaxation parameter can have a substantial impact on the convergence of the iterative solver.

We will provide an error analysis that connects the error of the low-rank update to the error between the identity matrix and the preconditioned Schur complement where the multiplicative low-rank update has been incorporated into the preconditioner.

In our numerical tests, we investigate the influence of the choice of initial preconditioner, relaxation parameter, low-rank update scheme and the update rank on the convergence behavior and computational times of the iterative solver.

The remainder of this paper is organized as follows: We start in Section 2 with an introduction of the initial preconditioners and the low-rank approximation methods. In Section 3, we review the update scheme introduced in [16] and derive left and right update formulas for a given initial preconditioner. This is followed by an analysis of the influence of the update on the error between the identity and the preconditioned Schur complement in Section 4. Numerical results illustrate the update method in Section 5. We conclude in Section 6 and give an outlook on future research.

## 2 Initial preconditioners and low-rank approximation

We start with a common block preconditioner and two well-known Schur complement preconditioners that we will use as initial preconditioners. Then, we will discuss two algorithms for the computation of a low-rank approximation that will be needed for the update derived in Section 3.

## 2.1 Initial preconditioners

Block preconditioners for saddle-point systems of the form in (1) can be obtained through block factorizations of the system matrix. We will use the block triangular preconditioner that is based on the block LU factorization

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I_{n_u} & 0 \\ BA^{-1} & I_{n_p} \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & -S \end{pmatrix}$$

where  $S := BA^{-1}B^T$  is the negative pressure Schur complement. The block triangular preconditioner is given by

$$P_{\text{BTP,ideal}} = \begin{pmatrix} A & B^T \\ 0 & -S \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & A^{-1}B^TS^{-1} \\ 0 & -S^{-1} \end{pmatrix}.$$

It is considered an ideal preconditioner since it can be shown that GMRES preconditioned with the block triangular preconditioner converges within at most two iterations [21, 22]. Since the inverses  $A^{-1}$ ,  $S^{-1}$  are not available we replace them by preconditioners  $\hat{A}^{-1}$ ,  $\hat{S}^{-1}$  which leads to the block triangular preconditioner

$$P_{\text{BTP}} = \begin{pmatrix} \hat{A}^{-1} & \hat{A}^{-1}B^T\hat{S}^{-1} \\ 0 & -\hat{S}^{-1} \end{pmatrix}. \quad (2)$$

The velocity block preconditioner  $\hat{A}^{-1}$  is often implemented by a multigrid method [23]. However, due to the advection in our system, algebraic multigrid methods may fail. Furthermore, the update technique presented in this paper leads to an algorithm that requires the transposed application of the preconditioners, i.e.,  $(\hat{A}^{-1})^T v$  and  $(\hat{S}^{-1})^T v$ , which is not available for multigrid techniques for unsymmetric systems. Hence, we choose an incomplete factorization  $\text{ilu}(0)$  as  $\hat{A}$ .

For the Schur complement, we use two well-known preconditioning techniques, the LSC method [5] and a SIMPLE-type method [3]. They are given by

$$S_{\text{LSC}}^{-1} = \left( BD_u^{-1}B^T \right)^{-1} BD_u^{-1}AD_u^{-1}B^T \left( BD_u^{-1}B^T \right)^{-1},$$

$$S_{\text{SIMPLE}}^{-1} = \left( B \text{diag}(A)^{-1}B^T \right)^{-1}$$

where  $D_u$  is a diagonal approximation of the velocity mass matrix. We use the diagonal of the velocity mass matrix as an approximation. We replace the Poisson-type problems by approximations  $M_{BDuBT} \approx BD_u^{-1}B^T$  and  $M_{BABT} \approx B \text{diag}(A)^{-1}B^T$

which are obtained through incomplete Cholesky factorizations  $\hat{S}^{-1}(0)$ . The resulting preconditioners are given by

$$\hat{S}_{\text{LSC}}^{-1} = M_{BDuBT}^{-1} B D_u^{-1} A D_u^{-1} B^T M_{BDuBT}^{-1}, \quad (3)$$

$$\hat{S}_{\text{SIMPLE}}^{-1} = M_{BAPT}^{-1}. \quad (4)$$

Our choice of (the components of) the initial preconditioners has been guided by the desire to use rather straightforward (standard) algorithms. In our section on numerical results, we include tests for some alternative choices such as multigrid methods for  $\hat{A}^{-1}$  and using inner iterations for  $\hat{S}^{-1}$ . These tests have led us to believe that our choices for initial preconditioners are reasonable.

## 2.2 Low-rank approximation

For the update method, we will need a low-rank approximation of a matrix that is only defined by its action on a vector. We consider two methods, a randomized algorithm that is based on approximating the range of a given matrix  $M \in \mathbb{R}^{n \times n}$  and the Arnoldi iteration.

We start with a description of the randomized algorithm. The goal is to compute a matrix  $Q \in \mathbb{R}^{n \times \ell}$ ,  $\ell \ll n$ , with as few as possible orthonormal columns that satisfies

$$M \approx QQ^T M$$

and hence yields a low-rank approximation of  $M$  in the form

$$M \approx QN^T \quad (5)$$

with  $N := M^T Q$ . We compute  $Q$  with the randomized power range finder from [19] which captures the dominant action of the given matrix  $M$  by sampling with a few random test vectors which form the columns of a test matrix  $G \in \mathbb{R}^{n \times l}$ . To improve accuracy and numerical robustness we apply  $q$  power iterations

$$Y = (MM^T)^q MG = QR$$

and obtain  $Q$  by computing the QR factorization of  $Y$ . The quality of  $Q$  can be improved by orthonormalizing the result after each multiplication with the given matrix  $M$  or its transpose. The needed steps for the randomized low-rank approximation are summarized in Algorithm 1. It can be beneficial to use more test vectors than the desired rank  $r$  for the low-rank approximation. We call the number of oversampling vectors the oversampling parameter  $k \geq 0$ . We create the  $l = r + k$  sample vectors with normally distributed, independent entries with mean zero and variance one. These sample vectors form the columns of the sample matrix  $G \in \mathbb{R}^{n \times l}$ .

According to [19], a small oversampling parameter of  $k = 5$  or  $k = 10$  is usually sufficient for Gaussian test matrices, i.e., choosing  $k > r$  usually does not improve the

approximation further. As stated in [20], a small number of power iterations ( $q \leq 3$ ) usually leads to satisfying accuracy.

---

**Algorithm 1** Randomized low-rank approximation with the power range finder from [19].

---

**Input:** Input matrix  $M \in \mathbb{R}^{n \times n}$ , rank  $r$ , subspace dimension  $l \geq r$ , number of power iterations  $q$ .  
**Output:** Low-rank approximation  $QN^T \approx M$  with  $Q \in \mathbb{R}^{n \times r}$  such that  $Q^T Q = I_r$ ,  $N \in \mathbb{R}^{r \times n}$ .  
 // Approximate the range of  $M$ .

- 1 Draw a random matrix  $G \in \mathbb{R}^{n \times l}$ ;
- 2 Compute  $Y_0 = MG$  and compute its QR factorization  $Y_0 = Q_0 R_0$ ;
- 3 **for**  $j = 1, \dots, q$  **do**
- 4     Form  $\tilde{Y}_j = M^T Q_{j-1}$  and compute its QR factorization  $\tilde{Y}_j = \tilde{Q}_j \tilde{R}_j$ ;
- 5     Form  $Y_j = M \tilde{Q}_j$  and compute its QR factorization  $Y_j = Q_j R_j$ ;
- 6 **end**
- // Compute the low-rank approximation of  $M$ .
- 7 Set  $Q = Q_q[:, 1:r]$ ;
- 8 Compute  $N = M^T Q$ ;

---

Next, we describe the computation of a low-rank approximation of a matrix  $M$  with the Arnoldi iteration. In each iteration, the matrix  $M$  is applied to a vector and the result is orthogonalized against the vectors obtained in previous iterations. After  $r$  such steps, we find a low-rank approximation of the given matrix  $M$  of the form

$$M \approx V_r H_r V_r^T$$

where  $V_r \in \mathbb{R}^{n \times r}$  has orthonormal columns and  $H_r \in \mathbb{R}^{r \times r}$  is an upper Hessenberg matrix. We use the Arnoldi iteration with modified Gram-Schmidt from [24]. We reorthogonalize the vectors if needed in the Gram-Schmidt procedure. Algorithm 2 summarizes the low-rank approximation with the Arnoldi method.

---

**Algorithm 2** Low-rank approximation with the Arnoldi method.

---

**Input:** Input matrix  $M \in \mathbb{R}^{n \times n}$ , rank  $r$ .  
**Output:** Low-rank approximation  $VHV^T \approx M$  with  $V \in \mathbb{R}^{n \times r}$  such that  $V^T V = I_r$ ,  $V \in \mathbb{R}^{n \times r}$  and  $H \in \mathbb{R}^{r \times r}$ .

- 1 Draw a random initial vector  $v_1$  of size  $n$  with norm 1.
- 2 **for**  $j = 1, \dots, r$  **do**
- 3     Compute  $y_j = M v_{j-1}$ .
- 4     **for**  $i = 1, \dots, j$  **do**
- 5         Compute  $H_{i,j} = y_j^T v_i$ .
- 6         Compute  $y_j = y_j - H_{i,j} v_i$ .
- 7     **end**
- 8     Compute  $H_{j+1,j} = |y_j|_2$ .
- 9     Compute  $v_{j+1} = y_j / H_{j+1,j}$ .
- 10 **end**
- 11 Set  $V = [v_1, v_2, \dots, v_r]$ .

---

We briefly discuss the differences between the randomized low-rank approximation and the Arnoldi method. In both methods, the computational costs are dominated by the repeated application of the given matrix to a vector. The randomized low-rank approximation requires  $(q+1)l$  applications and  $(q+1)l$  transposed applications. The Arnoldi method requires  $l$  applications and no transposed applications. So, the Arnoldi method is cheaper concerning setup times but requires the sequential application of the given matrix to  $l$  vectors. In the randomized low-rank approximation, we could apply the given matrix to  $l$  vectors in parallel.

### 3 Error-based update with relaxed initial preconditioner

In this section, we derive a left and a right preconditioner update following the ideas from [16]. We modify the updates by scaling the initial Schur complement preconditioner. Our goal is to adapt the given initial preconditioner  $\widehat{S}^{-1}$  with a low-rank update such that the new preconditioner is a more accurate approximation to the inverse Schur complement  $S^{-1}$ . To find a suitable update, we start from the error between the inverse Schur complement  $S^{-1}$  and the initial preconditioner  $\widehat{S}^{-1}$

$$\widehat{R} = S^{-1} - \widehat{S}^{-1}. \quad (6)$$

We derive left and right update schemes by multiplying (6) by  $S$  from the left and the right, respectively,

$$E_L := I_{n_p} - \widehat{S}^{-1} S, \quad E_R := I_{n_p} - S \widehat{S}^{-1}.$$

Solving for  $S$  and inverting, we obtain

$$S^{-1} = (I_{n_p} - E_L)^{-1} \widehat{S}^{-1}, \quad S^{-1} = \widehat{S}^{-1} (I_{n_p} - E_R)^{-1}. \quad (7)$$

These alternative formulas for the inverse Schur complement  $S^{-1}$  are not practical since the matrices  $E_L, E_R$  are not available and solving an additional system of size  $n_p \times n_p$  is problematic concerning computational complexity. We hence note that we are able to apply the error matrices  $E_L, E_R$  to vectors which allows us to compute low-rank approximations  $E_L \approx U_{L,r} V_{L,r}^T, E_R \approx U_{R,r} V_{R,r}^T$ , with  $U_{L,r}, V_{L,r}, U_{R,r}, V_{R,r} \in \mathbb{R}^{n_p \times r}, r \ll n_p$ , and obtain

$$S_L^{-1} = \left( I_{n_p} - U_{L,r} V_{L,r}^T \right)^{-1} \widehat{S}^{-1}, \quad S_R^{-1} = \widehat{S}^{-1} \left( I_{n_p} - U_{R,r} V_{R,r}^T \right)^{-1}.$$

By applying the Sherman-Morrison-Woodbury formula, we find

$$S_L^{-1} = \left( I_{n_p} + U_{L,r} \left( I_r - V_{L,r}^T U_{L,r} \right)^{-1} V_{L,r}^T \right) \widehat{S}^{-1},$$

$$S_R^{-1} = \widehat{S}^{-1} \left( I_{n_p} + U_{R,r} \left( I_r - V_{R,r}^T U_{R,r} \right)^{-1} V_{R,r}^T \right).$$

These approximations  $S_L^{-1}$ ,  $S_R^{-1}$  to the inverse Schur complement  $S^{-1}$  only require the additional solution of an  $r \times r$ -system,  $r \ll n_p$ , instead of the solution of an  $n_p \times n_p$ -system. For symmetric saddle-point systems with symmetric Schur complement preconditioner  $\widehat{S}^{-1}$ , the matrix  $E_L$  is the transpose of the matrix  $E_R$ , i.e., computing a low-rank approximation of  $E_L$  also yields a low-rank approximation of  $E_R$  and vice versa.

To improve the spectral properties of the preconditioner, we introduce a relaxation parameter  $\alpha$  that scales the initial preconditioner in the error matrices

$$E_{L,\alpha} = I_{n_p} - \alpha \widehat{S}^{-1} S, \quad E_{R,\alpha} = I_{n_p} - \alpha \widehat{S} \widehat{S}^{-1}. \quad (8)$$

We find the relaxed updates by scaling  $\widehat{S}^{-1}$  with  $\alpha$  and replacing  $E_L$ ,  $E_R$  with  $E_{L,\alpha}$ ,  $E_{R,\alpha}$ , respectively, in (7). Replacing  $E_{L,\alpha}$ ,  $E_{R,\alpha}$  with a low-rank approximation  $E_{L,\alpha} \approx U_{L,\alpha,r} V_{L,\alpha,r}^T$ ,  $E_{R,\alpha} \approx U_{R,\alpha,r} V_{R,\alpha,r}^T$ , with  $U_{L,\alpha,r}$ ,  $V_{L,\alpha,r}$ ,  $U_{R,\alpha,r}$ ,  $V_{R,\alpha,r} \in \mathbb{R}^{n_p \times r}$  and applying the Sherman-Morrison-Woodbury formula leads to the update formulas

$$\begin{aligned} S_{L,\alpha,r}^{-1} &= \alpha \left( I_{n_p} + U_{L,\alpha,r} \left( I_r - V_{L,\alpha,r}^T U_{L,\alpha,r} \right)^{-1} V_{L,\alpha,r}^T \right) \widehat{S}^{-1}, \\ S_{R,\alpha,r}^{-1} &= \alpha \widehat{S}^{-1} \left( I_{n_p} + U_{R,\alpha,r} \left( I_r - V_{R,\alpha,r}^T U_{R,\alpha,r} \right)^{-1} V_{R,\alpha,r}^T \right). \end{aligned}$$

We compute the low-rank approximations of  $E_{L,\alpha}$ ,  $E_{R,\alpha}$  with one of the two low-rank factorization algorithms described in Section 2.2. Both matrices depend on the Schur complement  $S = BA^{-1}B^T$  which we cannot evaluate exactly due to the large-scale inverse  $A^{-1}$ . We approximate  $S$  by replacing the inverse  $A^{-1}$  with the preconditioner  $\widehat{A}^{-1}$  which in our case is given by `ilu(0)`. Note that we also need to be able to evaluate  $\widehat{A}^{-T}v$  for the low-rank approximation by the power range finder of Section 2.2. This gives us the approximate error matrices

$$\widetilde{E}_{L,\alpha} = I - \alpha \widehat{S}^{-1} \widetilde{S}, \quad \widetilde{E}_{R,\alpha} = I - \alpha \widetilde{S} \widehat{S}^{-1}. \quad (9)$$

We approximate  $\widetilde{E}_{L,\alpha} \approx Q_{L,r,\alpha} N_{L,r,\alpha}^T$ ,  $\widetilde{E}_{R,\alpha} \approx Q_{R,r,\alpha} N_{R,r,\alpha}^T$ , with  $Q_{L,r,\alpha}$ ,  $N_{L,r,\alpha}$ ,  $Q_{R,r,\alpha}$ ,  $N_{R,r,\alpha} \in \mathbb{R}^{n \times r}$  with Algorithm 1 or Algorithm 2 and obtain the preconditioners

$$\widehat{S}_{L,\alpha,r}^{-1} = \alpha \left( I_{n_p} + Q_{L,\alpha,r} \left( I_r - N_{L,\alpha,r}^T Q_{L,\alpha,r} \right)^{-1} N_{L,\alpha,r}^T \right) \widehat{S}^{-1}, \quad (10)$$

$$\widehat{S}_{R,\alpha,r}^{-1} = \alpha \widehat{S}^{-1} \left( I_{n_p} + Q_{R,\alpha,r} \left( I_r - N_{R,\alpha,r}^T Q_{R,\alpha,r} \right)^{-1} N_{R,\alpha,r}^T \right). \quad (11)$$

Applying these updated preconditioners to a vector requires multiplication with thin rectangular matrices, solving a  $r \times r$  system, and scaling a vector of size  $n_p$  in addition to the application of the initial preconditioner  $\widehat{S}^{-1}$ . Numerical tests, as illustrated in Section 5.4, show that the relaxation parameter can have a significant impact on the convergence obtained with the updated preconditioner.



### 3.1 Construction and application of the update

We discuss the practical construction and application of the update scheme along with the respective computational complexities. In the following, we only consider the right update since the respective derivations for the left update are analogous. To construct the update, we compute the low-rank approximation  $\tilde{E}_{R,\alpha} \approx Q_{R,\alpha,r} N_{R,\alpha,r}^T$  with Algorithm 1 or 2. Furthermore, we (explicitly) precompute the small matrix  $H_r := I_r - N_{R,\alpha,r}^T Q_{R,\alpha,r} \in \mathbb{R}^{r \times r}$  and its exact LU factorization  $H_r = L_r U_r$ . Algorithm 3 summarizes these steps for the setup of the preconditioner update.

---

**Algorithm 3** Construction of the update.
 

---

**Input:** System matrix  $\mathcal{A} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$ , preconditioner  $\hat{A}^{-1}$ , initial Schur complement preconditioner  $\hat{S}^{-1}$ , rank  $r$ , relaxation parameter  $\alpha$ , number of power iterations  $q$ , oversampling parameter  $k$ .

**Output:** Matrices  $Q_{R,\alpha,r}$ ,  $N_{R,\alpha,r}$ ,  $L_r$ ,  $U_r$  defining the updated preconditioner  $\hat{S}_{R,\alpha,r}^{-1}$  (11).

- 1 Define the approximate Schur complement  $\tilde{S} := B \hat{A}^{-1} B^T$ .
- 2 Compute a low-rank approximation of  $E_\alpha := I - \alpha \tilde{S} \hat{S}^{-1} \approx Q_{\alpha,r} N_{R,\alpha,r}^T$  with Algorithm 1 (or 2).
- 3 Compute  $H_r = I_r - N_{R,\alpha,r}^T Q_{R,\alpha,r}$ .
- 4 Compute an LU factorization of  $H_r = L_r U_r$ .
- 5 Define the preconditioner  $\hat{S}_{R,\alpha,r}^{-1} = \alpha \hat{S}^{-1} (I_{n_p} + Q_{R,\alpha,r} (L_r U_r)^{-1} N_{R,\alpha,r}^T)$ .

---

We now discuss the setup costs for the update scheme. The randomized low-rank approximation with Algorithm 1 requires creating the  $n_p \times l$  random test matrix, i.e., generating  $n_p l$  random numbers. This costs  $N_{\text{fillG}} = \mathcal{O}(n_p l)$ . The most expensive part are the  $(q + 1)l$  matrix–vector products with  $\tilde{E}_{R,\alpha}$  and its transpose. The application of  $\tilde{E}_{R,\alpha}$  requires the application of  $\tilde{S}$  and  $\hat{S}^{-1}$  to a vector and a vector addition. Hence, one application of  $\tilde{E}_{R,\alpha}$  or its transpose to a vector costs

$$N_{\text{mult}\tilde{E}\alpha} = N_{\text{mult}\tilde{S}} + N_{\text{mult}\hat{S}^{-1}} + \mathcal{O}(n_p).$$

Furthermore, we orthogonalize  $2(q + 1)$  times with modified Gram-Schmidt with reorthogonalization if needed. The setup costs  $N_{\text{setup}}$  hence add up to

$$\begin{aligned} N_{\text{setup}} &= N_{\text{fillG}} + 2(q + 1)l N_{\text{mult}\tilde{E}\alpha} + 2(q + 1)N_{\text{orth}} + N_{\text{LU}} \\ &= \mathcal{O}(n_p l) + 2(q + 1)l (N_{\text{mult}\tilde{S}} + N_{\text{mult}\hat{S}^{-1}} + \mathcal{O}(n_p)) \\ &\quad + 2(q + 1)\mathcal{O}(4n_p l^2) + \mathcal{O}(r^3). \end{aligned}$$

The required steps for the application of the preconditioner  $\hat{S}_{R,\alpha,r}^{-1}$  are summarized in Algorithm 4.

To apply the update to a vector we only need matrix–vector products with  $n_p \times r$ - and  $r \times n_p$ -matrices, solve two triangular systems of size  $r \times r$ , and add and scale a

---

**Algorithm 4** Application of the updated preconditioner.
 

---

**Input:** Vector  $v \in \mathbb{R}^{n_p}$ , matrices  $Q_{R,\alpha,r}$ , relaxation parameter  $\alpha$ ,  $N_{R,\alpha,r}$ ,  $L_r$ ,  $U_r$  defining the updated preconditioner  $\widehat{S}_{R,\alpha,r}^{-1}$  (11).

**Output:** Vector  $y = \widehat{S}_{R,\alpha,r}^{-1} v \in \mathbb{R}^{n_p}$ .

- 1 Compute  $z_r = N_{\alpha,r}^T v$ .
  - 2 Solve  $L_r U_r w_r = z_r$ .
  - 3 Compute  $x = Q_{R,\alpha,r} w_r$ .
  - 4 Compute  $x += v$ .
  - 5 Apply the initial preconditioner  $y = \widehat{S}^{-1} x$ .
  - 6 Scale with the relaxation parameter  $y *= \alpha$ .
- 

vector of size  $n_p$ . This sums up to the application costs

$$\begin{aligned} N_{\text{apply}} &= N_{\text{multQ}} + N_{\text{multN}} + N_{\text{solveLU}} + N_{\text{vecadd}} + N_{\text{vecscale}} + N_{\text{mult}\widehat{S}^{-1}} \\ &= \mathcal{O}(n_p r) + \mathcal{O}(n_p r) + \mathcal{O}(r^2) + \mathcal{O}(n_p) + \mathcal{O}(n_p) + N_{\text{mult}\widehat{S}^{-1}} \\ &= \mathcal{O}(n_p r) + N_{\text{mult}\widehat{S}^{-1}}. \end{aligned}$$

If  $r \leq n_p$  (typically we have  $r \ll n_p$ ), the additional cost introduced by the application of the update is linear in the number of pressure degrees of freedom  $n_p$ .

## 4 Error analysis

In this section, we analyze how the right update with relaxation acts on the error between the identity and the preconditioned Schur complement. With the Sherman-Morrison-Woodbury formula and the precomputed LU factorization of  $H_r = I_r - N_{R,\alpha,r}^T Q_{R,\alpha,r} = L_r U_r$ , we have

$$\left( I_{n_p} - Q_{R,\alpha,r} N_{R,\alpha,r}^T \right)^{-1} = I_{n_p} + Q_{R,\alpha,r} (L_r U_r)^{-1} N_{R,\alpha,r}^T =: I + X \quad (12)$$

for the multiplicative update that is used in (11) with  $X := Q_{R,\alpha,r} (L_r U_r)^{-1} N_{R,\alpha,r}^T$ . We rearrange (12) and obtain

$$\begin{aligned} I_{n_p} &= I_{n_p} - Q_{R,\alpha,r} N_{R,\alpha,r}^T + \left( I_{n_p} - Q_{R,\alpha,r} N_{R,\alpha,r}^T \right) X, \\ \Leftrightarrow Q_{R,\alpha,r} N_{R,\alpha,r}^T &= \left( I_{n_p} - Q_{R,\alpha,r} N_{R,\alpha,r}^T \right) X. \end{aligned} \quad (13)$$

Now, we analyze the error between the identity and the preconditioned Schur complement after applying the update. We obtain

$$\begin{aligned}
 E_{R,\alpha,r,\text{upd}} &:= \mathbf{I}_{n_p} - S\widehat{S}_{R,\alpha,r}^{-1} = \mathbf{I}_{n_p} - \alpha S\widehat{S}^{-1} \left( \mathbf{I}_{n_p} + Q_{R,\alpha,r} (L_r U_r)^{-1} N_{R,\alpha,r}^T \right) \\
 &\stackrel{(8)}{=} E_{R,\alpha} - (\mathbf{I}_{n_p} - E_{R,\alpha}) X \\
 &= E_{R,\alpha} - (\mathbf{I}_{n_p} - Q_{R,\alpha,r} N_{R,\alpha,r}^T + Q_{R,\alpha,r} N_{R,\alpha,r}^T - E_{R,\alpha}) X \\
 &\stackrel{(13)}{=} E_{R,\alpha} - Q_{R,\alpha,r} N_{R,\alpha,r}^T + \left( E_{R,\alpha} - Q_{R,\alpha,r} N_{R,\alpha,r}^T \right) X. \tag{14}
 \end{aligned}$$

The term

$$R_\alpha = E_{R,\alpha} - Q_{R,\alpha,r} N_{R,\alpha,r}^T \tag{15}$$

describes the quality of the low-rank approximation  $Q_{R,\alpha,r} N_{R,\alpha,r}^T$  of  $E_{R,\alpha}$  which not only depends on the chosen rank  $r$  but also on the quality of the approximation  $\widetilde{S} = B\widehat{A}^{-1}B^T$  to the Schur complement  $S = BA^{-1}B^T$  used in its construction. We substitute  $R_\alpha$  into (14) and obtain

$$\begin{aligned}
 E_{R,\alpha,r,\text{upd}} &= R_\alpha + R_\alpha X = R_\alpha (\mathbf{I}_{n_p} + X), \\
 \|E_{R,\alpha,r,\text{upd}}\| &= \|R_\alpha (\mathbf{I}_{n_p} + X)\| \leq \|R_\alpha\| \|\mathbf{I}_{n_p} + X\|.
 \end{aligned}$$

Hence, the error after applying the update does not only depend on the low-rank approximation error  $R_\alpha$  but also on the product  $R_\alpha X$ . In terms of the ranks of the involved matrices and assuming best low-rank approximations, this implies

$$\text{rank}(R_\alpha) = \text{rank}(E_{R,\alpha}) - r$$

but only

$$\text{rank}(E_{R,\alpha,r,\text{upd}}) \leq \text{rank}(R_\alpha) + r = \text{rank}(E_{R,\alpha}),$$

i.e., the low-rank update typically does not reduce the rank of the error between the identity matrix and the (updated) preconditioned Schur complement.

## 5 Numerical results

In this section, we present numerical results obtained for a model problem describing buoyancy-driven atmospheric dynamics involving the incompressible Navier–Stokes equations. We start with a description of this model problem in Section 5.1. Then, we analyze the efficiency of the update and the influence of update parameters on convergence and computational times. In Section 5.2, we compare different initial preconditioners to justify our particular choice. Then, in Section 5.3, we discuss the influence of the parameters in the randomized low-rank approximation, i.e., the number of power iterations and the oversampling parameter. Furthermore, in Section 5.4, we

investigate the quality of the low-rank update in dependence of the relaxation parameter and the update rank.

## 5.1 Model problem

This subsection introduces the considered test model that describes atmospheric dynamics. It is based on the Navier–Stokes equations that describe the dynamics by the pressure  $p$  and the velocity  $\mathbf{u}$

$$\begin{aligned} \partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \nabla \cdot \left[ \frac{2}{\text{Re}} \boldsymbol{\varepsilon}(\mathbf{u}) \right] + \nabla p &= \mathbf{f}(\mathbf{u}), \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

on a domain  $\Omega \subset \mathbb{R}^d$ . Here,  $\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$  denotes the strain rate tensor and  $\partial_t$  the partial time derivative. The non-dimensional number  $\text{Re}$  denotes the Reynolds number. It is given as  $\text{Re} = \frac{\rho U L}{\mu}$  where  $U$  is the velocity scale,  $L$  is the length scale,  $\rho$  is the density of the fluid, and  $\mu$  is the dynamic viscosity of the fluid. The forcing  $\mathbf{f}(\mathbf{u})$  includes the buoyancy and the Coriolis force. We consider two three-dimensional domains, a cube and a thin shell. Figure 1 shows example meshes for both geometries. We choose periodic boundary conditions at the sides of the cube that mimic the shell geometry of the atmosphere. Furthermore, we have no-slip conditions at the lower boundary  $\partial\Omega_{\text{lower}}$  and no-flux conditions at the upper boundary  $\partial\Omega_{\text{upper}}$

$$\begin{aligned} \mathbf{u}|_{\partial\Omega_{\text{lower}}} &= 0, \\ \mathbf{n} \cdot \nabla \mathbf{u}|_{\partial\Omega_{\text{upper}}} &= 0. \end{aligned}$$

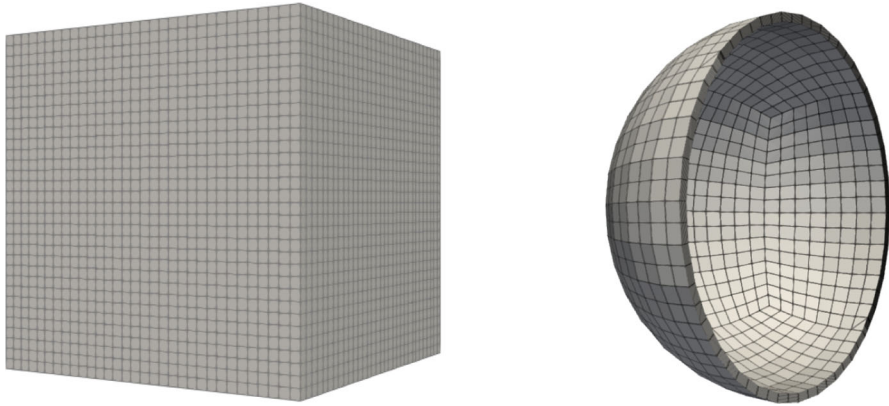
For the shell geometry, we use no-slip conditions at the inner boundary  $\partial\Omega_{\text{inner}}$  and no-flux conditions at the outer boundary  $\partial\Omega_{\text{outer}}$

$$\begin{aligned} \mathbf{u}|_{\partial\Omega_{\text{inner}}} &= 0, \\ \mathbf{n} \cdot \nabla \mathbf{u}|_{\partial\Omega_{\text{outer}}} &= 0. \end{aligned}$$

The Navier–Stokes equations are temporally discretized with the semi-implicit Euler method. All terms are discretized implicitly except for the forcing. For spatial discretization, we use Taylor-Hood elements of the lowest order on hexahedral cells. Linearization with the Picard iteration leads to (a sequence of) saddle-point systems of the form

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$

for each Picard iteration and each time step. The matrix blocks  $B$  and  $B^T$  correspond to the divergence and the gradient. The matrix block  $A$  is dominated by the velocity mass matrix. Furthermore, it includes diffusion and advection, both scaled with the time step length. The advection is scaled with the inverse Reynolds number. We set



(a) Grid of a cube.

(b) Grid of a shell.

Fig. 1 Example grids

the length scale to 1.0 m, the velocity scale to  $0.01 \text{ m s}^{-1}$ , the density to  $1.20 \text{ kg m}^{-3}$ , and the dynamic viscosity to  $1.82 \cdot 10^{-5} \text{ kg m}^{-1} \text{ s}^{-1}$ . This leads to a Reynolds number of about 708.8. We test our preconditioners for the system that we obtain during the first time step after five Picard iterations.

We solve the block system with FGMRes with a restart length of 40 iterations. We stop when the relative residual drops below  $10^{-8}$ . In the following, we show results for the right update scheme since we are using a right preconditioner.

The model and solvers are implemented in C++ using `deal.II 9.4.0` [25, 26] and `Trilinos 13.1` [27]. The numerical results are obtained on an Intel Xeon Gold 6240 processor with 2.60 GHz and 18 cores and 32 GB RAM.

## 5.2 Initial preconditioners

We start with a comparison of different preconditioners  $\hat{A}^{-1}$  and  $\hat{S}^{-1}$  for the block triangular preconditioner in (2) without any low-rank updates. A standard choice as a preconditioner  $\hat{A}^{-1}$  is given by multigrid methods. However, due to the advection in our system, algebraic multigrid methods may fail. Furthermore, we require the transposed application of the preconditioner for the setup of the update scheme which is not defined for multigrid techniques for unsymmetric systems. Hence, we have chosen an incomplete factorization (`ilu(0)`) from `Ifpack`, included in `Trilinos`. We compare this choice with an algebraic multigrid from the `ML` package in `Trilinos` where we use one V-cycle with a Gauss-Seidel smoother and a direct solver on the coarsest level.

Both SIMPLE and LSC Schur complement preconditioners in (3) and (4) require the (approximate) solution of Poisson-type systems  $BD^{-1}B^T$  with a diagonal matrix  $D$ . We compare an inner iterative solver, the conjugate gradient method with a relative (stopping) tolerance of 0.1, with an incomplete Cholesky factorization (`ic(0)`). The

inner solver approach once more is not suitable for the computation of low-rank updates since the multiplication with the transpose is not available. For the incomplete Cholesky factorization, we have to compute the matrix-matrix product  $M_{BDuBT} \approx BD_u^{-1}B^T$  for the LSC preconditioner (3) and  $M_{BABT} \approx B \operatorname{diag}(A)^{-1}B^T$  for the SIMPLE preconditioner (4). The respective timings are shown in Table 1 under the column headings “setup matmat”.

We test the initial preconditioners with relaxation parameters  $\alpha$  from (9) that have been deemed as favorable by our analysis in Section 5.4.

Table 1 shows timings for the setup and the solver using the initial preconditioners that have been described above. We observe that the setup times for the algebraic multigrid method are significantly higher than the setup times for the incomplete LU factorization, and the resulting solvers fail to converge for the cube domain. Setup costs for the shell are lower than those on the cube in view of the smaller problem size. The costs for setting up the initial Schur complement preconditioner (columns “matmat” and “IC”) are significantly lower than the setup costs for the preconditioner of the leading block  $A$ . While using an inner CG method for solving the Poisson-like problems in the Schur complement preconditioner reduces the outer number of FGMRES iterations, it increases the solution time since each step becomes more costly. Given these (admittedly very brief) comparisons to some other preconditioner components, we conclude that our initial preconditioner setup appears to be reasonable.

### 5.3 Parameters for the low-rank approximation

In this subsection, we discuss the influence of the parameters in the randomized low-rank approximation on the update, the number of power iterations  $q$  and the oversampling parameter  $k$ . We expect a higher accuracy of the low-rank approximation and thus a “better” preconditioner by increasing these parameters. The cheapest setup costs are obtained if we do not use power iterations or oversampling, i.e., we set  $q = k = 0$  in Algorithm 1. However, this choice leads to less accurate low-rank approximations and may hence deteriorate the update. Usually, a few oversampling vectors improve the accuracy of randomized low-rank approximations. Here, however, we orthogonalize the basis vectors of the approximate range, i.e., the columns of  $Q$  in (5), with the modified Gram-Schmidt method without pivoting. This means that the obtained vectors are not sorted and oversampling does not improve the low-rank approximation. Hence we set the oversampling parameter to zero for all the following tests.

Table 2 shows results obtained for varying the number of power iterations from  $q = 0$  to  $q = 3$  for an update with rank  $r = 20$  (and no oversampling, i.e.,  $k = 0$ ). We show the median and the standard deviation obtained from 200 test runs. We set the relaxation parameter  $\alpha$  in (11) to the optimal values in the set  $\{0.6, 0.7, \dots, 2.9\}$  obtained for an update rank of  $r = 20$ , see the next subsection for more details. Furthermore, we also include results obtained for a low-rank update using the Arnoldi method as well as results obtained with relaxation but without any update.

We observe that the number  $q$  of power iterations has a significant impact on the setup time. For the shell geometry,  $q = 1$  power iterations for both the SIMPLE and the

**Table 1** Setup and solver times (in seconds) as well as iteration counts for different initial preconditioners  $\hat{A}^{-1}$ ,  $\hat{S}_{LSC}^{-1}$  (3),  $\hat{S}_{SIMPLE}^{-1}$  (4) for the block preconditioner (2) for the shell system ( $n_u = 78438$ ,  $n_p = 3474$ ) and the cube system ( $n_u = 107811$ ,  $n_p = 4913$ )

|       | $\alpha$ | ILU, IC     |        |      | AMG, IC     |             |       | ILU, CG |             |             |        |        |      |    |       |
|-------|----------|-------------|--------|------|-------------|-------------|-------|---------|-------------|-------------|--------|--------|------|----|-------|
|       |          | setup (sec) | iters  |      | solve (sec) | setup (sec) | iters |         | solve (sec) | setup (sec) | iters  |        |      |    |       |
|       |          | ILU(A)      | matmat | IC   | AMG(A)      | matmat      | IC    | ILU(A)  | matmat      | IC          | AMG(A) | matmat | IC   |    |       |
| Shell | LSC      | 1.0         | 7.0    | 0.27 | 0.07        | 31          | 3.96  | 30.7    | 0.27        | 0.07        | 76     | 21.60  | 7.0  | 18 | 86.20 |
| Shell | SIMPLE   | 1.6         | 7.0    | 0.27 | 0.09        | 22          | 1.99  | 30.7    | 0.27        | 0.09        | 38     | 9.58   | 7.0  | 18 | 33.83 |
| Cube  | LSC      | 1.2         | 10.1   | 0.37 | 0.10        | 77          | 13.72 | 31.7    | 0.37        | 0.10        | —      | —      | 10.1 | 29 | 13.46 |
| Cube  | SIMPLE   | 1.9         | 10.1   | 0.37 | 0.12        | 47          | 5.99  | 31.7    | 0.37        | 0.12        | —      | —      | 10.1 | 36 | 8.68  |

**Table 2** Results for varying the number of power iterations  $q$  with rank  $r = 20$ , oversampling parameter  $k = 0$ , for the systems on the shell ( $n_u = 78438$ ,  $n_p = 3474$ ) and the cube ( $n_u = 107811$ ,  $n_p = 4913$ )

| (a) Shell, LSC, $\alpha = 1.0$ .    |           |              | (b) Cube, LSC, $\alpha = 1.2$ .    |           |           |              |              |
|-------------------------------------|-----------|--------------|------------------------------------|-----------|-----------|--------------|--------------|
| 0                                   | 27 (0.54) | 2.72 (0.02)  | 3.75 (0.09)                        | 0         | 78 (0.52) | 3.81 (0.02)  | 15.36 (0.27) |
| 1                                   | 19 (0.50) | 5.37 (0.02)  | 2.67 (0.08)                        | 1         | 64 (3.97) | 7.52 (0.08)  | 12.50 (0.77) |
| 2                                   | 19 (0.51) | 8.01 (0.02)  | 2.65 (0.08)                        | 2         | 44 (6.99) | 11.23 (0.07) | 8.83 (1.31)  |
| 3                                   | 18 (0.47) | 10.66 (0.02) | 2.56 (0.08)                        | 3         | 41 (2.99) | 14.94 (0.03) | 8.20 (0.57)  |
| Arnoldi                             | 26        | 1.73         | 3.37                               | Arnoldi   | 73        | 2.42         | 13.00        |
| no update                           | 32        | 0            | 4.08                               | no update | 77        | 0            | 13.70        |
| (c) Shell, SIMPLE, $\alpha = 1.6$ . |           |              | (d) Cube, SIMPLE, $\alpha = 1.9$ . |           |           |              |              |
| 0                                   | 21 (0.32) | 1.39 (0.01)  | 2.02 (0.05)                        | 0         | 66 (4.89) | 1.94 (0.04)  | 8.29 (0.62)  |
| 1                                   | 17 (0.50) | 2.72 (0.02)  | 1.64 (0.05)                        | 1         | 78 (1.43) | 3.79 (0.06)  | 9.76 (0.28)  |
| 2                                   | 16 (0.27) | 4.06 (0.02)  | 1.54 (0.04)                        | 2         | 73 (1.47) | 5.72 (0.10)  | 9.13 (0.23)  |
| 3                                   | 16 (0.43) | 5.40 (0.02)  | 1.54 (0.05)                        | 3         | 67 (2.39) | 5.72 (0.14)  | 8.35 (0.33)  |
| Arnoldi                             | 20        | 0.93         | 1.77                               | Arnoldi   | 59        | 1.29         | 7.27         |
| no update                           | 22        | 0            | 1.94                               | no update | 48        | 0            | 5.97         |

The needed iterations and computational times in seconds are given as the median (standard deviation) obtained from 200 test runs.  $t_{\text{setup}}$  includes only the costs for the update, not those for the initial preconditioner



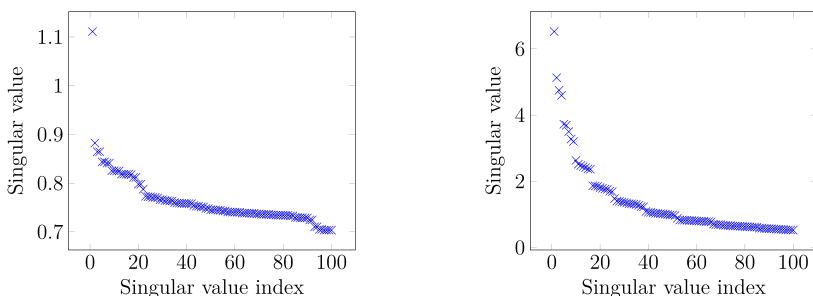
LSC preconditioner seem to be sufficient concerning the reduction of iteration counts, a further increase of  $q$  has hardly any impact on the iteration counts. The observations for the cube geometry are very different: the number of iterations with the updated LSC preconditioner is significantly reduced for each of the first two power iteration whereas the updated SIMPLE preconditioner even becomes worse by applying updates. Here, for  $q = 0$  to  $q = 3$ , increasing the number of power iterations does not decrease the iteration counts. A larger number of power iterations might be needed in this case but leads to even higher setup times. It is also interesting to note that the standard deviations of the iteration counts are larger for the cube geometry.

In an attempt to explain the somewhat disappointing results for updated preconditioners on the cube, we take a look at the singular values of the error matrix. Figure 2 compares the decay in approximate singular values for the shell and the cube. Figure 2a shows the slow decay of the 100 largest approximate singular values of  $\tilde{E}_{R,\alpha}$  with  $\alpha = 1.9$  from (9) for the cube using the SIMPLE preconditioner. It is seen that the singular values decay rather slowly for the cube which makes it difficult to resolve the dominant subspace of the range. Figure 2b shows that the singular values of the shell system decay faster compared those of the cube.

#### 5.4 Rank and relaxation parameter

Now, we test the update scheme for varying the rank  $r$  and the relaxation parameter  $\alpha$  in (11). Figure 3 shows the number of iterations obtained with the updated LSC and SIMPLE preconditioners on the shell and the cube. We illustrate results for low-rank updates obtained both with the randomized method and the Arnoldi iteration. The rank  $r$  is varied from 0 to 60 with step size 5 and relaxation parameters  $\alpha$  are varied from 0.6 to 2.9 with step size 0.1. The white color refers to the number of iterations obtained with a preconditioner without relaxation and update. Green areas denote a reduction of iteration counts and red areas denote an increase in iterations.

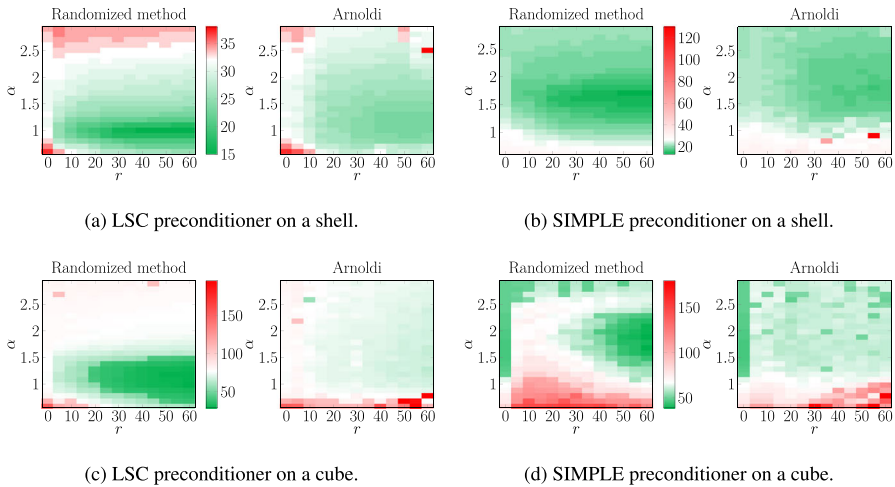
We observe that the relaxation parameter has a significant impact on the needed number of iterations. The interval of “optimal”  $\alpha$  seems to depend mainly on the



(a) Cube with relaxation parameter  $\alpha = 1.9$ .

(b) Shell with relaxation parameter  $\alpha = 1.6$ .

**Fig. 2** The 100 largest singular values of  $\tilde{E}_{R,\alpha}$  from (9) for the two domains using the SIMPLE preconditioner



**Fig. 3** Iteration counts for the Oseen systems on the cube ( $n_u = 107811$ ,  $n_p = 4913$ ) and the thin shell ( $n_u = 78438$ ,  $n_p = 3474$ ) for varying the parameters  $\alpha$ ,  $r$  in (11). For each subfigure, the left image shows results obtained with the randomized low-rank approximation (with  $q = 3$  power iterations and oversampling parameter  $k = 0$ ) and the right image shows results obtained with the Arnoldi iteration. The white color refers to results obtained without relaxation and updating

initial preconditioner but not on the system (at least for the considered test systems). Furthermore, the range of (nearly) optimal relaxation parameters is relatively large. For the LSC preconditioner, the optimal relaxation parameter is close to 1.0. For the SIMPLE preconditioner, the optimal relaxation parameter lies in the interval  $[1.6, 1.8]$  for the shell and in the interval  $[1.6, 2.3]$  for the cube for ranks  $r \geq 45$ . In these settings, the low-rank update computed with the randomized method leads to lower iteration counts compared to the Arnoldi method. However, the setup costs with the randomized method using  $q = 3$  power iterations are about six times higher than the setup costs of the update with the Arnoldi method.

Table 3 shows the needed number of iterations, setup, solver and total costs for different relaxation parameters and ranks for updates computed with the randomized method and the Arnoldi iteration. The hyphen denotes that the solver did not converge within 1000 iterations. Here, we do not use power iterations for the randomized method (i.e.,  $q = 0$ ). We observe that the relaxation parameter accelerates convergence. For the coarse shell, the low-rank update decreases iterations counts further but increases the total computational time. For the other systems, the update does not improve the preconditioners further. The results obtained with the Arnoldi iteration lead to lower iteration counts in most cases and about 35% lower setup costs.

## 6 Outlook

We have derived a right and a left low-rank update for a relaxed Schur complement preconditioner. The update vectors are computed with a (randomized) low-rank

**Table 3** Number of needed FGMRes iterations and setup, solver and total times (seconds) for the Oseen systems obtained with  $q = 0$  power iterations and  $k = 0$  oversampling vectors

|  | $\alpha$ | $r$ | Randomized method |       |       |        | Arnoldi |       |       |       |
|--|----------|-----|-------------------|-------|-------|--------|---------|-------|-------|-------|
|  |          |     | iters             | setup | solve | total  | iters   | setup | solve | total |
| shell ( $k_n = 5\text{e-}4$ , $n_u = 78438$ , $n_p = 3474$ )     |          |     |                   |       |       |        |         |       |       |       |
| LSC  | 1.0      | 0   | 32                | 0     | 4.08  | 4.08   | 32      | 0     | 4.08  | 4.08  |
|  | 1.0      | 20  | 27                | 2.66  | 3.45  | 6.11   | 26      | 1.73  | 3.37  | 5.10  |
|  | 1.0      | 40  | 22                | 5.30  | 2.82  | 8.12   | 23      | 3.39  | 2.95  | 6.34  |
| SIMPLE   | 1.0      | 0   | 26                | 0     | 2.27  | 2.27   | 26      | 0     | 2.27  | 2.27  |
|  | 1.6      | 0   | 22                | 0     | 1.94  | 1.94   | 22      | 0     | 1.94  | 1.94  |
|  | 1.6      | 20  | 21                | 1.34  | 1.84  | 3.18   | 20      | 0.93  | 1.77  | 2.70  |
|  | 1.6      | 40  | 19                | 2.68  | 1.67  | 4.35   | 18      | 1.81  | 1.60  | 3.41  |
| shell ( $k_n = 2.5\text{e-}4$ , $n_u = 608454$ , $n_p = 26146$ ) |          |     |                   |       |       |        |         |       |       |       |
| LSC  | 1.0      | 0   | —                 | —     | —     | —      | —       | —     | —     | —     |
|  | 1.0      | 20  | —                 | —     | —     | —      | —       | —     | —     | —     |
|  | 1.0      | 40  | —                 | —     | —     | —      | —       | —     | —     | —     |
| SIMPLE   | 1.0      | 0   | 76                | 0     | 54.84 | 54.85  | 76      | 0     | 54.84 | 54.85 |
|  | 1.6      | 0   | 54                | 0     | 39.20 | 39.20  | 54      | 0     | 39.20 | 39.20 |
|  | 1.6      | 20  | 75                | 11.40 | 54.75 | 66.14  | 62      | 7.38  | 46.86 | 54.24 |
|  | 1.6      | 40  | 64                | 22.75 | 46.81 | 69.57  | 38      | 14.78 | 28.40 | 43.18 |
| cube ( $k_n = 5\text{e-}3$ , $n_u = 107811$ , $n_p = 4913$ )     |          |     |                   |       |       |        |         |       |       |       |
| LSC  | 1.0      | 0   | 78                | 0     | 13.90 | 13.90  | 78      | 0     | 13.90 | 13.90 |
|  | 1.2      | 0   | 77                | 0     | 13.70 | 13.70  | 77      | 0     | 13.70 | 13.70 |
|  | 1.2      | 20  | 78                | 3.71  | 13.85 | 17.57  | 73      | 2.42  | 13.00 | 15.42 |
|  | 1.2      | 40  | 78                | 7.45  | 13.86 | 21.31  | 71      | 4.72  | 12.66 | 17.38 |
| SIMPLE   | 1.0      | 0   | 68                | 0     | 8.64  | 8.64   | 68      | 0     | 8.64  | 8.64  |
|  | 1.9      | 0   | 48                | 0     | 5.97  | 5.97   | 48      | 0     | 5.97  | 5.97  |
|  | 1.9      | 20  | 67                | 1.89  | 8.27  | 10.61  | 59      | 1.29  | 7.27  | 8.56  |
|  | 1.9      | 40  | 57                | 3.80  | 7.07  | 10.87  | 62      | 2.51  | 7.62  | 10.13 |
| cube ( $k_n = 2.5\text{e-}3$ , $n_u = 823875$ , $n_p = 35937$ )  |          |     |                   |       |       |        |         |       |       |       |
| LSC  | 1.2      | 0   | —                 | —     | —     | —      | —       | —     | —     | —     |
|  | 1.2      | 20  | —                 | —     | —     | —      | —       | —     | —     | —     |
|  | 1.2      | 40  | —                 | —     | —     | —      | —       | —     | —     | —     |
| SIMPLE   | 1.0      | 0   | 78                | 0     | 76.93 | 76.93  | 78      | 0     | 76.93 | 76.93 |
|  | 1.9      | 0   | 70                | 0     | 68.97 | 68.97  | 70      | 0     | 68.97 | 68.97 |
|  | 1.9      | 20  | 71                | 15.52 | 71.33 | 86.85  | 63      | 10.08 | 63.97 | 74.05 |
|  | 1.9      | 40  | 71                | 31.11 | 71.33 | 102.44 | 61      | 20.23 | 61.70 | 81.93 |

The “setup” columns include only the costs for the update, not those for the initial preconditioner. The parameter  $k_n$  denotes the time step length

approximation of the difference between the identity and the scaled preconditioned approximate Schur complement.

Relaxing the initial preconditioner has a significant impact on the convergence behavior of the iterative solver. The range of (nearly) optimal relaxation parameters seems to depend mainly on the initial preconditioner and is relatively broad.

We have observed that the update can decrease iteration counts but this is not guaranteed. The efficiency of the update depends on parameters for the underlying low-rank approximation, the relaxation parameter, and the update rank. Choosing the parameters for the low-rank approximation and the update rank is a trade-off between balancing setup costs and solver times. Low-rank updates computed with the randomized method using a few power iterations can decrease the iteration counts and solver times significantly but lead to higher setup costs and also higher total costs than low-rank updates computed with the Arnoldi iteration or the randomized method without power iterations. The drawback of high setup costs may be somewhat reduced when using parallel computing.

Future research concerns the adaptation of the initial preconditioners for the Picard iteration and for different time steps. The setup time needed to compute the low-rank update is lower than the setup time required for the initial preconditioners, so it may be beneficial to use a low-rank update instead of setting up a new preconditioner from scratch.

**Acknowledgements** The authors acknowledge the support by the Deutsche Forschungsgemeinschaft (DFG) within the Research Training Group GRK 2583 “Modeling, Simulation and Optimization of Fluid Dynamic Applications”. Furthermore, we are grateful for the input from Konrad Simon who provided the test cases.

**Author Contributions** All authors wrote and reviewed the main manuscript. Rebekka Beddig performed the numerical tests.

**Funding** Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) within the Research Training Group GRK 2583 “Modeling, Simulation and Optimization of Fluid Dynamic Applications”.

**Data Availability** Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

## Declarations

**Competing interests** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Benzi, M., Golub, G., Liesen, J.: Numerical solution of saddle point problems. *Acta Numer.* **14**, 1–137 (2005)
2. Elman, H., Silvester, D., Wathen, A.: *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press (2014). <https://doi.org/10.1093/acprof:oso/9780199678792.001.0001>
3. Vuik, C., Saghir, A.: The Krylov accelerated SIMPLE(R) method for incompressible flow. *Rep. Dep. Appl. Math. Phys.* 02-01 (2002). <http://resolver.tudelft.nl/uuid:c42d9354-75f2-4fe2-9f25-8f816bd3132a>
4. He, X., Vuik, C.: Comparison of some preconditioners for the incompressible Navier-Stokes equations. *Numer. Math. Theory Methods Appl.* **9**(2), 239–261 (2016). <https://doi.org/10.4208/nmtma.2016.m1422>
5. Elman, H., Howle, V., Shadid, J., Shuttleworth, R., Tuminaro, R.: Block preconditioners based on approximate commutators. *SIAM J. Sci. Comput.* **27**, 1651–1668 (2006)
6. Kay, D., Lohin, D., Wathen, A.: A preconditioner for the steady-state Navier-Stokes equations. *SIAM J. Sci. Comput.* **24**, 237–256 (2002). <https://doi.org/10.1137/S106482759935808X>
7. Börm, S., Le Borne, S.:  $\mathcal{H}$ -LU factorization in preconditioners for augmented Lagrangian and grad-div stabilized saddle point systems. *Internat. J. Numer. Methods Fluids* **68**, 83–98 (2010)
8. Le Borne, S., Rebholz, L.: Preconditioning sparse grad-div/augmented Lagrangian stabilized saddle point systems. *Comput. Vis. Sci.* **16**, 259–269 (2015)
9. Fiordilino, J.A., Layton, W., Rong, Y.: An efficient and modular grad-div stabilization. *Comput. Methods Appl. Mech. Engrg* **335**, 327–346 (2018). <https://doi.org/10.1016/j.cma.2018.02.023>
10. Cousins, B.R., Borne, S.L., Linke, A., Rebholz, L.G., Wang, Z.: Efficient linear solvers for incompressible flow simulations using Scott-Vogelius finite elements. *Numer. Methods Partial Differ. Equat.* **29**, 1217–1237 (2013)
11. He, X., Vuik, C.: Efficient and robust Schur complement approximations in the augmented Lagrangian preconditioner for the incompressible laminar flows. *J. Comput. Phys.* **408**, 109286 (2020)
12. Le Borne, S.: Hierarchical matrix preconditioners for the Oseen equations. *Comput. Vis. Sci.* **11**, 147–157 (2008)
13. Borne, S.L.: Preconditioned nullspace method for the two-dimensional Oseen problem. *SIAM J. Sci. Comput.* **31**, 2494–2509 (2009)
14. Bergamaschi, L.: A survey of low-rank updates of preconditioners for sequences of symmetric linear systems. *Algorithms* **13**(4) (2020). <https://doi.org/10.3390/a13040100>
15. Zanetti, F., Bergamaschi, L.: Scalable block preconditioners for linearized Navier–Stokes equations at high Reynolds number. *Algorithms* **13**(8) (2020). <https://doi.org/10.3390/a13080199>
16. Zheng, Q., Xi, Y., Saad, Y.: A power Schur complement low-rank correction preconditioner for general sparse linear systems. *SIAM J. Matrix Anal. Appl.* 659–682 (2021). <https://doi.org/10.1137/20M1316445>
17. Al Daas, H., Rees, T., Scott, J.: Two-level Nyström-Schur preconditioner for sparse symmetric positive definite matrices. *SIAM J. Sci. Comput.* **43**(6), 3837–3861 (2021). <https://doi.org/10.1137/21M139548X>
18. Zheng, Q.: Domain decomposition based preconditioner combined local low-rank approximation with global corrections. *Comput. Math. Appl.* **114**, 41–46 (2022). <https://doi.org/10.1016/j.camwa.2022.03.006>
19. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review* **53**(2), 217–288 (2011). <https://doi.org/10.1137/090771806>
20. Martinsson, P.-G., Tropp, J.: Randomized numerical linear algebra: Foundations and algorithms. *Acta Numer.* **29** (2020). <https://doi.org/10.1017/S0962492920000021>
21. Murphy, M.F., Golub, G.H., Wathen, A.J.: A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.* **21**(6), 1969–1972 (2000). <https://doi.org/10.1137/S1064827599355153>
22. Ipsen, I.C.F.: A note on preconditioning nonsymmetric matrices. *SIAM J. Sci. Comput.* **23**(3), 1050–1051 (2001). <https://doi.org/10.1137/S1064827500377435>
23. Trottenberg, U., Oosterlee, C., Schüller, A.: *Multigrid*. Elsevier Academic Press, London (2001)
24. Saad, Y.: *Iterative Methods for Sparse Linear Systems*, 2nd edn. Society for Industrial and Applied Mathematics, (2003). <https://doi.org/10.1137/1.9780898718003>

25. Arndt, D., Bangerth, W., Feder, M., Fehling, M., Gassmöller, R., Heister, T., Heltai, L., Kronbichler, M., Maier, M., Munch, P., Pelteret, J.-P., Sticko, S., Turcksin, B., Wells, D.: The deal.II library, version 9.4. *J. Numer. Math.* **30**(3), 231–246 (2022). <https://doi.org/10.1515/jnma-2022-0054>
26. Arndt, D., Bangerth, W., Davydov, D., Heister, T., Heltai, L., Kronbichler, M., Maier, M., Pelteret, J.-P., Turcksin, B., Wells, D.: The deal.II finite element library: Design, features, and insights. *Comput. Math. Appl.* **81**, 407–422 (2021). <https://doi.org/10.1016/j.camwa.2020.02.022>
27. Heroux, M.A., Bartlett, R.A., Howle, V.E., Hoekstra, R.J., Hu, J.J., Kolda, T.G., Lehoucq, R.B., Long, K.R., Pawlowski, R.P., Phipps, E.T., Salinger, A.G., Thornquist, H.K., Tuminaro, R.S., Willenbring, J.M., Williams, A., Stanley, K.S.: An overview of the Trilinos project. *ACM Trans. Math. Softw.* **31**(3), 397–423 (2005). <https://doi.org/10.1145/1089014.1089021>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.